Методические Указания

Для лабораторных работ
по дисциплине
«Анализ и проектирование алгоритмов»

Технический Университет Молдовы Департамент Программная Инженерия и Автоматика

Кишинёв, 2026

СОДЕРЖАНИЕ

ЛАБОРАТОРНАЯ РАБОТА № 1 3
ЛАБОРАТОРНАЯ РАБОТА № 2 9
ЛАБОРАТОРНАЯ РАБОТА № 3 14
ЛАБОРАТОРНАЯ РАБОТА № 4 20
ЛАБОРАТОРНАЯ РАБОТА № 5 24
ЛАБОРАТОРНАЯ РАБОТА № 6 37
ЛАБОРАТОРНАЯ РАБОТА № 7 55

ЛАБОРАТОРНАЯ РАБОТА № 1

Тема: Сравнение времени работы алгоритмов

Цель*:* Временной анализ функций

Краткие теоретические сведения

Для решения задачи часто приходится выбирать метод как из числа алгоритмов различных по принципу своей работы, так и из числа возможных реализаций одного алгоритма. За конечное число шагов, при разных исходных данных, все они приведут к правильному решению задачи. Но из всего спектра вариантов, следует выбирать оптимальные методы.

Оценка функции трудоемкости алгоритма называется *сложностью алгоритма* и позволяет определить предпочтения в использовании того или иного алгоритма для больших значений размерности исходных данных.

Выделяют *временную* и *пространственную* сложность. Временная сложность определяется количеством элементарных операций (инструкций), совершаемых алгоритмом для решения им поставленной задачи. Пространственная сложность измеряется объемом используемой алгоритмом памяти. Далее будет рассматриваться только временная сложность.

При анализе поведения функции трудоемкости алгоритма часто используют принятые в математике асимптотические обозначения, позволяющие показать порядок скорости роста временной функции, пренебрегая при этом конкретными значениями.

В асимптотическом анализе приняты следующие обозначения:

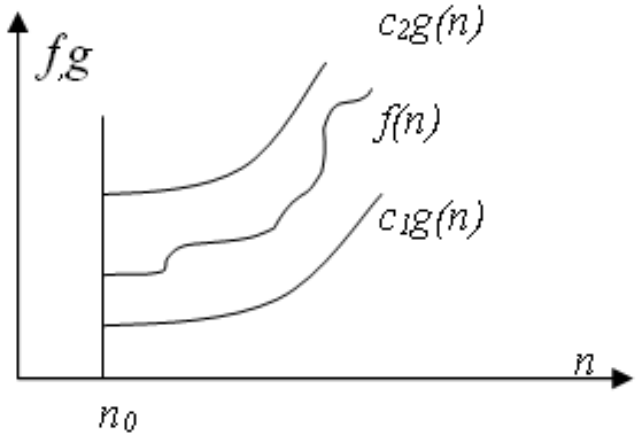
1 Асимптотически точная оценка θ (тетта)

Пусть $f(n)$ и $g(n)$ – положительные функции положительного аргумента, $n \geq 1$ (количество объектов на входе и количество операций – положительные числа), тогда:

$f(n) = \theta(g(n))$, если существуют положительные $c_1 > 0, c_2 > 0$ и n_0 такие, что:

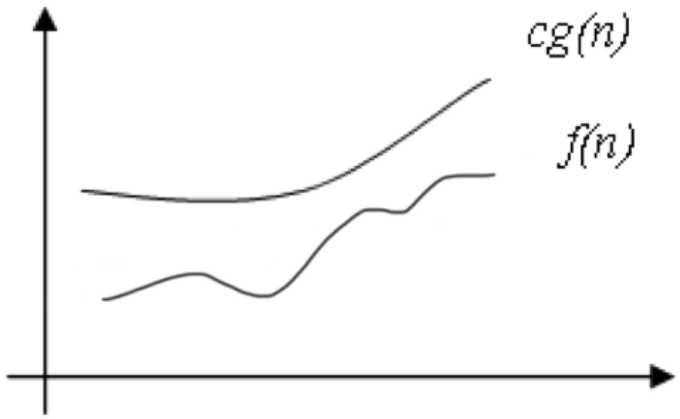
$c_1 g(n) \leq f(n) \leq c_2 g(n)$ при $\forall n \geq n_0$

Обычно говорят, что при этом функция $g(n)$ является асимптотически точной оценкой функции $f(n)$, так как по определению функция $f(n)$ не отличается от функции $g(n)$ с точностью до постоянного множителя.



2 Верхняя оценка O̓ (омикрон)

В отличие от оценки θ, оценка O̓ требует только, чтобы функция $f(n)$ не превышала $g(n)$ начиная с $\forall n \geq n_0$, с точностью до постоянного множителя:



$\exists$ $c > 0, n_0 > 0$:

$0 \leq f(n) \leq cg(n), \quad \forall n \geq n_0$

Вообще, запись $O̓(g(n))$ обозначает класс функций, таких, что все они растут не быстрее, чем функция $g(n)$ с точностью до постоянного множителя, поэтому иногда говорят, что $g(n)$ мажорирует функцию $f(n)$.

Например, для всех функций:

$$f(n) = 1/n,$$

$$f(n) = 12,$$

$$f(n) = 3n + 17,$$

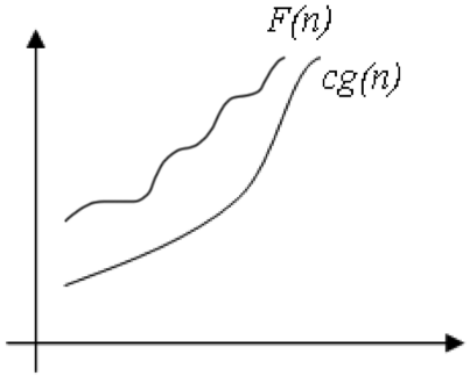
$$f(n) = n * ln(n),$$

$$f(n) = 6n^2 + 24n + 77$$

будет справедлива оценка $O(n^2)$, то есть все эти функции не будут расти быстрее чем n^2.

3 Нижняя оценка Ω (Омега)

В отличие от оценки O, оценка Ω является оценкой снизу, то есть определяет класс функций, которые растут не медленнее, чем $g(n)$ с точностью до постоянного множителя:



$$\exists\ c > 0,\ n_0 > 0:$$

$$cg(n) \le f(n) \le \infty$$

при $\forall n \ge n_0$

Например, запись $\Omega(n * ln(n))$ обозначает класс функций, которые растут не медленнее, чем $g(n) = n * ln(n)$, в этот класс попадают все полиномы со степенью большей единицы, равно как и все степенные функции с основанием большим единицы.

Запись **f(n)** = O **(g(n))** означает, что с ростом **n** отношение **f(n)/g(n)** остаётся ограниченным.

Если к тому же

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

то пишем **f(n)** = σ **(g(n))** (читается «эф от эн есть σ малое от же от эн»).

Формально говоря, **f(n)** = σ **(g(n))**, если для всякого положительного ε > 0 найдётся такое n_0 что

$0 \le f(n) \le \varepsilon g(n)$ при $\forall\ n \le n_0$.

Примеры:

$$1/3x^3 + 5x = O(x^3)$$

$$x^2 = o(x^3)$$

Задание:

I. Для алгоритма, решающий задачу размера n, определить время выполнения этой задачи. Найти его для функций, перечисленных в таблице и заполнить таблицу. Построить графики роста функции и расставить функции в порядке возрастания.

f(n) \ n	1	2	4	8	16	24
log n						
ln n						
$\sqrt{n}$						
n						
1/n						
n log n						
n^2						
n^3						
2^n						
n!						
n/ log n						
log log n						
log (n!)						
n^n						

II. Сравнение алгоритмов

Для всех клеток следующей таблицы ответьте «да» или «нет» на вопрос о том, можно ли записать A как O, σ, Ω, ω или Θ от B ($k \geq 1$, $\varepsilon > 0$, c, $m > 1$ – некоторые константы). Приведите расчеты.

A	B	O	σ	Ω	ω	Θ
$\text{Log}^k n$	**n^ε**					
n^k	**c^n**					
$\sqrt{n}$	**$n^{\sin n}$**					
2^n	**$2^{n/2}$**					
$n^{\log m}$	**$m^{\log n}$**					
log (n!)	**log (n^n)**					

Пример:

Дано:

$$f(n) = n, g(n) = 2n^2.$$

Определить асимптоту роста между функциями.

Решение:

Согласно определению $f(n) = \theta(g(n))$ записываем неравенство по формуле:

$c_1 2n^2 \leq n \leq 2n^2 c_2$

Делим все неравенство на $2n^2$

$$c_1 \leq \frac{1}{2n} \leq c_2,$$

Выбираем какое-либо $n_0 > 0$.

Пусть $n_0 = 1$.

Находим c_2:

$c_2 = 1/2$,

при $\forall n \geq n_0$ $c_1 \rightarrow 0$, поэтому выбираем оценку для O

$n = O(2n^2)$.

Проверяем для o малое, применяя свойства предела

$$\lim_{(n->\infty)} \frac{n}{2n^2} = 0.$$

Получаем оценку для o малое

$n = o(2n^2)$.

Результаты записываем в таблицу.

A	B	O	o	Ω	ω	Θ
n	$2n^2$	+	+	–	–	–

III. Сравнение скорости роста

Расположите следующие функции в порядке увеличения скорости роста. Приведите расчеты. Постройте графики этих функции для наглядности.

1. $2^{\log m}$ $(\sqrt{2})^{\log m}$
2. 2^{2**n} 2^{2**n+1}
3. n! (n+1)!
4. log (n!) (log n)!
5. $(3/2)^n$ e^n $n*2^n$

6.	n	2n	n^3	n^2
7.	$n^{1/\log n}$	$n^{\log\log n}$	$(\log n)^{(\log n)}$	
8.	ln n	ln ln n		
9.	4 log n	$\log^2 n$	n log n	
10.	`√ (log n)	2√ (2log n)		

Контрольные вопросы:

1. Что означает – асимптота роста? Какие обозначения вы знаете?
2. В чем отличие между **Ο̈** и **ο̈, Ω** и **ω?**
3. Какие способы существуют для сравнения скорости роста двух алгоритмов?
4. Каковы недостатки и преимущества каждого способа?
5. Как по графикам можно определить какой из алгоритмов имеет более быстрый рост?
6. Верно ли высказывание, что один и тот же алгоритм может иметь различную эффективность про малых и больших размерности входа? П

Содержание отчета

Задание 1.

1. Заполнение таблицы.
2. Построение графиков роста функций.
3. Упорядочивание функций в порядке возрастания их роста.

Задание 2.

1. Заполнение таблицы
2. Приведение алгоритма сравнения функций.
3. Построение графиков роста функций.
4. Обоснование результатов.

Задание 3.

1. Алгоритм сравнения функций.
2. Обоснование полученных результатов.

Задание 4.

1. Ответы на контрольные вопросы.
2. Вывод по лабораторной работе.

ЛАБОРАТОРНАЯ РАБОТА № 2

Тема: Методы быстрой сортировки и их сравнительный анализ

Цель: Реализация и анализ алгоритмов сортировки

Краткие теоретические сведения

Перечень быстрых методов сортировки

1. Сортировка вставками (INSERTIONSORT).
2. Сортировка методом слияния (MERGESORT).
3. Быстрая сортировка (QUICKSORT).
4. Сортировка подсчетом. (COUNTINGSORT).
5. Восходящая поразрядная сортировка. (MSD RADIXSORT).
6. Нисходящая поразрядная сортировка. (LSD RADIXSORT).

Существует еще множество алгоритмов сортировки, которые основаны на вышеприведенных. (TimSort, ShellSort, BucketSort и другие).

Задания к лабораторной работе

1. Для трех любых алгоритмов сортировки записать алгоритм с помощью псевдокода.

Пример:

INSERTION-SORT(A)

```
1.  for j←2 to length[A]
2.       key ← A[j]
3.       ·> добавить A[j] к отсортированной части A[1..j]
4.       i ← j - 1
5.      while i > 0 & A[i] > key
6.           A[i+1] ←A[i]
7.           i ← i - 1
8.      A[i+1] ← key
```

Таким же образом выполняется для двух других алгоритмов.

2. Продемонстрировать работу алгоритмов (вывести массив на каждом шагу и трассировку этапов) для заданного массива:

№ варианта	Массив А
1	21, 31, 24, 10, 31, 41, 8, 2, 57, 15
2	15, 25, 10, 16, 5, 30, 57, 15, 8, 2
3	3, 42, 31, 84, 89, 5, 73, 40, 44, 32
4	1, 99, 33, 5, 3, 37, 44, 32, 57, 15
5	7, 5, 2, 96, 37, 54, 49, 8, 44, 32
6	56, 83, 25, 9, 55, 1, 10, 90, 8, 2
7	21, 35, 56, 41, 24, 88, 6, 11, 57, 15
8	67, 82, 36, 34, 55, 89, 4, 18, 44, 32
9	17, 3, 7, 80, 23, 11, 64, 32, 8, 2
10	49, 28, 74, 56, 81, 14, 76, 44, 57, 15
11	87, 22, 57, 13, 9, 11, 31, 89, 8, 2
12	16, 54, 33, 34, 17, 19, 90, 2, 44, 32
13	53, 50, 60, 78, 5, 77, 86, 12, 57, 15
14	3, 5, 16, 9, 31, 15, 32, 87, 8, 2, 44
15	15, 25, 10, 16, 3, 5, 30, 57, 1, 44, 32
16	1, 99, 33, 5, 3, 37, 7, 44, 32, 57, 15
17	56, 83, 55, 9, 55, 1, 10, 12, 90, 44, 32
18	67, 3, 82, 36, 34, 55, 83, 4, 18, 8, 2
19	49, 28, 74, 56, 81, 19, 14, 76, 4, 57, 15
20	16, 54, 33,5, 58, 17, 19, 90, 2, 44, 32
21	3, 5, 16, 9, 31, 15, 32, 35, 87, 8, 2
22	21, 30, 7, 24, 10, 31, 41, 8, 2, 57, 15
23	83, 3, 42, 31, 84, 89, 5, 73, 46, 8, 2
24	7, 5, 2, 17, 96, 37, 54, 49, 8, 44, 32
25	11, 35, 56, 53, 41, 24, 88, 6, 21, 57, 15
26	17, 38, 7, 80, 3, 11, 64, 32, 36, 8, 2
27	87, 22, 57, 63, 13, 9, 11, 31, 89, 44, 32
28	53, 50, 60, 78, 5, 77, 11, 86, 12, 57, 15
29	9, 6, 93, 84, 67, 33, 52 ,99, 81, 8, 2
30	3, 5, 16, 9, 31, 15, 32, 87, 8, 2, 44

Пример (вывод массива, и трассировка шагов):

Трассировка работы алгоритма INSERTION-SORT

INSERTION-SORT(A) A = (5, 2, 4, 6, 1, 3)



1 for j←2 to 6

2 do key←A[2] = 2

3 ›добавить A[2] к отсортированной части A[1] = 5

4 i←j-1 = 1

5 while 1 > 0 and A[1] = 5 > key = 2
6 do A[2] ← A[1] = 5
7 i ← i - 1 = 0
5 while i > 0
8 A[1] ← key = 2 A = (2, 5, 4, 6, 1, 3)
i j

1 j = 3, 6
2 do key ← A[3] = 4
3 ›добавить A[3] к отсортированной части A[1, 2] = (2, 5)
4 i ← i - 1 = 2
5 while 2 > 0 and A[2] = 5 > key = 4
6 do A[3]←A[2] = 5
7 i ← i - 1 = 1
5 while 1 > 0 and A[1] = 2 > key = 4
8 A[2] ← key = 4 A = (2, 4, 5, 6, 1, 3)
i j

1 j = 4, 6
2 do key ← A[4] = 6
3 ›добавить A[3] к отсортированной части A[1, 2, 3] = (2, 4, 5)
4 i ← i - 1 = 3
5 while 3 > 0 and A[3] =5 > key = 6
8 A[4] ← key = 6 A = (2, 4, 5, 6, 1, 3)
i j

и т. д.

3. Провести временной анализ алгоритмов сортировки (проанализировав псевдокод программ, то есть теоретически), и сравнить данные с практическими. Построить графическую зависимость времени от размерности массива для каждого из алгоритмов на массиве от n случайных элементов, не менее 3-х раз с размерностями больше, чем на предыдущем шагу.

Пример:

Теоретически (по оценке псевдокода):

Время работы сортировки вставкой $T(n) = \theta(n^2)$

Время работы сортировки слиянием $T(n) = \theta(n \log n)$

Время работы быстрой сортировки $T(n) = \theta(n \log n)$

Практически (результаты работы программы)

Время работы алгоритмов:

– Количество элементов: 5000

  INSERTION SORT: 0.0372 с.

  MERGE SORT: 0.0093453 с.

  QUICK SORT: 0.003425 с.

– Количество элементов: 16000

  INSERTION SORT: 0.14725 с.

  MERGE SORT: 0.06683 с.

  QUICK SORT: 0.04999 с.

– Количество элементов: 246000

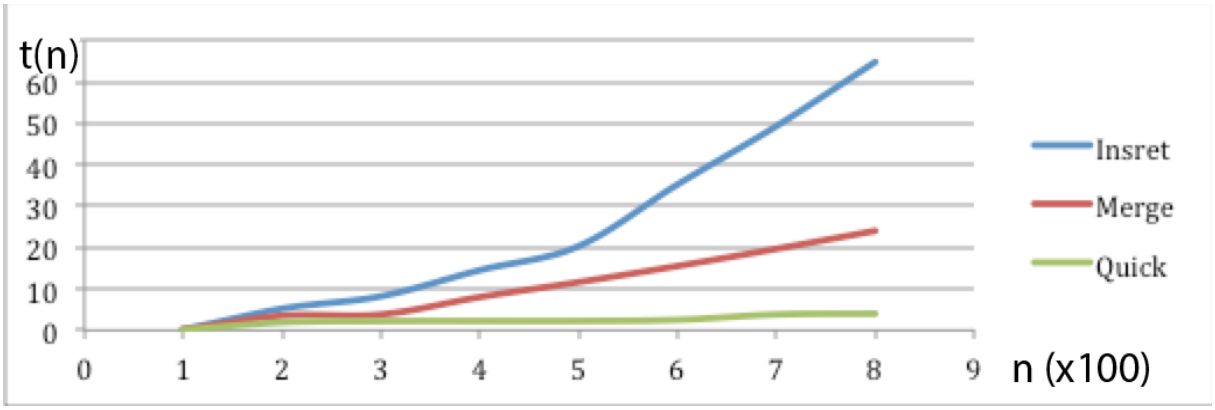
  INSERTION SORT: 0.24345 с.

  MERGE SORT: 0.033453 с.

  QUICK SORT: 0.035592 с.

Что совпадает с теоретическими данными.

Графики скорости работы алгоритмов:



4. Привести скриншоты результатов работы программы.

Пример:

[17 3 7 80 3 11 64 32] -> INSERTIONSORT -> [3 3 7 11 17 32 64 80] за 0.0002763с

[17 3 7 80 3 11 64 32] -> MERGESORT -> [3 3 7 11 17 32 64 80] за 0.0000492с

[17 3 7 80 3 11 64 32] -> QUICKSORT -> [3 3 7 11 17 32 64 80] за 0.0000187с

Примечание

При использовании старой библиотеки <time.h> может возникнуть проблема, что на экран выводится 0.0 – это связано с неточностью работу встроенных функций. Предлагается использовать <chrono> — это библиотека имеется только в стандарте C++11 и выше, но она позволяет вычислить очень точное время работы.

5. Привести листинг программы (исходный код программы).

Содержание отчета:

1. Псевдокод алгоритмов.
2. Трассировка работы алгоритмов для заданного массива.
3. Сравнительное время работы (теоретическое).
4. Графики скорости роста работы алгоритмов.
5. Результаты работы.
6. Вывод.
7. Листинг программы.

Контрольные вопросы:

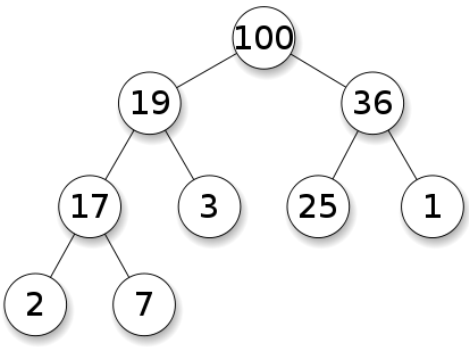
1. Какие виды сортировок вы знаете?
2. Почему разные алгоритмы работают с разной эффективностью?
3. Каким образом измеряется время работы алгоритма?
4. Для чего требуется улучшение эффективности алгоритмов?
5. Алгоритмы с какой скоростью роста являются применимы на практике, какие нет?

ЛАБОРАТОРНАЯ РАБОТА № 3

Тема: Организация очереди с приоритетом с помощью кучи

Цель: Изучение основного свойства кучи, ее построение и сортировка. Организация очереди с приоритетом с помощью кучи.

Краткие теоретические сведения



Куча (Пирамида) — это специализированная структура данных типа дерево, которая удовлетворяет *свойству кучи:* если B является узлом-потомком узла A, то ключ $(A) \geq$ ключ (B). Для данной выше кучи, 17 и 3 являются потомками 19, и следовательно, их ключи (значения) меньше 19-ти. Куча является самым эффективным методом реализации так называемой *очереди с приоритетом* – типа данных, в котором реализованы 2 основные операции – извлечение максимального элемента, и добавление нового элемента. Алгоритм сортировки кучей является одним из самых эффективных алгоритмов, как по времени, так и по памяти, и широко применяется в современных вычислительных системах.

Над кучами обычно проводятся следующие операции

- поддержание основного свойства кучи (HEAPIFY);
- построение кучи (BUILD-HEAP);
- сортировка массива (HEAP-SORT);
- взятие наибольшего элемента массива (HEAP-EXTRACT-MAX);
- добавление элемента (HEAP-INSERT).

Задания к лабораторной работе:

1. Привести, используя псевдокоды, алгоритмы основных операций над кучей:
   - поддержание основного свойства кучи (HEAPIFY);
   - построение кучи (BUILD-HEAP);
   - сортировка массива (HEAP-SORT);
   - взятие наибольшего элемента массива (HEAP-EXTRACT-MAX);

– добавление элемента (HEAP-INSERT);

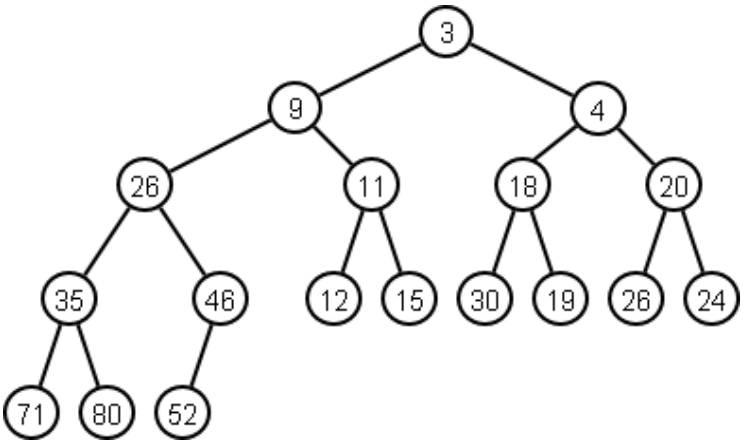
2. Реализовать алгоритм построения кучи.
3. Реализовать алгоритм сортировки с помощью кучи.
4. Организовать очередь с приоритетами и показать работу процедур:
   – HEAP-INSERT (A,key) для кучи A;
   – HEAP-EXTRACT-MAX для этой кучи.

№ варианта	Массив A	Элемент добавления, key
1	21, 31, 24, 10, 31, 41, 8, 2, 57, 15	26
2	15, 25, 10, 16, 5, 30, 57, 15, 8, 2	20
3	3, 42, 31, 84, 89, 5, 73, 40, 44, 32	40
4	1, 99, 33, 5, 3, 37, 44, 32, 57, 15	30
5	7, 5, 2, 96, 37, 54, 49, 8, 44, 32	50
6	56, 83, 25, 9, 55, 1, 10, 90, 8, 2	60
7	21, 35, 56, 41, 24, 88, 6, 11, 57, 15	40
8	67, 82, 36, 34, 55, 89, 4, 18, 44, 32	50
9	17, 3, 7, 80, 23, 11, 64, 32, 8, 2	20
10	49, 28, 74, 56, 81, 14, 76, 44, 57, 15	60
11	87, 22, 57, 13, 9, 11, 31, 89, 8, 2	30
12	16, 54, 33, 34, 17, 19, 90, 2, 44, 32	20
13	53, 50, 60, 78, 5, 77, 86, 12, 57, 15	65
14	3, 5, 16, 9, 31, 15, 32, 87, 8, 2, 44	20
15	15, 25, 10, 16, 3, 5, 30, 57, 1, 44, 32	27
16	1, 99, 33, 5, 3, 37, 7, 44, 32, 57, 15	40
17	56, 83, 55, 9, 55, 1, 10, 12, 90, 44, 32	60
18	67, 3, 82, 36, 34, 55, 83, 4, 18, 8, 2	50
19	49, 28, 74, 56, 81, 19, 14, 76, 4, 57, 15	50
20	16, 54, 33,5, 58, 17, 19, 90, 2, 44, 32	40
21	3, 5, 16, 9, 31, 15, 32, 35, 87, 8, 2	34
22	21, 30, 7, 24, 10, 31, 41, 8, 2, 57, 15	25
23	83, 3, 42, 31, 84, 89, 5, 73, 46, 8, 2	50
24	7, 5, 2, 17, 96, 37, 54, 49, 8, 44, 32	40
25	11, 35, 56, 53, 41, 24, 88, 6, 21, 57, 15	55
26	17, 38, 7, 80, 3, 11, 64, 32, 36, 8, 2	60
27	87, 22, 57, 63, 13, 9, 11, 31, 89, 44, 32	60
28	53, 50, 60, 78, 5, 77, 11, 86, 12, 57, 15	70
29	9, 6, 93, 84, 67, 33, 52 ,99, 81, 8, 2	70
30	3, 5, 16, 9, 31, 15, 32, 87, 8, 2, 44	20

Содержание отчета

1. Показать выбранный массив в виде неотсортированной кучи.

Пример:



2. Привести псевдокоды функций.

Например:

HEAP-EXTRACT (A)

1 if heap-size [A] < 1

2 error “Очередь пуста”

3. max ← A [1]

4. A [1] ← A [heap-size]

5. heap-size ← heap-size - 1

6. HEAPIFY (A, 1)

7. return max

3. Продемонстрировать эти алгоритмы на примере.

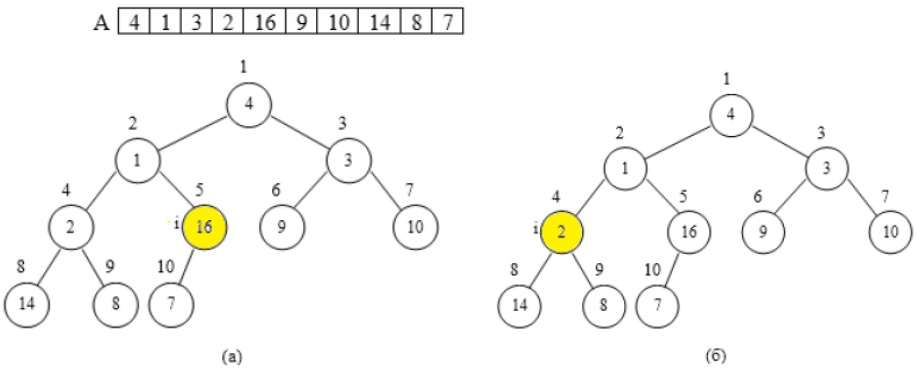
BUILD-HEAP (A)

1 heap-size[A] ← length[A]

2 for i ← ⌊length / 2⌋ down to 1

3 HEAPIFY(A, i)

Ниже показана трассировка алгоритма построения кучи:



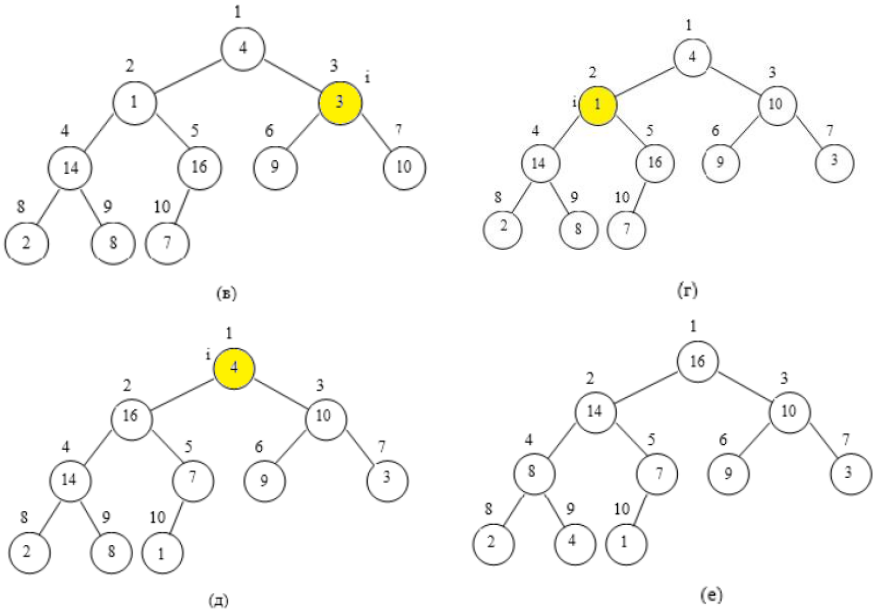


Рисунок 3.1 - Работа процедуры BUILD-HEAP (A)

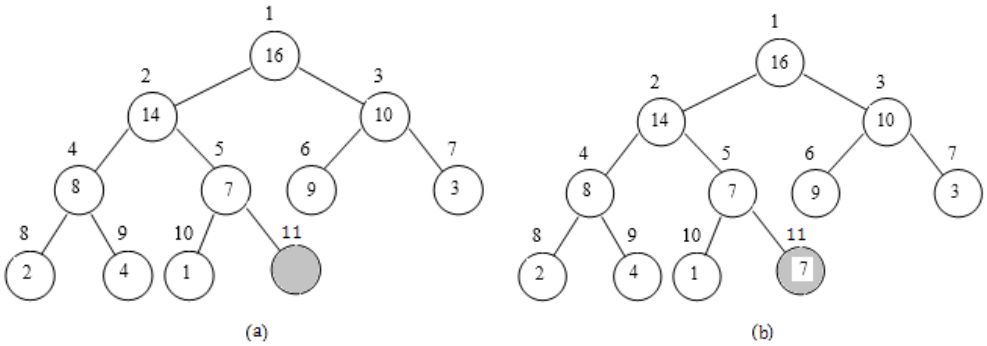
HEAP-INSERT(A, key)

1. heap-size [A] ← heap-size [A] + 1
2. i ← heap-size [A]
3. while i > 1 и A [PARENT(i)] < key
4. do A[i] ← A [PARENT(i)]
5. i ← PARENT(i)
6. A[i] ← key

Ниже показана трассировка алгоритма HEAP-INSERT(A, key):

HEAP-INSERT (A, 15)

1 heap-size[A] ← 11
2 i ← 11
3 11 > 1 & A[PARENT(11)] < 15 (7 < 15)
4 A[11] ← A[PARENT(5)] (A[11] ← 7)
5 i ← 5
3 5 > 1 & A[PARENT(5)] < 15 (14 < 15)
4 A[5] ← A[PARENT(2)] (A[5] ← 14)
5 i ← 2
3 2 > 1 & A[PARENT(2)] < 15 (16 < 15)
6 A[2] ← 15



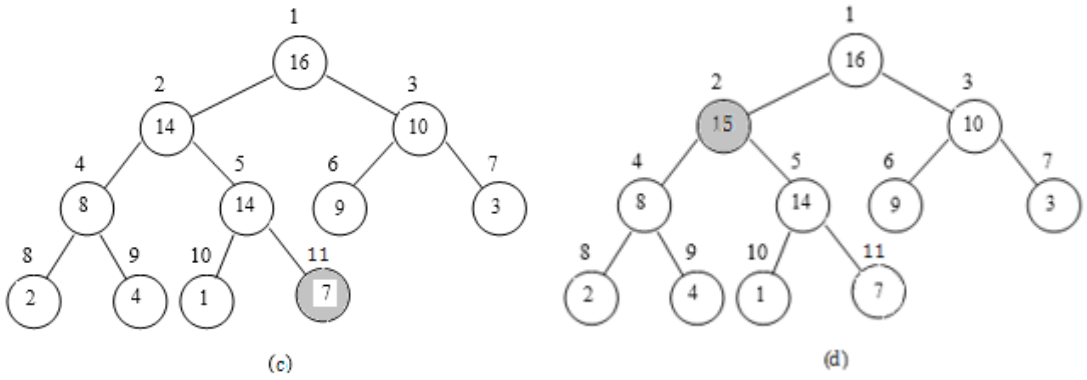


Рисунок 3.2 - Работа процедуры HEAP-INSERT. Добавляет элемент с ключевым значением 15 (темный кружок обозначает место для этого элемента)

Подобным образом показать трассировку работы для других алгоритмов. Можно продемонстрировать работу программы на массиве, без использования деревьев.

Из примера работы HEAP-INSERT видно, что есть возможность реализовать функцию BUILD-HEAP на основе HEAP-INSERT, так как после каждого шага HEAP-INSERT получается куча.

Дополнительное задание: реализовать оба вида алгоритма (BUILD-HEAP (A) и BUILD-HEAP′(A)) и сравнить их работу.

4. Провести сравнение времени работы алгоритма сортировки кучей с алгоритмами сортировки, рассмотренными в лабораторной работе № 2.

Сформулировать выводы.

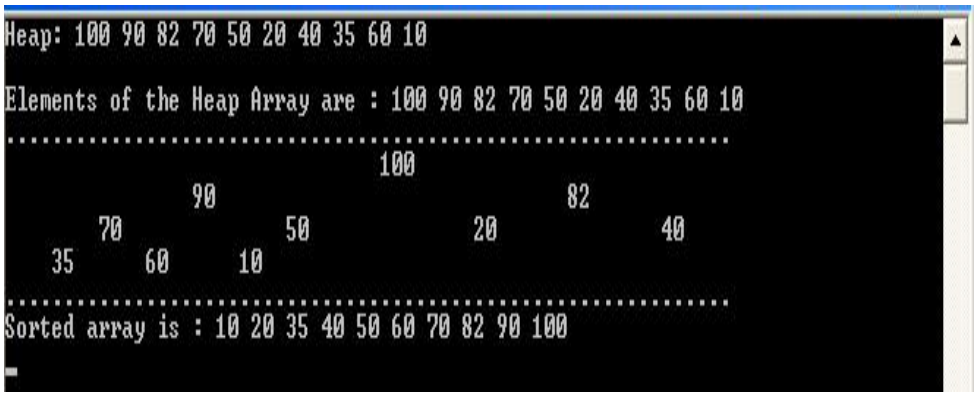
Пример:

Сравнение алгоритмов реализовано теоретически

Алгоритм	**Худший случай**	**Среднее время**	**Худший случай**
QUICKSORT	n log n	n log n	n^2
MERGE-SORT	n log n	n log n	n log n
HEAPSORT	n log n	n log n	n log n
INSERTION-SORT	n	n^2	n^2

Откуда видно, что HEAP-SRORT - самый эффективный (если учитывать затрачиваемую память).

5. Результаты работы программ.



6. Вывод

7. Листинг программы

Контрольные вопросы:

1. Каким образом работает сортировка кучей и другие процедуры для работы с кучей?
2. Какова эффективность (скорость работы) алгоритма сортировки?
3. Почему этот алгоритм является одним из наиболее эффективных алгоритмов?
4. Есть ли у него недостатки? Какие?
5. Какой способ построения кучи наиболее удобен и эффективен?

ЛАБОРАТОРНАЯ РАБОТА № 4

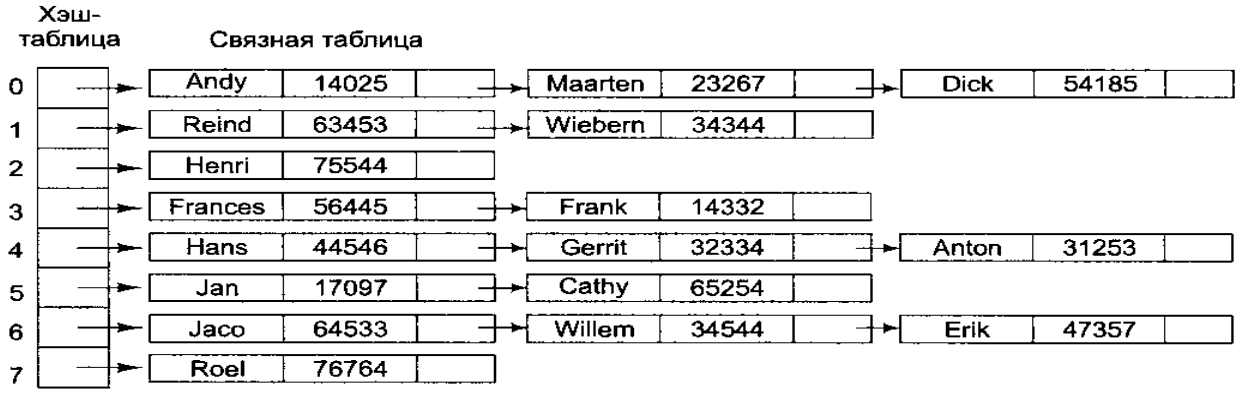
Тема: Хеш-таблицы

Цель: Изучить работу алгоритмов: прямой адресации, хеш-таблицы и открытой адресации. Реализовать перечисленные выше алгоритмы и провести их сравнительный анализ.

Краткие теоретические сведения

Хеш-таблица — это структура данных, реализующая интерфейс ассоциативного массива, а именно, она позволяет хранить пары (ключ, значение) и выполнять три операции: операцию добавления новой пары, операцию поиска и операцию удаления пары по ключу.

В общем виде хэш таблица выглядит следующим образом:



Для адресации (получения индекса элемента) используется хэш-функции.

В зависимости от кого, как они выглядят, различают 3 вида адресации:

1) **Прямая адресация** – каждому ключу соответствует равнозначная позиция в массиве. $h(k) = k$

2) Закрытая адресация (хэш-таблица)

Способы построения хеш-функции:

- деление с остатком:

  $h(k) = k \; mod \; m;$

- умножение:

  $h(k) = \lfloor m * ((k * A) \; mod \; 1) \rfloor,$

  где $A = 0{,}61803$

Например: ключ $k = 59$, то используя способ умножения построения хеш-функции, ключ будет записан в ячейку с индексом, равным $h(59) = 6$.

$$h(59) = \lfloor 20 * ((59 * 0{,}61803) mod\ 1) \rfloor = \lfloor 20 * ((36{,}312) mod\ 1) \rfloor =$$
$$= \lfloor 20 * 0{,}312 \rfloor = \lfloor 6{,}24 \rfloor = 6.$$

3) Открытая адресация

- *Линейное хеширование*

  $$h(k, i) = (h(k) + i)\ mod\ m.$$

  В данном случае считается количество попыток записать число (i), и, если ячейка по полученному индексу уже занята, пробуем еще раз, считая увеличенное число попыток (i + 1).

- *Квадратичное хеширование*

  $$h(k) = (h_1(k) + c_1 i + c_2 i^2)\ mod\ m\ ,\ h_1(k) = k\ mod\ m,$$

  c_1, c_2 - любые константы на выбор (например, $c_1 = 1, c_2 = 3$).

- *Двойное хеширование*

  $$h(k, i) = \ (h_1(k) + i h_2(k)) mod\ m,$$

  где $h_1(k) = k\ mod\ m,$

  $h_2(k) = (1 + k\ mod\ m')$, где $m' = m - 1 | m - 2.$

Задания к лабораторной работе

1. Пусть размер хеш-таблицы равен m = 20, а хеш-функция имеет вид

   h(k) = $\lfloor m * ((k * A)\ mod\ 1) \rfloor$, где A = 0,61803.

   В какие позиции попадут ключи (см. таблицу, столбец 3)?

2. Как будет выглядеть хеш-таблица с цепочками после того, как в неё последовательно поместили элементы с ключами (см. таблицу, столбец 2) (в указанном порядке)? Число позиций в таблице равно 9, хеш-функция имеет вид h(k) = k mod 9.

3. Выполните добавление ключей (в указанном порядке, см. таблицу. столбец 2) в хеш-таблицу с открытой адресацией размера m = 11. Для вычисления последовательности проб используется линейный метод с h'(k) = k mod m. Выполните то же задание, если используется квадратичный метод с той же h', c_1 = 1, c_2 = 3, а также для двойного хеширования с h_1 = h' и h_2(k) = 1 + (k mod (m - 1)).

Таблица

№ варианта	Массив	Ключи
1	10, 22, 31, 4, 15, 28, 17, 88, 59	61, 62, 63, 64 и 65
2	21, 31, 24, 10, 40, 41, 9, 2, 20	51, 52, 53, 54 и 55
3	15, 25, 10, 16, 5, 30, 57, 18, 19	41, 42, 43, 44 и 45
4	3, 42, 31, 84,9, 5, 73, 4, 13	31, 32, 33, 34 и 35
5	1, 99, 33, 5, 3, 37, 44, 32, 12	21, 22, 23, 24 и 25
6	7, 5, 2, 19, 37, 54, 49, 8, 10	46, 47, 48, 49 и 50
7	56, 18, 55, 9, 50, 1, 10, 88, 27	36, 37, 38, 39 и 40
8	11, 35, 56, 41, 24, 88, 6, 2, 26	44, 45, 46, 47 и 48
9	67, 76, 37, 34, 57, 13, 4, 1, 10	56, 57, 58, 59 и 60
10	17, 3, 9, 73, 6, 10, 64, 32, 21	66, 67, 68, 69 и 70
11	49, 32, 75, 53, 85, 19, 76, 44, 5	81, 82, 83, 84 и 85
12	80, 22, 57, 13, 9, 18, 31, 86, 42	43, 44, 45, 46 и 47
13	16, 5, 33, 34, 17, 29, 35, 6, 38	33, 34, 35, 36 и 37
14	59, 50, 4, 28, 5, 77, 86, 12, 22	53, 54, 55, 56 и 57
15	3, 5, 16, 9, 21, 14, 36, 44, 17	87, 86, 88, 89 и 90
16	3, 42, 31, 12, 33, 24, 13, 40, 10	63, 64, 65, 66 и 67
17	19, 37, 54, 49, 10, 1, 45, 18, 13	83, 84, 85, 86 и 87
18	17, 88, 59, 4, 15, 28, 10, 22, 31	65,63, 64, 62 и 61
19	9, 2, 20, 10, 40, 41, 21, 31, 24	55, 54, 53, 52 и 51
20	57, 18, 19, 16, 5, 30, 15, 25, 10	45, 44, 43, 42 и 41
21	73, 4, 13, 84,9, 5, 3, 42, 31	35, 34, 33, 32 и 31
22	44, 32, 12, 5, 3, 37, 1, 99, 33	25, 24, 23, 22 и 21
23	49, 8, 10, 19, 37, 54, 7, 5, 2	50, 49, 48, 47 и 46
24	10, 88, 27, 9, 50, 1, 56, 18, 55	40, 39, 38, 37 и 36
25	88, 6, 2, 26, 56, 41, 24, 56, 18	48, 47, 46, 45 и 44
26	6, 2, 26, 41, 24, 88, 11, 35, 56	60, 59, 58, 57 и 56
27	64, 32, 21, 73, 6, 10, 17, 3, 9	70, 69, 68, 67 и 66
28	76, 44, 5, 53, 85, 19, 80, 22, 57	85, 84, 83, 82 и 81
29	31, 86, 42, 13, 9, 18, 80, 22, 57	47, 46, 45, 44 и 43
30	35, 6, 38, 34, 17, 29, 16, 5, 33	37, 36, 35, 34 и 33

Содержание отчета

1. Привести все используемые алгоритмы в псевдокодах.

   Здесь необходимо указать использованные хэш-функции HASH(k), а также процедуры добавления элементов HASH-INSERT(k).

2. Продемонстрировать работу алгоритмов на примере (см. раздел «задание к лабораторной работе»).

   Для этого приведите окончательные результаты в виде массива с ключами, расставленными соответствующе правильному ответу.

Например:

Хэш функция: $h(k) = \lfloor 20 * (k * 0{,}61803 \bmod 1) \rfloor$.

Массив ключей: 51, 52, 63, 54 и 55

Конечный массив:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	1
		52					54	46		51								63	

3. Привести сравнительный анализ хэш-функций для каждого типа адресации и между разными типами адресации по их эффективности и реализации.
4. Результаты работы программ.
5. Вывод.
6. Листинг программ.

Контрольные вопросы:

1. Почему использование хэширования повышает эффективность работы со структурами данных?
2. Каким образом надо выбирать хэш-функцию?
3. Почему используются методы открытой адресации?
4. Какими недостатками обладает данный метод хранения данных (хэш-таблицы)?
5. Какой из данных видов хэширования вы считаете наиболее эффективным и почему?

ЛАБОРАТОРНАЯ РАБОТА № 5

Тема: Решение задач линейного программирования табличным симплексным методом и методом искусственных переменных.

Цель работы:

- изучить алгоритм табличного симплекс-метода;
- изучить алгоритм метода искусственных переменных;
- реализовать изученные алгоритмы либо в среде Excel, либо в какой-либо другой программной среде;
- проверить результат графическим методом.

Теоретическая часть

Переход к каноническому виду

Расширенная форма задачи ЛП

Пусть задача ЛП задана в общем виде:

$$W = \sum_{i=1}^{n} c_i x_i \to \max\ (\min)$$

при системе ограничений:

$$\begin{cases} \sum_i a_{ji} x_i = b_j, \text{ где } i = \overline{1, n}\ ,\ j = \overline{1, m_1}\ , \\ \sum_i a_{ji} x_i \le b_j, \text{ где } i = \overline{1, n},\ j = \overline{m_1+1, m_2}, & (*) \\ \sum_i a_{ji} x_i \ge b_j, \text{ где } i = \overline{1, n},\ j = \overline{+1, m}, & (**) \end{cases}$$

и условиях:

$$x_k \ge 0,\ k = \overline{1, r},\ r \le n.$$

Перейдем к каноническому виду:

1) При переходе от ограничений неравенств к равенствам вводят дополнительно неотрицательные переменные, количество которых равно числу неравенств x_{n+q}, где $q=\overline{m_1+1, m}$, которые прибавляются к левым частям ограничений (*) и вычитаются из левых частей ограничений (**). В целевую функцию дополнительные переменные входят с нулевыми коэффициентами. В таком виде задача ЛП называется **расширенной**.
2) Если в постановке задачи на некоторые переменные не наложены условия не отрицательности, то их представляют в виде разности неотрицательных переменных.

   Например, для задачи (*), (**) вводим:

   $x_k = x_k^1 - x_k^2,\ k>r$

   $x_k^1,\ x_k^2 \ge 0$

   и теперь получили канонический вид.

Метод симплекс-таблиц

Рассмотрим задачу ЛП с системой ограничений в каноническом виде:

$$\sum_{i=1}^{n} a_{ji}x_i = b_j, \quad j=\overline{1, m}$$

Опр. Если при неотрицательности правых частей каждое ограничение системы ограничений-равенств имеет переменную, входящую в левую часть, с коэффициентом тождественно равным 1, а во все остальные уравнения с коэффициентом равным 0, то говорят, что система представлена в **предпочтительном виде**.

Пример симплекс-таблицы, когда $A_1, A_2\ldots, A_m$ – единичный базис.

Таблица

C			C_1	C_2	C_3	…	C_i	C_e	C_n
	X	B	A_1	A_2	A_3	…	A_i	…	A_n
C_1	x_1	b_1	a_{11}	a_{12}	a_{13}		a_{1i}		a_{1n}
C_2	x_2	b_2	a_{21}	a_{22}	a_{23}		a_{2i}		a_{2n}
…									
C_j	x_j	b_j	a_{j1}	a_{j2}	a_{j3}		a_{ji}	a_{je}	a_{jn}
C_r		b_r					a_{ri}	a_{re}	
C_m	x_m	b_m	a_{m1}	a_{m2}	a_{m3}		a_{mi}		a_{mn}
	W	b_0	$Q_1 = a_{01}$	$Q_2 = a_{02}$	$Q_3 = a_{03}$		$Q_i = a_{0i}$		$Q_n = a_{0n}$

В первой строке и первом столбце записываются значения коэффициентов целевой функции $\mathbf{c}_i$; $i = \overline{1, n}$.

В столбце **X** записывают базисные переменные x_j; $j = \overline{1, m}$. Их значения определяются столбцом **B** свободных коэффициентов b_i, то есть $x_j^* = b_j$, $j \in I_б$.

В столбцах $A_1, A_2\ldots, A_n$ записывают значения с коэффициентом a_{jr}, $j = \overline{1, m}$, $i = \overline{1, n}$.

Последняя строка называется индексной. В ней записывают: в столбце **X** обозначение целой функции:

$$W = \sum_{j \in I} \mathbf{c}_j x_j,$$

а также оценки векторов $\theta_i = a_{01}$.

Алгоритм метода симплекс-таблиц

1. Рассчитывают и заполняют начальную таблицу (с дополнительным единичным базисом), включая индексную строку.
2. В качестве направляющего столбца выбирают столбец $\mathbf{A_i}$, для которого

   $\theta_I = a_{01} = \min\{\theta_e | \theta_e < 0\}$, $e = \overline{1, n}$, $e \notin I_б$

   (т.е. вектор $\mathbf{A_i}$ вводится в базис).
3. Направляющая строка $\mathbf{A_j}$ выбирается из условия:

$b_j/a_{ji} = \min\{b_r/a_{ri} \mid a_{ri} > 0\}$, $j \in I_б$, $i \notin I_б$

(перебираем строки i-го столбца).

4. Делают один шаг симплекс преобразования с направляющим элементом $\mathbf{a_{ji}}$.

Для элементов направляющей строки:

$$a_{je}^{(k+1)} = \begin{cases} a_{je}^{(k)} / a_{ji}^{(k)}; & e=\overline{1, n}, \quad e \neq i, \\ 1, & \text{если } e = i, \end{cases} \tag{1}$$

$b_j^{(k+1)} = b_j^{(k)} / a_{ji}^{(k)}$.

Для элементов направляющего столбца:

$$a_{ri}^{(k+1)} = \begin{cases} 0; & r \neq j;\ r \in I_б, \\ 1; & r = j;\ r \in I_б. \end{cases} \tag{2}$$

Для остальных элементов:

$$\begin{aligned} a_{re}^{(k+1)} &= a_{re}^{(k)} - (a_{je}^{(k)} / a_{ji}^{(k)})\, a_{ri}^{(k)}; \quad \text{где } e \neq i,\ r \neq j, \\ b_r^{(k+1)} &= b_r^{(k)} - (b_j^{(k)} / a_{ji}^{(k)})\, a_{ri}^{(k)}; \quad \text{где } r \neq j. \end{aligned} \tag{3}$$

Элементы индексной строки новой таблицы вычисляются по формулам:

$$W^{(k+1)} = b_0^{(k+1)} = b_0^k - b_j^k a_{0i}^k / a_{ji}^k, \tag{4}$$

$$\theta_e^{(k+1)} = a_{0e}^{(k+1)} = a_{0e}^k - a_{je}^k a_{0i}^k / a_{ji}^k, \quad e = \overline{1, n}. \tag{5}$$

Правильность вычислений контролируют по формулам непосредственного счета

$$W^{(k+1)} = b_0^{(k+1)} = \sum_{j \in I} c_j b_j^{(k+1)}. \tag{6}$$

Эта формула только для контроля:

$$\theta_e^{(k+1)} = a_{0e}^{(k+1)} = \sum_{j \in I} c_j a_{je}^{(k+1)} - c_e. \tag{7}$$

Примечание:

Так как базисные вектора образуют единичную матрицу, то:

$$\alpha_{je} = a_j$$

В столбце **X** заменяют x_j на x_i. А в столбце **C** заменяют c_j на c_i.

5. Если все оценки $\theta_e^{(k+1)} = a_{0e}^{(k+1)} > 0$, $e = \overline{1, n}$, (для задачи на max), то новое базисное решение

$x_j = b_j^{(k+1)}$, $j \in I_б$ оптимально.

В противном случае переходят к этапу 2 и выполняют очередную итерацию.

6. Второй, третий и четвёртый этапы повторяются до тех пор, пока одна из итераций не закончится одним из двух исходов:

   a) $\theta_e = a_{0e} \geq 0$, $e = \overline{1, n}$ (для задачи на max);

   b) Найдётся такое $\theta_i = a_{0i} < 0$, что все элементы этого столбца $a_{ri} < 0$, $r \in I_б$ – это признак неограниченности целевой функции W на множестве допустимых решений.

Метод искусственных переменных

При приведении системы ограничений к предпочтительному виду возможны следующие случаи:

1. Пусть ограничение задачи ЛП имеет вид:

$$\sum_{i=1}^{n} a_{ji} x_i \leq b_j, \qquad j=\overline{1, m}$$

Добавив к левым частям дополнительную переменную x_{n+j}; $j = \overline{1, m}$, получим расширенную задачу, эквивалентную исходной и в которой система ограничений имеет предпочтительный вид:

$$\sum_{i=1}^{n} a_{ji} x_i + x_{n+j} = b_j, \qquad j=\overline{1, m}$$

Это означает, что в начальный единичный базис можно включить вектора $\{A_{n+j}\}$ $j=\overline{1,m}$.

Следовательно, начальный опорный план:

$X = (0; 0, \ldots, 0, b_1, b_2, \ldots, b_m)$

← n → ← m →

В целевую функцию дополнительные переменные вводятся с нулевыми коэффициентами, то есть

$C_{n+j} = 0$; $j = \overline{1, m}$

2. Пусть ограничения задачи ЛП имеют вид:

$$\sum_{i=1}^{n} a_{ji} x_i \geq b_j, \qquad b_j \geq 0$$

Вычтем из левых частей дополнительные переменные $x_{n+j} \geq 0$, получим расширенную задачу, эквивалентную исходной. Но теперь в ней система ограничений не имеет предпочтительного вида, так как дополнительные переменные входят в левую часть с коэффициентом = -1

$$\sum_{i=1}^{n} a_{ji} x_i - x_{n+j} = b_j, \qquad j=\overline{1, m}$$

По тому плану начальный план недопустим

$X_H = (0; 0,\ldots., 0, -b_1, -b_2, \ldots,- b_m)$

← n → ← m →

В этом случае вводят так называемый искусственный базис. К левым частям системы ограничений типа равенств, не имеющих предпочтительного вида, добавляют искусственные переменные.

В целевую функцию искусственные переменные вводят с коэффициентом +M при решении задачи на min и с коэффициентом –M при решении задачи на max, где M-большое положительное число

$M >> C_i \quad i=\overline{1, n}$

Полученная задача называется M-задачей и соответствует исходной. Она всегда имеет предпочтительный вид.

Теорема. Если в оптимальном плане M-задачи

$X_{opt\ M} = (x_{01}; x_{02}, \ldots., x_{0n}, x_{0n+1}, \ldots, x_{0n+m})$

← n → ← m →

все искусственные переменные $x_{n+j} = 0$, то план $X_{opt} = (x_{01}, x_{02}, \ldots, x_{0n})$

оптимален для исходной задачи.

Если же хотя бы одна из искусственных переменных отлична от нуля, то система ограничений исходной задачи несовместна.

Искусственные переменные при использовании симплекс-метода образуют начальное базисное решение. Применив симплекс-метод, необходимо вывести из базиса все искусственные переменные. Если доказано, что от них нельзя избавиться, то задача, не имеет решения, ее ограничения противоречивы.

Псевдокод МЕТОД ИСКУССТВЕННЫХ ПЕРЕМЕННЫХ

```
ФУНКЦИЯ Симплекс _M _ Метод (Коэффициенты _C, Матрица _A, Вектор _B, Знаки _Ограничений):
  M = ВЕЛИКОЕ_ЧИСЛО // Теоретический бесконечно большой штраф
  // Шаг 1: Приведение к каноническому виду и ввод искусственных переменных
  ДЛЯ каждого ограничения i ОТ 1 ДО количества _строк:
    ЕСЛИ Знаки _ Ограничений[i] == "<=":
      Добавить _ Дополнительную _Переменную (s_i, коэффициент = +1)
      Сделать _Базисной (s_i, строка = i)
    ЕСЛИ Знаки _ Ограничений[i] == ">=":
      Добавить _ Дополнительную _Переменную (s_i, коэффициент = -1)
      Добавить _ Искусственную _ Переменную (M_i, коэффициент = +1)
      Сделать _ Базисной (M_i, строка = i)
```

```
        Коэффициенты _ C. Добавить _ Штраф (M_i, значение = -M) // Для задачи MAX
    ЕСЛИ Знаки _ Ограничений[i] == "=":
        Добавить _ Искусственную _ Переменную (M_i, коэффициент = +1)
        Сделать _ Базисной (M_i, строка = i)
        Коэффициенты _ C. Добавить _ Штраф (M_i, значение = -M)
  // Шаг 2: Модификация строки целевой функции (Обнуление M над базисом)
  Строка_ W= Сформировать _ Строку _ ЦФ (Коэффициенты _C)
  ДЛЯ каждой Искусственной _ Переменной M_i в Базисе:
    Строка_W = Строка_W - (M * Строка _ Где _ Находится (M_i))
  // Шаг 3: Основной цикл Симплекс-метода
  ПОКА ИСТИНА:
    Индекс _ Направляющего _ Столбца = Найти _ Индекс _ Мин _ Элемента (Строка_W) // Для MAX ищем самый отрицательный
    ЕСЛИ Элемент (Строка_W, Индекс _ Направляющего _ Столбца) >= 0:
      ПРЕРВАТЬ_ЦИКЛ // Оптимальное решение найдено или проверено
    Индекс _ Направляющей _ Строки = Найти _ Мин _ Отношение (Вектор _ B, Направляющий _ Столбец)
    ЕСЛИ Индекс _ Направляющей _ Строки == НЕ_НАЙДЕН:
      ВЫВЕСТИ "Целевая функция не ограничена сверху"
      ВЫХОД
    Выполнить _ Шаг _ Исключения _ Гаусса _ Жордана (Направляющая _Строка, Направляющий _Столбец) Обновить _Базис()
  // Шаг 4: Финальный анализ на допустимость
  ДЛЯ каждой переменной в Базисе:
    ЕСЛИ Переменная Является _ Искусственной () И Ее _ Значение _ B_ B > 0:
      ВЫВЕСТИ "Задача не имеет допустимых решений (Ограничения противоречивы)"
      ВЫХОД
  ВЫВЕСТИ "Оптимальное решение найдено", Базисные _ Значения, Значение_W
КОНЕЦ ФУНКЦИИ
```

Графический метод решения задачи ЛП

Графический метод основан на геометрической интерпретации задачи ЛП и применяется для решения задач двумерного пространства. Задачи трехмерного пространства

решаются очень редко, так как построение их решения неудобно и лишено наглядности. Рассмотрим метод на примере двумерной задачи.

Найти решение $X = (x_1, x_2)$, удовлетворяющее системе неравенств

$$\begin{cases} 3x_1 + x_2 \le 21 & \text{(I)} \\ 2x_1 + 3x_2 \le 30 & \text{(II)} \\ 2x_2 \le 16 & \text{(III)} \end{cases} \quad (8)$$

$$x_1, x_2 \ge 0, \quad (9)$$

при котором значение целевой функции

$$W(X) = 3x_1 + 2x_2 \to \max \quad (10)$$

достигает максимума.

Построим на плоскости в декартовой прямоугольной системе координат X_1OX_2 область допустимых решений задачи.

Первым неравенством (8) определяются две области на плоскости (рисунок 1). Одна из них — это область возможных планов задачи, другая — область, где этих планов нет. Границей между ними будет прямая, которую построим, заменив неравенство равенством $3x_1 + 2x_2 = 21$. По знаку первого неравенства находим область решения задачи. Аналогично, заменив неравенства равенствами, строим прямые II, III и по знакам неравенств определяем область решений задачи. Неравенства $X_1 \ge 0$, $X_2 \ge 0$ означают, что область решения будет расположена справа от оси ординат и над осью абсцисс. Таким образом, заштрихованная на рисунке 1 область OABCD будет областью допустимых решений, определенной ограничениями задачи. Крайние точки полученной выпуклой многогранной области будут соответствовать допустимым базисным решениям задачи (8) — (10). Значение целевой функции $W(X) = 3x_1+2x_2$ можно определить в любой точке $X = (x_1, x_2)$ области допустимых решений. Прямая линия, перпендикулярная вектору N = (3,2), будет геометрическим местом точек $X = (x_1, x_2)$, в которых целевая функция принимает одинаковые фиксированные значения. Так, в точке X' = (3,2) и в любой точке прямой, перпендикулярной вектору N и проходящей через точку X′ значением функции будет W = 3 • 3 + 2 - 2 = 13. Вектор N показывает направление параллельного перемещения прямой X'X", соответствующее увеличению целевой функции.

Максимального значения целевая функция достигает в крайней точке C многогранника, являющегося областью допустимых решений задачи. Координаты точки C будут оптимальным решением задачи $X_{орт} = (x_{01}, x_{02})$ и могут быть найдены при решении уравнений методом Крамера:

$$3x_1 + x_2 = 21$$
$$2x_1 + 3x_2 = 30.$$

Вычислим

$$\Delta = \begin{vmatrix} 3 & 1 \\ 2 & 3 \end{vmatrix} = 7;\ \Delta_1 = \begin{vmatrix} 21 & 1 \\ 30 & 3 \end{vmatrix} = 33;\ \Delta_2 = \begin{vmatrix} 3 & 21 \\ 2 & 30 \end{vmatrix} = 48;$$

$$x_{01} = \frac{\Delta 1}{\Delta} = \frac{33}{7} = 4\frac{5}{7}, \quad x_{02} = \frac{\Delta 2}{\Delta} = \frac{48}{7} = 6\frac{6}{7}.$$

$W_{max} = 3 * 4\frac{5}{7} + 2 * 6\frac{6}{7} = 27\frac{6}{7}$.

Следовательно, $X_{опт} = (4\frac{5}{7}, 6\frac{6}{7})$, $W_{max} = 27\frac{6}{7}$.

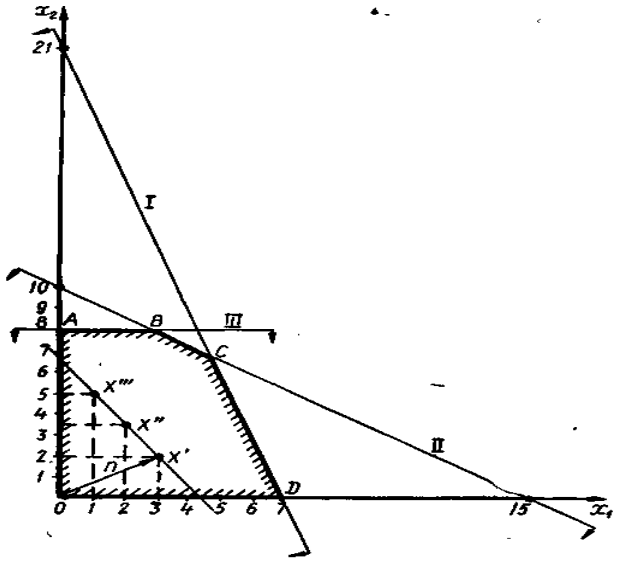


Рисунок 1 - Графическое решение исходной задачи

Область допустимых решений системы может быть:

- пустой, если система ограничений несовместна;
- одной точкой;
- выпуклым многогранником;
- неограниченной выпуклой многогранной областью.

Для вариантов, представленных на рисунках ограниченной областью допустимых решений, могут встретиться два случая:

- максимум целевой функции достигается в единственной точке;
- максимальное значение целевой функции иметь место в любой точке отрезка границы области допустимых значений.

В случае, когда область допустимых решений является неограниченной, могут встретиться варианты:

- целевая функция иметь экстремум;
- функция не ограничена сверху и снизу, т. е. Wmax = ∞, Wmin = — ∞

Контрольные вопросы

1. Что называется решением или планом задачи ЛП?

2. Допустимое множество решений.
3. Оптимальное решение задачи ЛП.
4. Каноническая форма записи ограничений задачи ЛП.
5. Предпочтительный вид представления ограничений задачи ЛП.
6. Какие есть способы нахождения значения целевой функции?
7. Критерий оптимальности в задачах ЛП на max (min).
8. Какой метод решения систем линейных уравнений лежит в основе метода симплекс-таблиц?
9. Как определяется вектор для включения в базис, если первоначальный план не является оптимальным?
10. Как определить вектор, подлежащий исключению из базиса? Какой элемент называется направляющим?
11. Какая переменная называется искусственной, когда она вводится и какой коэффициент соответствует целевой функции?
12. Зачем в системе ограничений необходим единичный базис?
13. Когда оптимальный план расширенной задачи является оптимальным планом исходной задачи?
14. Когда исходная задача не совместна и как это определить с помощью решения расширенной задачи?
15. Как определяется вектор, подлежащий включению в базис при использовании искусственного базиса?
16. Что понимается под предпочтительным видом записи уравнений ограничений?
17. Алгоритм графического метода решения задач ЛП.
18. Примеры областей допустимых решений задачи ЛП.

Пример решения задачи ЛП табличным симплекс-методом в среде Excel

Условие задачи

$$W=3X_1+3X_2 \rightarrow MAX$$

$$\begin{cases} X_1 - 4X_2 \le 4 \\ 3X_1+2X_2 \le 6 \end{cases}$$

$$X_1, X_2 \ge 0$$

Предпочтительная форма записи задачи ЛП:

$$W=3X_1+3X_2 + 0X_3+ 0X_4 \rightarrow MAX$$

$$\begin{cases} X_1 - 4X_2+1X_3+ 0X_4 = 4 \\ 3X_1+ 2X_2+ 0X_3+1X_4 = 6 \end{cases}$$

$$X_1, X_2, X_3, X_4 \ge 0$$

Решение задачи табличным симплекс-методом:

Таблица 1

	A	C	D	E	F	G	H	I	J	K
1		c			3	3	0	0		
2			X	B	A1	A2	A3	A4		
3		0	X3	4	1	-4	1	0		
4		0	X4	6	3	2	0	1		
5		W		0	-3	-3	0	0		
6										
7		MAX							0	
8		MIN							-3	A1
9										2
10										3
11		c			3	3	0	0		ИСТИНА
12			X	B	A1	A2	A3	A4		
13		0	X3	4	0	-4	1	0		
14		3	X1	2	1	2/3	0	1/3		
15		W		6	0	-1	0	1		
16										
17		MAX							6	
18		MIN							-1	A2
										0
										2/3
21		c			3	3	0	0		ИСТИНА
22			X	B	A1	A2	A3	A4		
23		0	X3	4	0	0	1	0		
24		3	X2	3	1 1/2	1	0	1/2		
25		W		9	1,5	0	0	1,5		
26		MAX							9	
27		MIN							0	A3
28	*Оптимал.план*	X1	0							
29		X2	3							
30		X3	4							
31		X4	0							

X3 в таблице1 определяется по следующей формуле:

ЕСЛИ(И((F3=1);(F4=0));"X1";ЕСЛИ(И((G3=1);(G4=0));"X2";ЕСЛИ(И((H3=1);(H4=0));"X3";ЕСЛИ(И((I3=1);(I4=0));"X4"))))

Определение коэффициента целевой функции в столбце **C**:

ЕСЛИ(И((F13=1);(F14=0));F11;ЕСЛИ(И((G13=1);(G14=0));G11;ЕСЛИ(И((H13=1);(H14=0));H11;ЕСЛИ(И((I13=1);(I14=0));I11))))

Нахождение значения целевой функции на каждой итерации:

СУММ(C13*E13;C14*E14)-E11

Нахождение значения оценки для вектора A1 в индексной строке:

СУММ(C13*F13;C14*F14)-F11

Варианты задач к лабораторной работе

№ 1

$W=2X_1 + X_2 \rightarrow MIN$

$$\begin{cases} 3X_1 - 2X_2 \le 12 \\ -X_1 + 2X_2 \le 8 \\ 2X_1 + 3X_2 \ge 5 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 2

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} 4X_1 - 2X_2 \le 12 \\ -X_1 + 3X_2 \le 6 \\ 2X_1 + 4X_2 \ge 4 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 3

$W=-2X_1 + 3X_2 \rightarrow MIN$

$$\begin{cases} 2X_1 + X_2 \le 24 \\ X_1 + 2X_2 \le 22 \\ X_1 - X_2 \ge 10 \\ X_1, X_2, \ge 0 \end{cases}$$

№ 4

$W=7X_1 - 2X_2 \rightarrow MIN$

$$\begin{cases} 5X_1 - 2X_2 \le 3 \\ X_1 + X_2 \ge 1 \\ 2X_1 + X_2 \le 4 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 5

$W=2X_1 + 7X_2 \rightarrow MAX$

$$\begin{cases} 12X_1 + 13X_2 \ge 17 \\ 3X_1 + X_2 \le 5 \\ X_1 + 4X_2 \le 6 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 6

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} X_1 + X_2 \le 6 \\ 3X_1 + 10X_2 \le 30 \\ X_1 + 11X_2 \ge 22 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 7

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} 4X_1 - 2X_2 \le 10 \\ -X_1 + 3X_2 \le 6 \\ 2X_1 + 4X_2 \ge 15 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 8

$W=3X_1 + 4X_2 \rightarrow MAX$

$$\begin{cases} 6X_1 + 2X_2 \le 175 \\ X_1 + X_2 \le 30 \\ X_1 + 4X_2 \ge 84 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 9

$W=2X_1 + 3X_2 \rightarrow MAX$

$$\begin{cases} 2X_1 + X_2 \le 10 \\ -2X_1 + 3X_2 \le 6 \\ 2X_1 + 4X_2 \ge 8 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 10

$W=X_1 + X_2 \rightarrow MIN$

$$\begin{cases} 2X_1 + 4X_2 \le 16 \\ -4X_1 + 2X_2 \le 8 \\ X_1 + 3X_2 \ge 9 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 11

$W=4X_1 + X_2 \rightarrow MAX$

$$\begin{cases} 2X_1 - X_2 \le 12 \\ X_1 + 3X_2 \le 18 \\ 2X_1 + 5X_2 \ge 10 \\ X_1, X_2, \ge 0 \end{cases}$$

№ 12

$W=2X_1 + 3X_2 \rightarrow MAX$

$$\begin{cases} 2X_1 - X_2 \le 16 \\ 3X_1 + 2X_2 \le 18 \\ -X_1 + 3X_2 \ge 24 \\ X_1, X_2 \ge 0 \end{cases}$$

№ 13

$W=-2X_1 + X_2 \rightarrow MIN$

$$\begin{cases} 2X_1 + 2X_2 \geq 10 \\ 3X_1 + X_2 \geq 15 \\ X_1 \leq 8 \end{cases}$$
$$X_1, X_2 \geq 0$$

№ 14

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} X_1 + X_2 \leq 4 \\ 3X_1 + X_2 \geq 4 \\ X_1 + 5X_2 \geq 4 \\ 0 \leq X_1 \leq 3 \end{cases}$$
$$X_2 \geq 0$$

№ 15

$W=2X_1 - X_2 \rightarrow MAX$

$$\begin{cases} X_1 - X_2 \geq -3 \\ 6X_1 + 7X_2 \leq 42 \\ 3X_1 - 2X_2 \leq 6 \end{cases}$$
$$X_1, X_2, \geq 0$$

№ 16

$W=7X_1 - X_2 \rightarrow MIN$

$$\begin{cases} X_1 + X_2 \geq 3 \\ 5X_1 + X_2 \leq 5 \\ X_1 + 5X_2 \geq 4 \end{cases}$$
$$X_1, X_2 \geq 0$$

№ 17

$W=7X_1 + X_2 \rightarrow MAX$

$$\begin{cases} X_1 + X_2 \leq 14 \\ 3X_1 - 5X_2 \leq 15 \\ 5X_1 + 3X_2 \geq 21 \end{cases}$$
$$X_1, X_2 \geq 0$$

№ 18

$W=8X_1 + 7X_2 \rightarrow MAX$

$$\begin{cases} X_1 - 2X_2 \leq 12 \\ 4X_1 + X_2 \leq 16 \\ 5X_1 + 5X_2 \geq 25 \end{cases}$$
$$X_1, X_2 \geq 0$$

№ 19

$W=2X_1 + 3X_2 \rightarrow MAX$

$$\begin{cases} 2X_1 - X_2 \leq 16 \\ 3X_1 + 2X_2 \leq 18 \\ -X_1 + 3X_2 \geq 10 \end{cases}$$
$$X_1, X_2 \geq 0$$

№ 20

$W=2X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} 3X_1 - 2X_2 \geq -6 \\ X_1 + X_2 \geq 6 \\ 0 \leq X_1 \leq 9 \end{cases}$$
$$0 \leq X_2 \leq 6$$

№ 21

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} 5X_1 - 2X_2 \leq 4 \\ X_1 - 2X_2 \geq -4 \\ X_1 + X_2 \geq 4 \end{cases}$$
$$X_1, X_2 \geq 0$$

№ 22

$W=2X_1 - 4X_2 \rightarrow MIN$

$$\begin{cases} 8X_1 - 5X_2 \leq 16 \\ X_1 + 3X_2 \geq 2 \\ 2X_1 + 7X_2 \leq 9 \end{cases}$$
$$X_1, X_2 \geq 0$$

№ 23

$W=X_1 + X_2 \rightarrow MIN$

$$\begin{cases} 3X_1 + X_2 \geq 8 \\ X_1 + 2X_2 \geq 6 \\ X_1 - X_2 \leq 3 \end{cases}$$
$$X_1, X_2 \geq 0$$

№ 24

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} -3X_1 + 2X_2 \leq 9 \\ 3X_1 + 4X_2 \leq 27 \\ 2X_1 + X_2 \geq 14 \end{cases}$$
$$X_1, X_2 \geq 0$$

№ 25

$W=7X_1 - 2X_2 \rightarrow MIN$

$$\begin{cases} X_1 + X_2 \geq 1 \\ -3X_1 + X_2 \leq 3 \\ 2X_1 + X_2 \leq 4 \\ X_1, X_2 \geq 0 \end{cases}$$

№26

$W=X_1 + X_2 \rightarrow MAX$

$$\begin{cases} X_1 + 2X_2 \leq 14 \\ -5X_1 + 3X_2 \leq 15 \\ 4X_1 + 6X_2 \geq 24 \\ X_1, X_2 \geq 0 \end{cases}$$

№ 27

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} 4X_1 - 2X_2 \leq 10 \\ -X_1 + 3X_2 \leq 6 \\ 2X_1 + 4X_2 \geq 16 \\ X_1, X_2 \geq 0 \end{cases}$$

№ 28

$W=X_1 + X_2 \rightarrow MIN$

$$\begin{cases} 2X_1 + 4X_2 \leq 16 \\ -4X_1 + 2X_2 \leq 8 \\ X_1 + 3X_2 \geq 6 \\ X_1, X_- \geq 0 \end{cases}$$

№29

$W=-2X_1 + X_2 \rightarrow MIN$

$$\begin{cases} 3X_1 - 2X_2 \leq 12 \\ -X_1 + 2X_2 \leq 8 \\ 2X_1 + 3X_2 \geq 6 \\ X_1, X_2 \geq 0 \end{cases}$$

№ 30

$W=7X_1 - 2X_2 \rightarrow MIN$

$$\begin{cases} 5X_1 - 2X_2 \leq 10 \\ X_1 + X_2 \geq 1 \\ -3X_1 + X_2 \leq 3 \\ X_1, X_2 \geq 0 \end{cases}$$

№ 31

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} 4X_1 - 2X_2 \leq 8 \\ -X_1 + 3X_2 \leq 6 \\ 2X_1 + 4X_2 \geq 16 \\ X_1, X_2 \geq 0 \end{cases}$$

№ 32

$\mathbf{W}=3X_1 + 4X_2 \rightarrow MAX$

$$\begin{cases} 6X_1 + 2X_2 \leq 18 \\ X_1 + X_2 \leq 3 \\ X_1 + 4X_2 \geq 8 \\ X_1, X_2 \geq 0 \end{cases}$$

№ 33

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} X_1 + X_2 \leq 6 \\ 3X_1 + 10X_2 \leq 30 \\ X_1 + 4X_2 \geq 8 \\ X_1, X_2 \geq 0 \end{cases}$$

№ 34

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} 4X_1 - 2X_2 \leq 10 \\ -X_1 + 3X_2 \leq 6 \\ 2X_1 + 4X_2 \geq 12 \\ X_1, X_2 \geq 0 \end{cases}$$

№ 35

$\mathbf{W}=3X_1 + 4X_2 \rightarrow MAX$

$$\begin{cases} 6X_1 + 2X_2 \leq 18 \\ X_1 + X_2 \leq 3 \\ X_1 + 4X_2 \geq 8 \\ X_1, X_2 \geq 0 \end{cases}$$

№ 36

$W=X_1 + 2X_2 \rightarrow MAX$

$$\begin{cases} 4X_1 - 2X_2 \leq 10 \\ -X_1 + 3X_2 \leq 6 \\ 2X_1 + 4X_2 \geq 8 \\ X_1, X_2 \geq 0 \end{cases}$$

ЛАБОРАТОРНАЯ РАБОТА № 6

Тема: Транспортная задача. Метод потенциалов

Цель работы:

- изучить и сравнить методы нахождения начального опорного плана (метод северо-западного угла и метод минимального элемента),
- реализовать алгоритмы обоих методов.
- изучить метод потенциалов для нахождения оптимального опорного плана транспортной задачи;
- применить алгоритм метода потенциалов для оптимизации найденных начальных опорных планов, сравнить результаты.

Постановка транспортной задачи по критерию стоимости

Пусть имеются пункты-поставщики:

$\boldsymbol{A_j}$; $j = \overline{1, m}$

располагающие некоторым однородным продуктом объемом по a_j единиц, и пункты потребители

$\boldsymbol{B_i}$; $I = \overline{I, n}$

с объемом потребления по b_i единиц.

Задана матрица $\mathbf{C} = \{\boldsymbol{C_{ji}}\}$ - матрица тарифов или транспортных издержек, где $\boldsymbol{C_{ji}}$- стоимость перевозки единицы продукции от j поставщика i-му потребителю.

Возникает задача определения плана перевозок

$\mathbf{X} = \{\boldsymbol{x_{ji}}\}$,

(где x_{ji}-количество единиц продукции, поставляемой из пункта A_j в B_i) такого, что запросы всех потребителей будут полностью удовлетворены, то есть весь продукт от поставщиков вывезен и суммарные транспортные издержки будут min.

Условия этой задачи удобно представить в виде таблицы 2.

Таблица 2

		Объем потребления					
	b_i / a_j	b_1	b_2	……	b_i	….....	b_n
Объем производства	a_1	c_{11} x_{11}	c_{12} x_{12}	……	c_{1i} x_{1i}	….....	c_{1n} x_{1n}
	a_2	c_{21} x_{21}	….....	……	c_{2i} x_{2i}	….....	c_{2n} x_{2n}
	….	…….	…….	……	……	……	……
	a_j	c_{j1} x_{j1}	…….	……	c_{ji} x_{ji}	……	c_{jn} x_{jn}
	….	…….	…….	……		……	
	a_m	c_{m1} x_{m1}	…….	……	c_{mi} x_{mi}	……	c_{mn} x_{mn}

Построим математическую модель задачи.

Функция цели – минимизация суммарных затрат на перевозку продукции:

$$W = \sum_{j=1}^{m}\sum_{i=1}^{n} c_{ji}x_{ji} \rightarrow min \qquad (1)$$

Ограничения:

1) на запасы продукции у поставщиков, которые по условию задачи должны быть полностью вывезены:

$$\sum_{i=1}^{n} x_{ji} = a_j,\ j=\overline{1,\ m} \qquad (2)$$

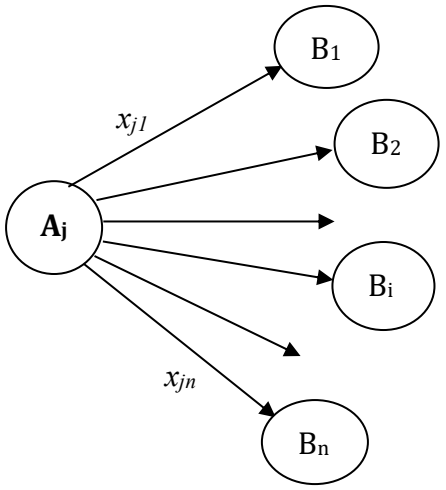


Рисунок 2

2) на запросы потребителей, которые по условию задачи должны быть полностью удовлетворены:

$$\sum_{j=1}^{m} x_{ji} = b_i,\ i = \overline{1,\ n} \qquad (3)$$

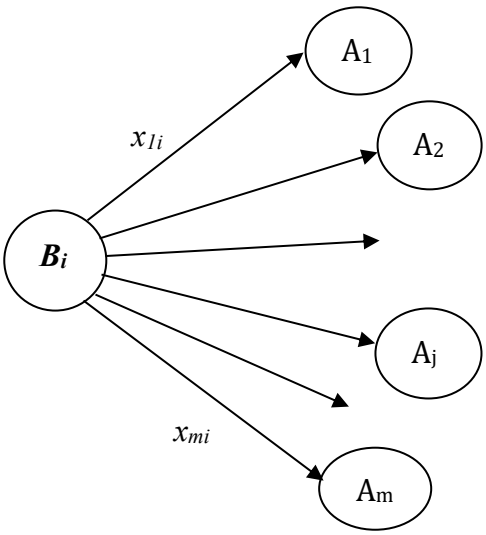


Рисунок 3

3) условие неотрицательности, исключающее обратные перевозки

$$x_{ji} \geq 0 \qquad \forall j,\ i \tag{4}$$

На практике существуют задачи, в которых выполняются равенство между суммарными ресурсами (запасами) и суммарными потребностями (условие баланса)

$$\sum_{j=1}^{m} a_j = \sum_{i=1}^{n} b_i \tag{5}$$

(объем производства равен объему потребления), так задачи, в которых объем производства не равен объему потребления:

$$\sum_{j=1}^{m} a_j \neq \sum_{i=1}^{n} b_i \tag{6}$$

Задача (1) ÷ (4) при условии (5) называется **закрытой моделью**, а при условии (6) – **открытой**.

Теорема: Любая транспортная задача, у которой суммарный объем запаса совпадает с суммарным объемом потребления, имеет решение.

Таким образом, условие (5) является необходимым и достаточным условием совместимости (2) и (3) и, следовательно, разрешимости задачи.

Открытая модель должна быть сведена к закрытой транспортной модели.

При

$$\sum_{j=1}^{m} a_j > \sum_{i=1}^{n} b_i$$

(запасы поставщиков превышают потребности потребителей) вводят дополнительный (фиктивный) пункт потребления, запросы которого равны излишку запаса

$$b_{n+1} = \sum_{j=1}^{m} a_j - \sum_{i=1}^{n} b_i \tag{7}$$

При

$$\sum_{j=1}^{m} a_j > \sum_{i=1}^{n} b_i$$

(запросы больше запасов) вводится дополнительный (фиктивный) пункт производства, запасы которого считают равными недостающей продукции:

$$a_{m+1} = \sum_{i=1}^{n} b_i - \sum_{j=1}^{m} a_i \tag{8}$$

В обоих случаях при выполнении условия (7) или (8) получим расширенную закрытую модель.

Матрицу $X = \{x_{ji}\}$, удовлетворяющую условию (2)÷(5), называют **допустимым** планом перевозок, а x_{ji} – **допустимыми перевозками**.

Допустимый план X, доставляющий W (формула (1)) минимальное значение, называется **оптимальным** X_0.

Матрица $C = \{c_{ji}\}$ называется **матрицей тарифов** или матрицей транспортных издержек.

Оптимальный план расширенной закрытой модели дает оптимальный план исходной задачи. В случае введения фиктивного потребителя (7) поставщики $x_{j,n+1}$ в оптимальном плане расширенной задачи покажут остатки продукции на складах поставщиков. Тарифы $C_{j,n+1}$ считаются равными нулю.

В случае введения фиктивного поставщика (8) поставки $x_{m+1,i}$ в оптимальном плане расширенной задачи покажут объемы недостачи продукции.

Метод северо-западного угла

Рассмотрим идею метода на примере.

Дана транспортная задача с четырьмя пунктами производства с объемами производства $\{a_j\}=\{1, 2, 3, 4\}$ и четырьмя пунктами потребления с объемами потребления $\{b_i\}=\{5, 1, 2, 2\}$. Построим матрицу транспортной задачи.

					a_j					
	1	0	0	0	1	0	0	0	0	0	0
	2	0	0	0	2	2	0	0	0	0	0
X =	2	1	0	0	3	3	3	1	0	0	0
	0	0	2	2	4	4	4	4	4	2	0
bi	5	1	2	2							
	4	1	2	2							
	2	1	2	2							
	0	1	2	2							
	0	0	2	2							
	0	0	0	2							
	0	0	0	0							

Для удобства вычисления в столбцах справа будем записывать остатки не вывезенного продукта, а в строках неудовлетворенные потребности. Заполнение таблицы начинают с левого верхнего угла.

Сравнив $a_j = 1$ и $b_1 = 5$, выбираем меньшее из них и получаем $x_{11} = 1$. Так как выбор произведен по строке, то остальная часть левой строки должна быть заполнена 0 (из данного пункта больше ничего вывезти нельзя).

Во вспомогательном столбце записываются остатки не вывезенного продукта. $a_{ji} = 0$. Во вспомогательной строке не удовлетворенные потребности после одного шага заполнения. Переходим ко второй строке и начинаем заполнение со строки x_{21}: $a_2^{(1)} = 2$; $b_1^{(1)} = 4$, выбираем меньшее из них. Поэтому $x_{21} = 2$. Остальную часть второй строки заполняем нулями. И так далее.

Алгоритм метода северо-западного угла

1. Определим верхний левый элемент матрицы X.

   $x_{11} = min(a_1, b_1)$

   Возможны три случая:

   a) если $a_1 < b_1$, то $x_{11} = a_1$ и вся первая строка, начиная со второго элемента, заполняется нулями. Больше ничего из пункта A_1 вывезти нельзя.

   b) если $a_1 > b_1$, то $x_{11} = b_1$, а все оставшиеся элементы первого столбца заполняются нулями. В пункт B_1, больше ничего привозить не нужно.

   c) если $a_1 = b_1$, то $x_{11} = a_1 = b_1$, а все оставшиеся элементы первого столбца и первой строки заполняются нулями.

   На этом первый шаг метода заканчивается.

2. Пусть проделано k шагов.

   k+1 шаг состоит в следующем: определяют верхний левый элемент незаполненной верхней части матрицы X.

   Пусть этот элемент $x_{\lambda\mu}$ $(\lambda + \mu = k + 2)$, $k = 0, 1, 2\ldots$, причем $x_{\lambda\mu} = \min\{a_\lambda^{(k)}, b_\mu^{(k)}\}$, где

$$a_\lambda^{(k)} = a_\lambda - \sum_{i=1}^{m} x_{\lambda i}$$

$$b_\mu^{(k)} = b_\mu - \sum_{j=1}^{\lambda-1} x_{j\mu}$$

Если $a_\lambda^{(k)} \le b_\mu^{(k)}$, то заполняем нулями строки λ, начиная с $\mu+1$ элемента. В противном случае заполняют нулями оставшуюся часть μ-го столбца. Метод северо-западного угла хорошо реализуется на ЭВМ.

Число итераций для получения оптимального решения транспортной задачи существенно зависит от начального плана, так как при использовании метода северо-западного угла не учитываются стоимости перевозок, то есть сам начальный план в общем случае будет далек от оптимального. Поэтому при решении задачи вручную, целесообразнее применять другие методы, основанные на учете стоимости.

Метод минимального элемента

Он позволяет построит начальный опорный план транспортной задачи и является вариантом метода северо-западного угла, учитывающий специфику матрицы тарифов $C=\{c_{ji}\}$.

В отличие от метода северо-западного угла данный метод позволяет сразу получить достаточно экономичный план, сокращая общее количество итераций по его оптимизации.

Алгоритм метода минимального элемента

Элементы матрицы C нумеруют, начиная с минимального в порядке возрастания, и затем в этом же порядке заполняется матрица X, используя метод северо-западного угла.

Пример: нумеруем матрицу тарифов (затемненные прямоугольники).

A_j \ B_i	1	2	3	4	B_i / A_j
1	7 **10**	8 **11**	5 **7**	3 **5**	11
2	2 **3**	4 **6**	5 **8**	9 **12**	11
3	6 **9**	3 **4**	1 **1**	2 **2**	8
A_j / B_i	5	9	9	7	B_i \ A_j

Проверяем условие баланса

$$\sum_{j=1}^{3} a_j = 11+11+8 = 30,$$

$$\sum_{i=1}^{4} b_i = 5+9+9+7 = 30.$$

$$\sum_{j=1}^{3} a_j = \sum_{i=1}^{4} b_i$$

поэтому задача разрешима.

Заполняется матрица $X_н$ (начальный опорный план).

0 **10**	3 **11**	1 **7**	7 **5**	11	4	3	0
5 **3**	6 **6**	0 **8**	0 **12**	11	6	0	0
0 **9**	0 **4**	8 **1**	0 **2**	8	0	0	0
5	9	9	7				
0	3	1	7				
0	0	0	0				

В результате получаем следующий начальный опорный план:

$$X_H = \begin{vmatrix} 0 & 3 & 1 & 7 \\ 5 & 6 & 3 & 0 \\ 0 & 0 & 8 & 0 \end{vmatrix}$$

Соответствующее значение целевой функции:

$$W(X_H) = \sum_{j=1}^{3} \sum_{i=1}^{4} c_{ji}x_{ji} = 3*8+1*5+7*3+5*2+6*4+8*1 = 92.$$

Метод потенциалов

Идея метода состоит в следующем. Сначала строится начальный опорный план одним из методов, изложенных выше, а затем осуществляется его улучшение, пока не будет удовлетворяться признак оптимального плана.

Алгоритм метода потенциалов решения транспортной задачи состоит из предварительного этапа и конечного числа итераций.

На предварительном этапе строят начальный опорный план X_o. Затем для этого плана рассчитывают оценочную матрицу

$$D_0 = ||C_{ij} - (V_j - U_i)||_{mxn}, \quad (1)$$

где V_j и U_i – потенциалы пунктов отправления A_i и пунктов назначения B_j.

Предварительные потенциалы выбирают таким образом, чтобы для связанных коммуникациями пар пунктов, для которых в плане X_o, $x_{ij} \neq 0$, разность потенциалов была равна c_{ij}:

$$V_j - U_i = C_{ij} \quad (2)$$

Если матрица D_o не содержит отрицательных элементов, то X_o – оптимальный план. В противном случае X_o - неоптимальный план и его можно улучшить.

Описание алгоритма метода потенциалов

Предварительный этап

1. Определяем начальный опорный план X_o одним из описанных способов.
2. Вычисляем оценочную матрицу D_o. Для расчета элементов матрицы необходимо первоначально определить все потенциалы V_j и U_i. Строим схему перевозок, соответствующую начальному опорному плану X_o, то есть соединяем коммуникациями пункты отправления и назначения, для которых $x_{ij} \neq 0$. Пользуясь соотношением (2), определяем последовательно все потенциалы пунктов отправления и назначения, принимая для удобства $U_1 = 0$. Затем по (1) находим элементы матрицы D_o. Очевидно, что позиции матрицы D_o, отвечающие базисным элементам плана X_o, заняты нулями.

(k+1)-я итерация

1. Если в оценочной матрице D_k все элементы положительны, план $X_{(k)}$ – оптимальный, в противном случае следует приступить к его улучшению.
2. Выбираем наибольший по модулю отрицательный элемент оценочной матрицы $d_{st}^{(k)}$ и, начиная с соответствующего ему элемента x_{st}, в матрице $X_{(k)}$ строим замкнутую цепочку, в которую входят элементы $x_{ij}^{(k)} \neq 0$.

Затем определяем θ - минимальный элемент среди всех нечетных по порядку расположения в цепочке, считая первым следующий за x_{st} элемент.

3. Строим новый план $X_{(k+1)}$, прибавляя θ по всем четным элементам цепочки и вычитая из нечетных. Элементы матрицы $X_{(k)}$, не входящие в цепочку, переносятся в матрицу $X_{(k+1)}$ без изменения.

4. С помощью эквивалентных преобразований матрицы $D_{(k)}$ находим оценочную матрицу $D_{(k+1)}$ для нового плана $X_{(k+1)}$. Для этого подчеркиваем в матрице $D_{(k)}$ все элементы, соответствующие ненулевым элементам матрицы $X_{(k+1)}$ (они обязательно равны нулю). В матрице $D_{(k)}$ зачеркиваем строку, содержащую элемент d_{st}. Если в этой строке имеются подчеркнутые элементы, то зачеркиваем соответствующие этим элементам столбцы. Если в каждом зачеркнутом столбце имеются подчеркнутые элементы, зачеркиваем соответствующие им строки, и так до тех пор, пока описанная процедура выполнима. После этого по всем элементам зачеркнутых строк прибавляем $|d_{st}|$, а от элементов зачеркнутых столбцов вычитаем $|d_{st}|$. Получаем новую оценочную матрицу. Если в матрице $D_{(k+1)}$ нет отрицательных элементов, то план $X_{(k+1)}$ – оптимальный, иначе переходим к следующей итерации.

На этом итерация завершается.

Пример

Дано: Т – задача, где С – матрица тарифов, a_j – объёмы производства, b_i – объёмы потребления

					a_j
C =	7	8	5	3	11
	2	4	5	9	11
	6	3	1	2	8
b_i	5	9	9	7	

Определить оптимальный план Т- задачи.

Решение:

Так как условия баланса выполняется

$$\sum_{j=1}^{3} a_j = \sum_{i=1}^{4} b_i = 30,$$

то задача разрешима.

Предварительный этап

1. определяем начальный опорный план с помощью метода северо-западного угла:

a_j

Хн =

5	6	0	0	11	6	0			
0	3	8	0	11	11	8	0		
0	0	1	7	8	8	8	8	7	0

b_i

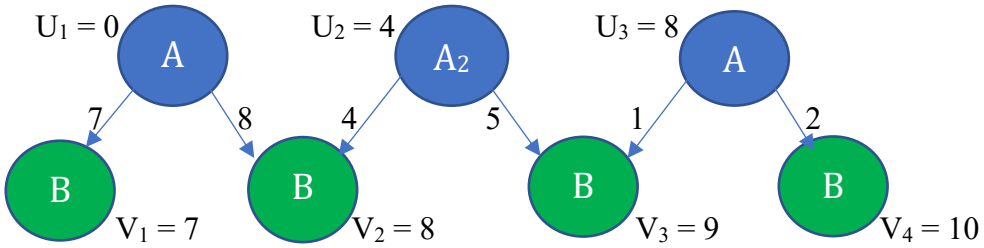
5	9	9	7
0	9	9	7
0	3	9	7
	0	9	7
		1	7
		0	7
			0

Таким образом, опорный план является невырожденным, так как число занятых клеток равно 6 и rang A = m + n – 1 = 6.

Хн =

5	6	0	0
0	3	8	0
0	0	1	7

2. Строим схему основных коммуникаций.



2. Находим предварительные потенциалы:

$Vi^{(н)} = Cji + Uj^{(н)}$; $Uj^{(н)} = Cji - Vi^{(н)}$.

$U_1^н = 0$; $V_1^н = U_1^н + 7 = 7$; $V_2^н = U_1^н + 8 = 8$; $U_2^н = V_2^н - 4 = 8 - 4 = 4$;

$V_3^н = U_2^н + 5 = 4 + 5 = 9$; $U_3^н = V_3^н - 1 = 9 - 1 = 8$; $V_4^н = U_3^н + 2 = 10$.

3. Определяем элементы оценочной матрицы по формуле (9.3)

$Dн = \{dji\} = \{Cji - (V_i^{(н)} - U_j^{(н)})\} = \{Cji - V_i^{(н)} + U_j^{(н)}\}$:

$d_{11} = C_{11} - V_1^{(н)} + U_1^{(н)} = 7 - 7 + 0 = 0$; $d_{14} = C_{14} - V_4^{(н)} + U_1^{(н)} = 3 - 10 + 0 = -7$;

$d_{12} = C_{12} - V_2^{(н)} + U_1^{(н)} = 8 - 8 + 0 = 0$; $d_{21} = C_{21} - V_1^{(н)} + U_2^{(н)} = 2 - 7 + 4 = -1$;

$d_{13} = C_{13} - V_3^{(н)} + U_1^{(н)} = 5 - 9 + 0 = -4$; и т. д.

Dн =

0	0	-4	-7
-1	0	0	3
7	3	0	0

Δн = - 7

4. Вычисляем целевую функцию:

$$W(X_н) = \sum_{j=1}^{3}\sum_{i=1}^{4} c_{ji}\, x_{ji} = 7*5+8*6+4*3+5*8+1*1+2*7 = 150.$$

5. Проверяем условие оптимальности: план не оптимален, так как содержатся элементы меньше 0.

0 – я итерация

I этап: Улучшаем план Хн

1. Выбираем наименьший отрицательный элемент оценочной матрицы $d_{14}^{(н)} = -7$ и, начиная с соответствующего ему элемента $x_{14}^{(н)}$.

2. Строим замкнутую цепочку

Хн =

5	6^{-}	0	0^{+}
0	3^{+}	8^{-}	0
0	0	1^{+}	7^{-}

3. Определяем $\theta_н$ - минимальный элемент среди всех помеченные знаком « – »:

$$\theta_н = \min\{x_{ji}^{(н)}\} = \min\{6^{-}; 8^{-}; 7^{-}\} = 6.$$

4. Строим новый план X_1, прибавляя $\theta_н$ по всем четным элементам цепочки, помеченные знаком « + », и вычитая из нечетных, помеченные знаком « – ».

X_1 =

5	0	0	6
0	9	2	0
0	0	7	1

C =

7	8	5	3
2	4	5	9
6	3	1	2

5. Вычисляем значение целевой функции W_1, соответствующей плану X_1:

$W_1 = W_н + \theta_н * \Delta_н = 150 – 6*7 = 108.$

6. Контроль вычислений:

$$W_1 = \sum_j \sum_i C_{ji}\, x_{ji}^{(1)} = 5*7 + 6*3 + 9*4 + 2*5 + 7*1 + 1*2 = 108.$$

II этап: Преобразуем матрицу $D_н$ в D_1

1. Подчеркиваем в матрице $D_н$ все элементы, соответствующие ненулевым элементам матрицы X_1 (они обязательно равны нулю в матрице D_1).
2. Находим минимальный отрицательный элемент матрицы D_1:

   $\Delta н = d_{14}^{(н)} = -7$.
3. В матрице D_1 зачеркиваем строку, содержащую элемент $\Delta_н = d_{14}$. Если в этой строке имеются подчеркнутые элементы, то зачеркиваем соответствующие этим элементам столбцы. Если в каждом зачеркнутом столбце имеются подчеркнутые элементы, зачеркиваем соответствующие им строки, и так до тех пор, пока описанная процедура выполнима.

$D_н =$

0	0	- 4	-7	+|-7|
-1	0	0	3	
7	3	0	0	
- |-7|				

4. Строим матрицу D_1. Для этого величину $|\Delta_н|$ прибавляем к элементам всех вычеркнутых строк, а от элементов вычеркнутых столбцов вычитаем $|\Delta_н|$. Получаем новую оценочную матрицу.

$D_1 =$

0	7	3	0
-8	0	0	3
0	3	0	0

$\Delta_1 = -8$

5. Проверяем выполнимость условия оптимальности. В матрице D_1 есть отрицательный элемент, то план X_1 – неоптимальный, переходим к улучшению плана, следующей итерации.

2 – я итерация. Улучшаем план X_1

$X_1 =$

5^-	0	0	6^+
0^+	9	2^{--}	0
0	0	7^+	1^-

$X_2 =$

4	0	0	7
1	9	1	0
0	0	8	0

$\theta_1 = \min \{x_{ji}^{(1)}\} = \min \{5^-; 2^-; 1^-\} = 1)$

$D_1 =$

0	7	3	0
-8	0	0	3
0	3	0	0

+|-8|

+|-8|

- |-8| - |-8|

$D_2 =$

0	- 1	- 5	0
0	0	0	11
8	3	0	8

$\Delta_2 = - 5$

$W_2 = W_1 + \theta_1 * \Delta_1 = 108 - 8*1 = 100.$

Контроль вычислений:

$W_2 = \sum_j \sum_i cji\, x_{ji}^{(2)} = 28 + 21 + 2 + 36 + 5 + 8 = 100.$

3 – я итерация. Улучшаем план X_2

$X_2 =$

4 -	0	0 +	7 -
1 +	9	1 -	0
0	0	8 -	0

$X_3 =$

3	0	1	7
2	9	0	0
0	0	8	0

$\theta_2 = \min \{xji^{(2)}\} = \min \{4^-; 1^-\} = 1.$

$D_2 =$

0	- 1	- 5	0
0	0	0	11
8	3	0	8

+|-5|

+|-5|

- |-5| - |-5| - |-5|

$D_3 =$

0	- 1	0	0
0	0	5	11
3	- 2	0	3

$\Delta_3 = - 2$

$W_3 = W_2 + \theta_2 * \Delta_2 = 100 - 1*5 = 95.$

Контроль вычислений:

$W_3 = \sum_j \sum_i c_{ji}\, x_{ji}^{(3)} = 21 + 5 + 21 + 4 + 36 + 8 = 95.$

4 – я итерация. Улучшаем план X_3

$X_3 =$

3 -	0	1 +	7
2 +	9 -	0	0
0	0 +	8 -	0

$X_4 =$

0	0	4	7
5	6	0	0
0	3	5	0

$\theta_3 = \min \{\overline{xji^{(3)}}\} = \min \{3^-; 9^-; 9^-\} = 3.$

$D_3 =$

0	- 1	0	0	+|-2|
0	0	5	11	
3	- 2	0	3	+|-2|
	- |-2|	- |-2|		

$D_4 =$

2	1	0	0
0	0	3	9
1	0	0	3

Проверяем выполнимость условия оптимальности. В матрице D_4 нет отрицательных элементов, поэтому план X_4 – оптимальный $W_{min} = W_4 = W_3 + \theta_3 * \Delta_3 = 95 - 3*2 = 89$.

Контроль вычислений:

$$W_{min} = W_4 = \sum_j \sum_i cji\, x_{ji}^{(4)} = 20 + 21 + 10 + 24 + 9 + 5 = 89.$$

Контрольные вопросы

1. Сформулируйте транспортную задачу ЛП и напишите ее математическую модель.
2. Сформулируйте теорему о существовании решения транспортной задачи.
3. Какие существуют методы построения начального опорного плана и постройте опорный план с помощью этих методов?
4. Сколько положительных перевозок должен содержать невырожденный опорный план и почему?
5. В чем заключается опорность плана транспортной задачи, условия которой записаны в виде таблицы?
6. Какая модель транспортной задачи называется закрытой, а какая – открытой?
7. Как открытую модель преобразовать в закрытую?
8. Для решения каких экономических задач применяется транспортная задача?
9. Дайте определение системе потенциалов, расскажите, как она строится.
10. В каком случае опорный план транспортной задачи является оптимальным?

Задание к лабораторной работе

Найти начальный опорный план транспортной задачи двумя методами.

Ниже приведены задачи по вариантам:

№1

$$C = \left|\begin{array}{cccc|c} 3 & 1 & 7 & 5 & 30 \\ 7 & 3 & 5 & 6 & 75 \\ 6 & 6 & 4 & 3 & 45 \end{array}\right| \quad \begin{matrix} a_j \\ \\ \\ \\ \end{matrix}$$

b_i 40 35 25 60

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№2

$$C = \left|\begin{array}{cccc|c} 5 & 2 & 4 & 1 & 50 \\ 6 & 3 & 1 & 5 & 30 \\ 3 & 2 & 4 & 2 & 40 \end{array}\right| \quad \begin{matrix} a_j \\ \\ \\ \\ \end{matrix}$$

b_i 20 30 10 30

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№3

$$C = \left|\begin{array}{ccc|c} 3 & 5 & 6 & 30 \\ 4 & 4 & 2 & 80 \\ 7 & 6 & 9 & 20 \\ 12 & 10 & 4 & 20 \end{array}\right| \quad \begin{matrix} a_j \\ \\ \\ \\ \end{matrix}$$

b_i 50 30 55

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,3) - объем потребления i-го предприятия.

№4

$$C = \left|\begin{array}{cccc|c} 4 & 5 & 6 & 2 & 30 \\ 1 & 3 & 2 & 3 & 90 \\ 4 & 1 & 5 & 1 & 40 \end{array}\right| \quad \begin{matrix} a_j \\ \\ \\ \\ \end{matrix}$$

b_i 70 30 50 30

где C - матрица тарифов,

a_j (j=1,5) - объем производства j-го предприятия,

b_j (j=1,3) - объем потребления i-го предприятия.

№5

$$C = \left|\begin{array}{cccc|c} 2 & 3 & 4 & 3 & 80 \\ 5 & 3 & 1 & 2 & 60 \\ 2 & 1 & 4 & 2 & 80 \end{array}\right| \quad \begin{matrix} a_j \\ \\ \\ \\ \end{matrix}$$

b_i 20 40 60 80

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№6

$$C = \left|\begin{array}{ccc|c} 1 & 4 & 2 & 10 \\ 6 & 2 & 7 & 15 \\ 5 & 4 & 7 & 20 \\ 5 & 4 & 8 & 25 \end{array}\right| \quad \begin{matrix} a_j \\ \\ \\ \\ \end{matrix}$$

b_i 25 15 10

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,3) - объем потребления i-го предприятия.

№7

$$C = \left|\begin{array}{cccc|c} 1 & 2 & 4 & 1 & 50 \\ 2 & 3 & 1 & 5 & 30 \\ 3 & 2 & 4 & 4 & 20 \end{array}\right| \quad \begin{matrix} a_j \\ \\ \\ \\ \end{matrix}$$

b_i 30 30 10 20

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№8

$$C = \left|\begin{array}{cccc|c} 1 & 4 & 7 & 3 & 50 \\ 5 & 6 & 8 & 9 & 90 \\ 7 & 2 & 4 & 8 & 20 \end{array}\right| \quad \begin{matrix} a_j \\ \\ \\ \\ \end{matrix}$$

b_i 50 70 20 40

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№9

					a_j
C =	1	3	6	2	30
	9	1	8	3	40
	7	5	3	1	70
b_i	35	40	25	70	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№10

					a_j
C =	5	8	4	4	80
	1	2	3	8	45
	4	7	6	1	60
b_i	45	60	70	40	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№11

					a_j
C =	4	5	5	7	10
	8	7	5	4	20
	1	6	4	5	50
	3	2	1	3	30
b_i	40	30	20	40	

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№12

					a_j
C =	2	7	3	6	30
	3	4	5	7	70
	5	7	6	2	50
b_i	20	40	35	60	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№13

					a_j
C =	1	3	3	8	20
	8	6	2	6	20
	2	7	3	1	40
	5	2	4	5	45
b_i	25	30	40	15	

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№14

					a_j
C =	1	3	6	2	70
	9	1	8	3	40
	7	5	3	1	30
b_i	35	50	20	70	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№15

					a_j
C =	3	1	4	7	30
	7	3	5	8	85
	6	3	4	6	45
b_i	40	35	15	60	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№16

					a_j
C =	4	2	4	1	50
	2	3	6	5	30
	6	2	4	1	20
b_i	30	30	10	20	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№17

					a_j
C =	3	1	6	5	20
	7	3	5	6	70
	6	7	5	3	45
b_i	40	30	25	50	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№18

					a_j
C =	1	2	4	1	50
	6	3	1	5	30
	3	6	4	2	30
b_i	20	30	15	30	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№19

				a_j
C =	3	5	6	30
	6	4	2	80
	7	6	9	20
	2	10	4	20
b_i	50	40	45	

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,3)-объем потребления i-го предприятия.

№20

					a_j
C =	2	5	6	1	30
	1	3	2	3	80
	4	1	5	2	40
b_i	60	30	50	30	

где C - матрица тарифов,

a_j (j=1,5) - объем производства j-го предприятия,

b_j (j=1,3) - объем потребления i-го предприятия.

№21

					a_j
C =	1	3	5	3	80
	5	3	1	2	60
	2	1	4	2	70
b_i	20	40	60	70	

где C-матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№22

				a_j
C =	1	4	2	10
	6	2	1	15
	5	3	7	20
	5	4	3	25
b_i	25	15	20	

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,3) - объем потребления i-го предприятия.

№23

					a_j
C =	5	2	6	1	50
	2	3	1	5	30
	3	2	4	4	20
b_i	30	30	15	20	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№24

					a_j
C =	1	4	6	3	50
	5	6	8	3	70
	5	2	4	8	20
b_i	50	60	20	30	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№25

					a_j
C =	4	5	5	7	80
	8	7	5	4	20
	3	2	9	3	30
b_i	40	20	70	20	

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№26

					a_j
C =	2	7	3	6	30
	3	4	5	7	70
	5	7	6	2	50
b_i	15	40	30	60	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№27

					a_j
C =	1	3	3	8	20
	7	7	3	8	40
	5	2	4	5	50
b_i	25	30	40	20	

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№28

					a_j
C =	1	3	6	2	30
	9	1	8	3	40
	7	5	3	1	70
b_i	35	30	20	70	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№29

					a_j
C =	3	1	7	5	30
	7	3	5	6	70
	6	6	4	3	40
b_i	40	35	15	60	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№30

					a_j
C =	1	2	4	1	50
	2	3	1	5	30
	3	2	4	4	10
b_i	30	40	10	20	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№31

				a_j
C =	3	5	6	30
	4	4	2	80
	7	6	9	20
	12	10	4	20
b_i	50	40	60	

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,3) - объем потребления i-го предприятия.

№32

					a_j
C =	4	5	6	8	30
	10	3	2	3	50
	4	10	5	1	40
b_i	10	30	50	40	

где C - матрица тарифов,

a_j (j=1,5) - объем производства j-го предприятия,

b_j (j=1,3) - объем потребления i-го предприятия.

№33

					a_j
C =	2	3	4	3	80
	5	3	1	2	60
	2	1	4	2	70
b_i	20	40	60	80	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№34

				a_j
C =	1	4	2	10
	6	2	7	15
	5	4	7	20
	5	4	8	25
b_i	25	15	20	

где C - матрица тарифов,

a_j (j=1,4) - объем производства j-го предприятия,

b_j (j=1,3) - объем потребления i-го предприятия.

№35

					a_j
C =	1	2	4	1	50
	2	3	1	5	30
	3	2	4	4	10
b_i	30	30	10	20	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

№36

					a_j
C =	1	4	7	3	50
	5	6	8	9	60
	7	2	4	8	20
b_i	70	50	20	10	

где C - матрица тарифов,

a_j (j=1,3) - объем производства j-го предприятия,

b_j (j=1,4) - объем потребления i-го предприятия.

ЛАБОРАТОРНАЯ РАБОТА № 7

Тема: Динамическое программирование. Задача об умножении последовательности матриц

Цель: найти такую (полную) расстановку скобок в произведении матриц, чтобы вычисление произведения требовало наименьшего числа умножений

Краткие теоретические сведения

Если имеется, например, 4 матрицы, которые будут перемножаться, то для этих четырех матриц $A_1A_2A_3A_4$ массив p задает размеры матриц:

<$p_0\ p_1p_2p_3p_4$> то есть матрицы будут иметь размеры:

$A_1 = (p_0 x p_1)\quad A_2 = (p_1 x p_2)\quad A_3 = (p_2 x p_3)\quad A_4 = (p_3 x p_4)$

Следовательно, необходимо найти оптимальный способ расстановки скобок при умножении, при котором количество операций умножения будет минимальным.

Если умножаем матрицы $A_1A_2A_3$, то при этом существует 2 варианта умножения:

$(A_1A_2)A_3$ и $A_1(A_2A_3)$

при размерах матриц A_1 (5x10), A_2 (10x15), A_3 (15x5).

Для первого варианта количество перемножений будет равно n=5x10x15 + 5x10x15 = 1500, а для второго n = 10x15x5 + 5x10x5 = 1000, откуда видно, что в данном случае второй вариант оптимальнее по количеству операции, и соответственно - скорости работы.

Перемножение двух матриц

```
MATRIX-MULTIPLY (A, B)
1 if columns[A] ≠ rows[B]
2       then error «умножить нельзя»
3       else for r ← 1 to rows [A]
4                 do for j ← 1 to columns [B]
5                           do C [i, j] ← 0
6                                     for k ← 1 to columns [A]
7                                               do C [i, j] ← C [i, j] + A [i, k] · B [k, j]
8 return C
```

Вычисление оптимальной стоимости

```
MATRIX-CHAIN-ORDER(p)
1   n ← length[p] – 1
2   for i ←1 to n
3           do m[i, i] ← 0
4   for l ←2 to n
5           do for i ←1  to n – l + 1
6                   do j ← i + l – 1
7                         m[i, j] ← ∞
8                 for k ← i to j – 1
9                   do q← m[i, k] + m[k+1, j] + p_{i-1} p_k p_j
10                        if q < m[i, j]
11                          then m[i, j] ← q
12                                  s[i, j] ← k
13  return m, s
```

Построение оптимального решения

```
MATRIX- CHAIN- MULTIPLY (A, s, i, j)
1     if j > i
2        then X ← MATRIX-CHAIN-MULTIPLY (A, s,i, s[i, j])
3           У ← MATRIX-CHAIN-MULTIPLY (A, s, s[i, j] + 1, j)
4           return MATRIX-MULTIPLY (X, У)
5        else return A_i
```

Распечатка скобок в произведении последовательности матриц

```
PRINT_OPTIMAL_PARENS (S, i, j)

1   if i = j
2      then print "A" i
3      else print "("
4      PRINT_OPTIMAL_PARENS (S, i, S[i, j])
5      PRINT_OPTIMAL_PARENS (S, S[i, j]+1, j)
6      PRINT ")"
```

Пример

Найти оптимальную расстановку скобок в произведении последовательности матриц, размерности которых равны

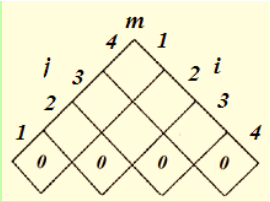
p = (4, 10, 15, 8, 5), A_1 = (4 x 10);

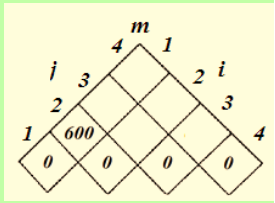
A_2 = (10 x 15); A_3 = (15 x 8); A_4 = (8 x 5)

Дано: p = < 4, 10, 15, 8, 5 >, n= 4

p_0 p_1 p_2 p_3 p_4

```
MATRIX-CHAIN-ORDER(p)
1   n ← length[p] - 1
2   for i ←1 to n
3    do  m[i, j] ← O
4   for l ←2 to n
5    do for i ←1  to n - l + 1
6          do j ← i + l – 1
7                m[i, j] ← ∞
8                for k ← i to j - 1
9                   do q-← m[i, k] + m[k+1, j] + p_{i-1} p_k p_j
10                    if q < m[i, j]
11                       then m[i, j] ← q
12                            s[i, j] ← k
13  return m, s
```





Трассировка:

1 n = 4

2 for I = $\overline{1, 4}$

3 m[1, 1] = ∅, m[2, 2] = ∅,
m[3, 3] = ∅, m[4, 4] = ∅,

4 for l = $\overline{2, 4}$

5 do for i=1 to 4 – 2 + 1 = 3

6 do j=1 + 2 – 1 = 2

7 m[1, 2] ← ∞

8 for k ← 1 to 2 – 1 = 1

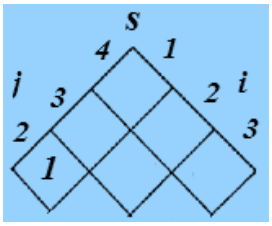
9 do q ← m[1, 1] + m[2, 2] + $p_0p_1p_2$ = 0 + 0 + 4·10·15 = 600

10 if 600 < ∞

11 then m[1, 2] ← 600

12 s[1, 2] ← 1

13 return m[1, 2] ← 600 s[1, 2] ← 1



5 do for i =2 to 4 – 2 + 1 = 3

6 do j=2 + 2 – 1 = 3

7 m[2, 3] ← ∞

8 for k ← 2 to 3 – 1 = 2

9 do q ← m[2, 2] + m[3, 3] + $p_1p_2p_3$ = 0 + 0 +10·15·8 =1200

10 if 1200 < ∞

11 then m[2, 3] ← 1200

12 s[2, 3] ← 2

13 return m[2, 3] ← 1200 s[2, 3] ← 2

5 do for i=3 to 4 – 2 + 1 = 3

6 do j=3 + 2 – 1 = 4

7 m[3, 4] ← ∞

8 for k ← 3 to 4 – 1 = 3

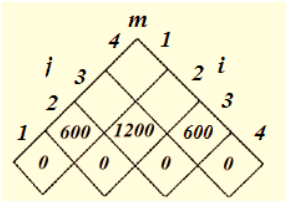
9 do q ← m[3, 3] + m[4, 4] + $p_2p_3p_4$ = 0 + 0 + 15·8·5 = 600

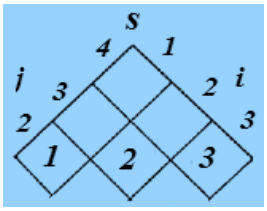
10 if 600 < ∞

11 then m[3, 4] ← 600

12 s[3, 4] ← 3

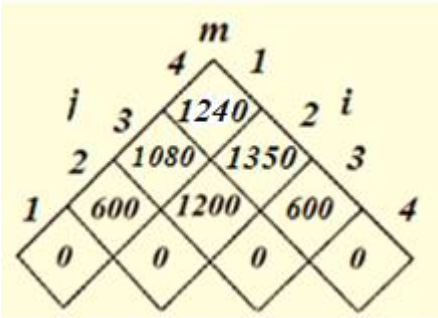
13 return m[3, 4] ← 600 s[3, 4] ← 3

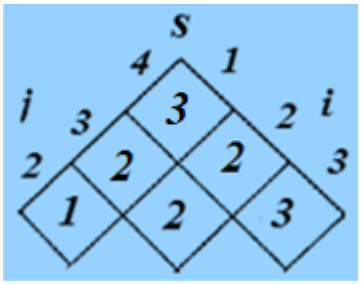




и т. д.

В результате получаем:





Расставим скобки в последовательности перемножения матриц.

```
PRINT_OPTIMAL_PARENS (S, i, j)
1. if i = j
2.     then print "A" i
3.     else print "("
4.     PRINT_OPTIMAL_PARENS (S, i, S[i, j])
5.     PRINT_OPTIMAL_PARENS (S, S[i, j]+1, j)
6.     PRINT ")"
```

PRINT_OPTIMAL_PARENS (S, 1, 4)

1. if 1 ≠ 4

3. else print "("

4. PRINT_OPTIMAL_PARENS (S, 1, S[1, 4]=3)

4 PRINT_OPTIMAL_PARENS (S, 1, 3)

1. if 1 ≠ 3

3. else print "("

4. PRINT_OPTIMAL_PARENS (S, 1, S[1, 3]=2)

4 PRINT_OPTIMAL_PARENS (S, 1, 2)

1. if 1 ≠ 2

3. else print "("

4. PRINT_OPTIMAL_PARENS (S, 1, S[1, 2]=1)

4 PRINT_OPTIMAL_PARENS (S, 1, 1)

1. if 1 = 1

2. then print "A" 1

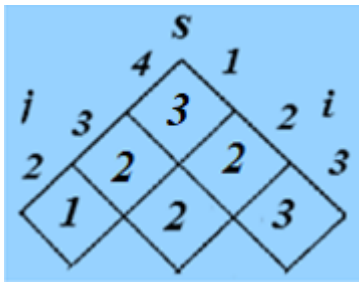
5 PRINT_OPTIMAL_PARENS (S, 2, 2)

1. if 2 = 2

2. then print "A" 2

6. PRINT ")"

((



(((A1A2)

5 PRINT_OPTIMAL_PARENS (S, 3, 3)

1. if 3 = 3
2. then print “A” 3

(((A1A2) A3)

6. PRINT “)”

```
PRINT_OPTIMAL_PARENS (S, i, j)
1. if i = j
2.     then print “A” i
3.     else print “(”
4.     PRINT_OPTIMAL_PARENS (S, i, S[i, j])
5.     PRINT_OPTIMAL_PARENS (S, S[i, j]+1, j)
6.     PRINT “)”
```

5 PRINT_OPTIMAL_PARENS (S, 4, 4)

1. if 4 = 4
2. then print “A” 4

(((A1A2) A3)A4)

6. PRINT “)”

Вычисление оптимальной стоимости

m[1, 1] = ∅, m[2, 2] = ∅, m[3, 3] = ∅, m[4, 4] = ∅;

$m[1, 2] = m[1, 1] + m[2, 2] + p_0p_1p_2 = 0 + 0 + 4 \cdot 10 \cdot 15 = 600$; k =1;

$m[2, 3] = m[2, 2] + m[3, 3] + p_1p_2p_3 = 0 + 0 + 10 \cdot 15 \cdot 8 = 1200$; k =2;

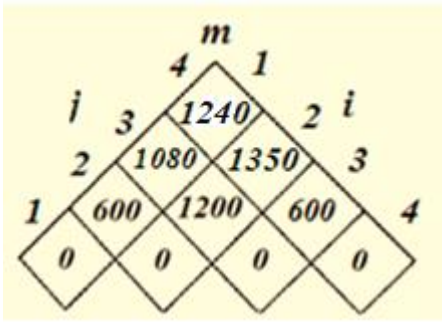
$m[3, 4] = m[3, 3] + m[4, 4] + p_2p_3p_4 = 0 + 0 + 15 \cdot 8 \cdot 5 = 600$; k = 3;

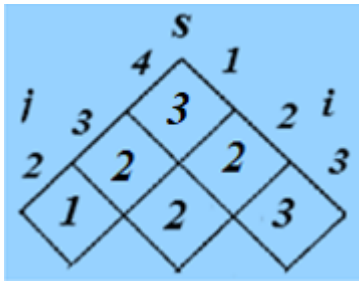
$$m[1, 3] = \min_{1 \le k < 3} \begin{cases} m[1,1] + m[2,3] + p_0p_1p_3 = 0 + 1200 + 4 \cdot 10 \cdot 8 = 1520, & k = 1; \\ m[1,2] + m[3,3] + p_0p_2p_3 = 600 + 0 + 4 \cdot 15 \cdot 8 = 1080, & k = 2; \end{cases}$$

$$m[2, 4] = \min_{1 \le k < 3} \begin{cases} m[2,2] + m[3,4] + p_1p_2p_4 = 0 + 600 + 10 \cdot 15 \cdot 5 = 1350, & k = 2; \\ m[2,3] + m[4,4] + p_1p_3p_4 = 1200 + 600 + 10 \cdot 8 \cdot 5 = 2200, & k = 3; \end{cases}$$

$$m[1, 4] = \min_{1 \le k < 4} \begin{cases} m[1,1] + m[2,4] + p_0p_1p_4 = 0 + 1350 + 4 \cdot 10 \cdot 5 = 1550, & k = 1; \\ m[1,2] + m[3,4] + p_0p_2p_4 = 600 + 600 + 4 \cdot 15 \cdot 5 = 1500, & k = 2; \\ m[1,3] + m[4,4] + p_0p_3p_4 = 1080 + 0 + 4 \cdot 8 \cdot 5 = 1240, & k = 3; \end{cases}$$

Результаты работы алгоритмов





(((A1A2) A3) A4)

Задания к лабораторной работе:

1. Реализовать алгоритм вычисления оптимальной стоимости MATRIX-CHAIN-ORDER(p).
2. Реализовать алгоритм построения оптимального решения MATRIX-CHAIN-MULTIPLY(A, s, i, j).
3. Реализовать алгоритм PRINT_OPTIMAL_PARENS (S, i, j), печатающий оптимальную расстановку скобок.

Найдите оптимальную расстановку в задаче о перемножении матриц, и выведите ответ на экран, если

№ варианта	Массив P
1	<5, 10, 3, 12, 6>
2	<12, 5, 50, 6, 3>
3	<10, 3, 5, 12, 4>
4	<12, 30, 5, 6, 10>
5	<3, 10, 5, 50, 5>
6	<50, 5, 10, 6, 4>
7	<4, 8, 3, 20, 5>
8	<20, 4, 8, 5, 10>
9	<10, 5, 20, 15, 10>
10	<10, 3, 5, 12, 4>
11	<2, 10, 5, 15, 10>
12	<10, 15, 5, 12, 4>
13	<10, 4, 5, 15, 8>
14	<6, 8, 4, 10, 15>
15	<10, 8, 5, 12, 5>
16	<4, 10, 6, 8, 5>
17	<4, 8, 6, 10, 20>
18	<10, 4, 10, 12, 5>
19	<6, 4, 15, 8, 5>
20	<4, 6, 8, 10, 5>
21	<4, 10, 8, 5, 10>
22	<5, 8, 6, 10, 5>
23	<6, 10, 15, 10, 5>
24	<4, 10, 15, 10, 4>
25	<15, 4, 8, 10, 15>
26	<4, 15, 20, 10, 5>
27	<10, 6, 15, 10, 5>
28	<10, 5, 20, 15, 10>
29	<4, 8, 3, 20, 5>
30	<20, 4, 8, 5, 10>

Содержание отчета:

1. Привести, используя псевдокоды, перечисленные ниже алгоритмы:
   - вычисление оптимальной стоимости;
   - построение оптимального решения;
   - вывод решения на экран (расстановка скобок).
2. Продемонстрировать эти алгоритмы на примере (см. задание к лабораторной работе).
   В данном случае требуется привести полученный массив стоимостей (если возможно – на каждом шагу) и массив для вычисления оптимального решения, а также само решение в виде списка матриц, с расставленным скобками.

   Пример:

   Для $A_1 = (5\text{x}10),\ \ A_2 = (10\text{x}15),\ \ A_3 = (15\text{x}5)$

   m

   1000 1000 0
   1500 0
   0 – массив стоимостей

   s

   2 2 0
   1 0
   0 – массив для расстановки скобок

   Вывести решение.
   $(A_1(A_2A_3))$ - решение.

3. Результаты работы программ.
4. Вывод.
5. Листинги программ.

Контрольные вопросы:

1. Какое значение имеет данный подход к построению алгоритмов в вопросе оптимизации работы программы?
2. Каким образом заполняется массив стоимостей каждого перемножения?
3. Почему требуется запоминать минимальную стоимость каждого перемножения?
4. Знаете ли вы другие подходы для решения данной проблемы и каковы они?
5. Всегда ли есть возможность расставить скобки?
6. К какому типу оптимизации алгоритмов относится данная задача?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. T. CORMEN, Ch. LEISERSON, R. RIVEST. Introducere în algoritmi. Computer Libris Agora, Cluj-Napoca, 2000.
2. OLTEAN Mihai. Proiectarea şi implementarea algoritmilor. Computer Libris Agora, Cluj-Napoca, 1999.
3. КОРМЕН Т., ЛЕЙЗЕРСОН Ч., РИВЕСТ Р. Алгоритмы: построение и анализ. – М.: МЦНМО, 2001. – 960с.
4. Donald E. KNUTH. Seminumerical Algorithms, volume 2 of The Art of Computer Programming. Addison-Wesley, 1981.
5. А. АХО, Д. УЛЬМАН, Д. ХОПКРОФТ. Структуры данных и алгоритмы. Издательский дом «Вильямс», 2000 – 348 с.
6. Хэзфилд Р. Кирби Л. и др. Искусство программирования на С. Фундаментальные алгоритмы, структуры данных и примеры приложений. Энциклопедия программиста. К.: Издательство “ДиаСофт”, 2001. – 736 с.
7. Зайченко Ю.Н. Исследование операций. Учебное пособие для студентов ВУЗов. - Киев: Вища школа, 1979 (1975), 4(14) экз.
8. Вентцель Е.С. Исследование операций. - М.: Советское радио, 1972, 8 экз.
9. Кузнецов Ю.Н., Кузубов В.Н., Волощенко А.Б. Математическое программирование. - М.: Высшая школа, 1980, 34 экз.
10. Костевич Л.С., Лапко А.А. Теория игр. Исследование операций. - Минск: Высшейшая школа, 1982, 66 экз.