Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Лабораторные работы

Математическое программирование

Выполнила:

Студентка 2 курса 11 группы

Дрожжа Анастасия Дмитриевна

Минск 2021

**Лабораторная работа 4**

**ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**

**ЦЕЛЬ РАБОТЫ:** освоить общие принципы решения задач методом динамического программирования, сравнить полученные решения задач с рекурсивным методом.

**Условие:**

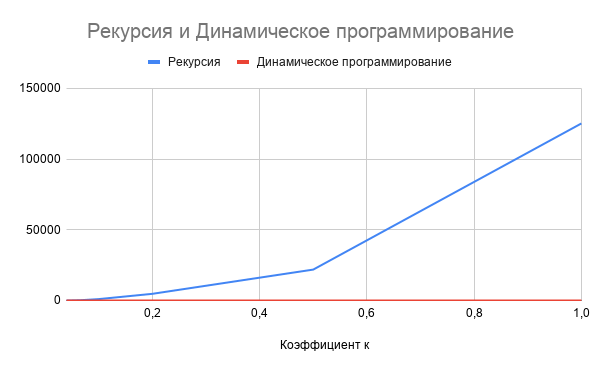
**Задание 1.** На языке С++ сгенерировать случайным образом строку букв латинского алфавита  длиной  символов и длиной .

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  int main()  {  printf("GENERATE 300 SYMBOLS\n\n");  int i;  srand(time(NULL));  unsigned char c;  for (i = 0; i <= 300; ++i)  {  c = (rand() % ('z' - 'a' + 1)) + 'a';  printf("%c", c);  }  printf("\n\nGENERATE 250 SYMBOLS\n\n");  srand(time(NULL));  unsigned char cC;  for (i = 0; i <= 300; ++i)  {  cC = (rand() % ('z' - 'a' + 1)) + 'a';  printf("%c", cC);  }  return 0;  } |
|  |

**Задание 2.** Вычислить двумя способами (рекурсивно и с помощью динамического программирования)  – дистанцию Левенштейна для k=0,04; 0,05; 0,07; 0,1; где - длина строки ,  - строка состоящая из первых  символов строки . (копии экрана и код вставить в отчет).

|  |
| --- |
| // - Levenshtein.h  // -- дистанции Левенштeйна (динамическое программирование)  int levenshtein(  int lx, // длина слова x  const char x[], // слово длиной lx  int ly, // длина слова y  const char y[] // слово y  );    // -- дистанции Левенштeйна (рекурсия)  int levenshtein\_r(  int lx, // длина строки x  const char x[], // строка длиной lx  int ly, // длина строки y  const char y[] // строка y  );  // - Levenshtein.cpp  #include "stdafx.h"  #include <iomanip>  #include <algorithm>  #include "Levenshtein.h"  #define DD(i,j) d[(i)\*(ly+1)+(j)]  int min3(int x1, int x2, int x3)  { return std::min(std::min(x1,x2),x3); }  int levenshtein(int lx, const char x[],int ly, const char y[])  {  int \*d = new int[(lx+1)\*(ly+1)];  for(int i = 0; i <= lx; i++) DD(i, 0) = i;  for(int j = 0; j <= ly; j++) DD(0, j) = j;  for (int i = 1; i <= lx; i++)  for (int j = 1; j <= ly; j++)  {  DD(i,j) = min3(DD(i-1, j) + 1, DD(i, j-1) + 1,  DD(i-1, j-1) + (x[i-1]==y[j-1]?0:1));  }  return DD(lx,ly);  }  int levenshtein\_r(  int lx, const char x[],  int ly, const char y[]  )  {  int rc = 0;  if (lx == 0) rc = ly;  else if (ly == 0) rc = lx;  else if (lx == 1 && ly == 1 && x[0] == y[0]) rc = 0;  else if (lx == 1 && ly == 1 && x[0] != y[0]) rc = 1;  else rc = min3(  levenshtein\_r(lx-1, x, ly, y)+1,  levenshtein\_r(lx, x, ly-1, y)+1,  levenshtein\_r(lx-1, x, ly-1, y)+(x[lx-1] == y[ly-1]?0:1)  );  return rc;  };  // --- main  // вычисление дистанции (расстояния) Левенштейна  #include "stdafx.h"  #include <algorithm>  #include <iostream>  #include <ctime>  #include <iomanip>  #include "Levenshtein.h"  int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_ALL, "rus");  clock\_t t1 = 0, t2 = 0, t3, t4;  char x[] = "abcdefghklmnoxm", y[] = "xyabcdefghomnkm";  int lx = sizeof(x)-1, ly = sizeof(y)-1;  std::cout<<std::endl;  std::cout<<std::endl<< "-- расстояние Левенштейна -----"<< std::endl;  //std::cout << std::endl << x << "->" << y << " = " << levenshtein\_r(sizeof(x)-1, x, sizeof(y)-1, y) << std::endl;  std::cout<<std::endl<< "--длина --- рекурсия -- дин.програм. ---"<<std::endl;  for (int i = 8; i < std::min(lx,ly); i++)  {  t1 = clock();  levenshtein\_r(i,x,i-2, y);  t2 = clock();  t3 = clock();  levenshtein(i,x,i-2, y);  t4 = clock();  std::cout<<std::right<<std::setw(2)<<i-2<<"/"<<std::setw(2)<<i  << " "<<std::left<<std::setw(10)<<(t2-t1)  <<" "<<std::setw(10)<<(t4-t3)<<std::endl;  }  system("pause");  return 0;  } |
|  |

**Задание 3.** Выполнить сравнительный анализ времени затраченного на вычисление дистанции Левенштейна для двух методов решения. Построить графики зависимости времени вычисления от . (копии экрана и график вставить в отчет).



**Задание 4.** Реализовать вручную пример вычисления дистанции Левенштейна при помощи рекурсивного алгоритма (в соответствии с вариантом) (каждый шаг алгоритма по примеру из лекции вставить в отчет).

1. **Вычисление дистанции Левенштейна**



 – количество символов в заданной строке. Например, 

 – заданная строка без последнего символа. Например, 

 – последний символ заданной строки. Например, 

|  |  |  |
| --- | --- | --- |
| 8 | Вар | Баран |

Пусть необходимо вычислить L (“вар”, “баран”) Тогда имеем следующую последовательность шагов:

1. L (“вар”, “баран”) =
2. L (“ва”, “баран”) =
3. L (“вар”, “бара”) =
4. L (“ва”, “бара”) =
5. L (“в”, “баран”) =

L (“”, “баран”)=5, L (“”, “бара”)=4

1. L (“в”, “бара”) =

L (“”, “бара”)=4, L (“”, “бар”)=3

1. L (“вар”, “бар”) =
2. L (“ва”, “бар”) =
3. L (“вар”, “ба”) =
4. L (“вар”, “б”) =

L (“вар”, “”)=3, L (“ва”, “”)=2

1. L (“в”, “бар”) =

L (“”, “бар”)=3, L (“”, “ба”)=2

1. L (“ва”, “ба”) =

L (“в”, “б”)=1

1. L (“в”, “ба”) =

L (“”, “ба”)=2, L (“в”, “б”)=1, L (“”, “б”)=1

1. L (“ва”, “б”) =

L (“в”, “б”)=1, L (“ва”, “”)=2, L (“в”, “”)=1

1. L (“ва”, “б”)=min(2, 3, 2)=2
2. L (“в”, “ба”)=min(3, 2, 2 )=2
3. L (“ва”, “ба”)=min(2, 2, 1)=1
4. L (“в”, “бар”)=min(4, 3, 3)=3
5. L (“вар”, “б”)=min(3, 4, 3)=3
6. L (“вар”, “ба”)=min(2, 4, 3)=2
7. L (“ва”, “бар”)=min(4, 2, 3)=2
8. L (“вар”, “бар”)=min(3, 3, 1)=1
9. L (“в”, “бара”)=min(5, 4, 4)=4
10. L (“в”, “баран”)=min(6, 5, 5)=5
11. L (“ва”, “бара”)=min(5, 3, 3)=3
12. L (“вар”, “бара”)=min(4, 2, 3)=2
13. L (“ва”, “баран”)=min(6, 4, 5)=4
14. L (“вар”, “баран”)=min(5, 3, 4)=3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | \* | б | ба | бар | бара | баран |
| \* | 0 | 1 | 2 | 3 | 4 | 5 |
| в | 1 | 1 | 2 | 3 | 4 | 5 |
| ва | 2 | 2 | 2 | 2 | 3 | 4 |
| вар | 3 | 2 | 2 | 2 | 2 | 3 |

Таким образом дистанция Левенштейна для данных строк равна 3.

**Задание 5.**

**Четные варианты**. Выполнить сравнительный анализ времени затраченного на решение задачи об оптимальной расстановке скобок при умножении нескольких матриц для двух методов решения (рекурсивное решение, динамическое программирование). Размерность матриц взять в соответствии с вариантом. Объяснить в отчете принцип расставления скобок по итоговой матрице + код + копии экрана.

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | Вар | Баран | 10\*15, 15\*80, 80\*23, 23\*50, 50\*40, 40\*71 |

|  |
| --- |
| #define OPTIMALM\_PARM(x) ((int\*)x) // для представления 2мерного массива  int OptimalM( // рекурсия  int i, // [in] номер первой матрицы  int j, // [in] номер последней матрицы  int n, // [in] количество матриц  const int c[], // [in] массив размерностей  int\* s // [out] результат: позиции скобок  );  int OptimalMD( // динамическое программирование  int n, // [in] количество матриц  const int c[], // [in] массив размерностей  int\* s // [out] результат: позиции скобок  );  // расстановка скобок (рекурсия)  #include"stdafx.h"  #include<memory.h>  #include"MultiMatrix.h"  #define INFINITY 0x7fffffff  #define NINFINITY 0x80000000  int OptimalM(int i, int j, int n, constint c[], int \*s)  {  #define OPTIMALM\_S(x1,x2) (s[(x1-1)\*n+x2-1])  int o =INFINITY, bo = INFINITY;  if (i<j)  {  for (int k = i; k<j;k++)  {  bo = OptimalM(i,k, n, c, s)+  OptimalM(k+1,j,n, c, s)+ c[i- 1]\*c[k]\*c[j];  if (bo < o)  {  o = bo;  OPTIMALM\_S(i,j) = k;  }  }  }  else o = 0;  return o;  #undef OPTIMALM\_S  };  // --- MultyMatrix.cpp (продолжение)  // расстановка скобок (динамическое программирование)  int OptimalMD(int n, const int c[], int\* s)  {  #define OPTIMALM\_S(x1,x2) (s[(x1-1)\*n+x2-1])  #define OPTIMALM\_M(x1,x2) (M[(x1-1)\*n+x2-1])  int \*M = new int[n\*n], j = 0, q = 0;  for (int i = 1; i <= n; i++) OPTIMALM\_M(i,i) = 0;  for (int l = 2; l <= n; l++)  {  for (int i = 1; i <= n-l+1; i++)  {  j = i+l-1;  OPTIMALM\_M(i,j) = INFINITY;  for (int k = i; k <= j-1; k++)  {  q = OPTIMALM\_M(i,k) + OPTIMALM\_M(k+1,j)+c[i-1]\*c[k]\*c[j];  if (q < OPTIMALM\_M(i,j))  {  OPTIMALM\_M(i,j) = q; OPTIMALM\_S(i,j)= k;  }  }  }  }  return OPTIMALM\_M(1,n);  #undef OPTIMALM\_M  #undef OPTIMALM\_S  };  // --- main  // расстановка скобок  #include"stdafx.h"  #include<cmath>  #include<memory.h>  #include<iostream>  #include"MultiMatrix.h"// умножение матриц  #define N 6  int main()  {  int Mc[N+1] = {10, 15, 80, 23, 50, 40, 71}, Ms[N][N], r = 0, rd = 0;  memset(Ms,0,sizeof(int)\*N\*N);  r = OptimalM(1, N, N, Mc, OPTIMALM\_PARM(Ms));  setlocale(LC\_ALL, "rus");  std::cout<<std::endl;  std::cout<<std::endl<<"-- расстановка скобок (рекурсивное решение) "<< std::endl;  std::cout<<std::endl<<"размерности матриц: ";  for (int i = 1; i <= N; i++) std::cout<<"("<<Mc[i-1]<<","<<Mc[i]<<") ";  std::cout<<std::endl<<"минимальное количество операций умножения: "<< r;  std::cout<<std::endl<<std::endl<<"матрица S"<<std::endl;  for (int i = 0; i < N; i++)  {  std::cout<<std::endl;  for (int j = 0; j < N; j++) std::cout<<Ms[i][j]<<" " ;  }  std::cout<<std::endl;  memset(Ms,0,sizeof(int)\*N\*N);  rd = OptimalMD(N, Mc, OPTIMALM\_PARM(Ms));  std::cout<<std::endl  <<"-- расстановка скобок (динамичеое программирование) "<< std::endl;  std::cout<<std::endl<<"размерности матриц: ";  for (int i = 1; i <= N; i++)  std::cout<<"("<<Mc[i-1]<<","<<Mc[i]<<") ";  std::cout<<std::endl<<"минимальное количество операций умножения: "<< rd;  std::cout<<std::endl<<std::endl<<"матрица S"<<std::endl;  for (int i = 0; i < N; i++)  {  std::cout<<std::endl;  for (int j = 0; j < N; j++) std::cout<<Ms[i][j]<<" " ;  }  std::cout<<std::endl<<std::endl;  system("pause");  return 0;  } |
|  |

Принцип расстановки скобок по итоговой матрице:

Скобки расставляются по принципу «сначала внешние – затем внутренние». Имеется 6 матриц, вот их размерность:

А1=10,

А2=15,

А3=80,

А 4 =23,

А 5 =50,

А 6 =71.

Найдем элемент (1,6) в матрице S, он равен 5. Это означает, что точка разрыва между 1-ой и 6-ой матрицей находится после 5-ой матрицы. Что позволяет расставить скобки следующим образом:

(A1\*A2\*A3\*A4\*A5)\*A6

Точку разрыва между первой и пятой матрицей определяет элемент (1,5). Он равен 4. Следовательно разрыв будет после четвертой матрицы.

((A1\*A2\*A3\*A4)\*A5)\*A6

Далее берем элемент (1,4) и получаем, что он равен 3. Следовательно получаем:

(((A1\*A2\*A3)\*A4)\*A5)\*A6

И на последнем шаге мы возьмем элемент (1,3) и он равен 2:

(((A1\*A2)\*A3)\*A4)\*A5)\*A6

Это выражение и есть конечное.

Полученная расстановка скобок позволяет получить минимальное количество операций умножения, равное 90300.

**Вывод:** освоили общие принципы решения задач методом динамического программирования, сравнили полученные решения задач с рекурсивным методом.

**Лабораторная работа №5**

**АЛГОРИТМЫ НА ГРАФАХ**

**(алгоритмы поиска в ширину и глубину, топологическая сортировка)**

**Задание 1.**  Ориентированный граф **G** взять в соответствии с вариантом. Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на лекциях. Оформить отчет, включив в него **каждый** шаг выполнения алгоритмов.

***Поиск в ширину:***

Текущее состояние алгоритма хранится в следующих структурах памяти:

**Q** – очередь вершин,

**С** – массив окраски вершин,

**D** – массив расстояний

**P** – массив предшествующих вершин.

По условию, граф имеет 5 вершин, пронумерованных начиная с нуля. В качестве стартовой вершины выбрана вершина с номером 0.

Шаг 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 0 |  |  |  |  |
| C | G | W | W | W | W |
| D | 0 | I | I | I | I |
| P | N | N | N | N | N |

Шаг 2 Извлекаем из очереди 0-ую вершину и закрашиваем в black (B) цвет; помещаем 1-ую и 4 вершину в очередь.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 1 | 4 |  |  |  |
| C | B | G | W | W | G |
| D | 0 | 1 | I | I | 1 |
| P | N | 0 | N | N | 0 |

Шаг 3 Извлекаем 4-ую вершину, красим в B-цвет и помещаем 2 и вершину в очередь.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 2 |  |  |  |  |
| C | B | B | G | W | B |
| D | 0 | 1 | 2 | I | 1 |
| P | N | 0 | 1 | N | 0 |

Шаг 3 Извлекаем 2-ую вершину, красим в B-цвет и помещаем 3 и вершину в очередь.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 2 |  |  |  |  |
| C | B | B | B | G | B |
| D | 0 | 1 | 2 | 3 | 1 |
| P | N | 0 | 1 | 2 | 0 |

Шаг 4 Извлекаем 3-ую вершину, красим в B-цвет. Все вершины черные, а значит алгоритм закончил свою работу.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 2 |  |  |  |  |
| C | B | B | B | B | B |
| D | 0 | 1 | 2 | 3 | 1 |
| P | N | 0 | 1 | 2 | 0 |

Ответ: Все вершины чёрного цвета, соответственно, алгоритм закончил свою работу. В результате получили вот такое BFS-дерево:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | N | 0 | 1 | 2 | 0 |

***Поиск в глубину:***

Текущее состояние алгоритма хранится в следующих структурах памяти:

C – массив окраски вершин,

D – время окраски вершин в серый цвет,

P – массив предшествующих вершин,

F – время окраски в чёрный цвет.

Кроме того, используется переменная t, текущее значение которой – номер шага алгоритма.

По условию, граф имеет 5 вершин, пронумерованных начиная с нуля. В качестве стартовой вершины выбрана вершина с номером 0.

алгоритмов.

Также вводится дополнительная переменная t-номер операции алгоритма.

Шаг 1

Т=1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | W | W | W | W |
| D | 1 | I | I | I | I |
| P | N | N | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 2

Т=2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | W | W | W |
| D | 1 | 2 | I | I | I |
| P | N | 0 | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 3

Т=3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | W | W |
| D | 1 | 2 | 3 | I | I |
| P | N | 0 | 1 | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 4

Т=4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | G | W |
| D | 1 | 2 | 3 | 4 | I |
| P | N | 0 | 1 | 2 | N |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 5

Т=5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | G | G |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 0 | 0 |

Шаг 6

Т=6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | G | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 0 | 6 |

Шаг 7

Т=7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 7 | 6 |

Шаг 8

Т=8

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 8 | 7 | 6 |

Шаг 9

Т=9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | B | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 9 | 8 | 7 | 6 |

Шаг 10

Т=10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | B | B | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 10 | 9 | 8 | 7 | 6 |

Ответ: DFS-дерево имеет вид:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | N | 0 | 1 | 2 | 3 |

**Топологическая рекурсия:**

По условию, граф имеет 5 вершин, пронумерованных начиная с нуля. В качестве стартовой вершины выбрана вершина с номером 0.

Топологическая сортировка – это процедура упорядочивания вершин ориентированного графа, не имеющего циклов.

При реализации топологической сортировки с помощью алгоритма поиска в глубину используется массив меток вершин, с помощью которого моделируется удаление вершин из графа и сохраняются новые номера вершин.

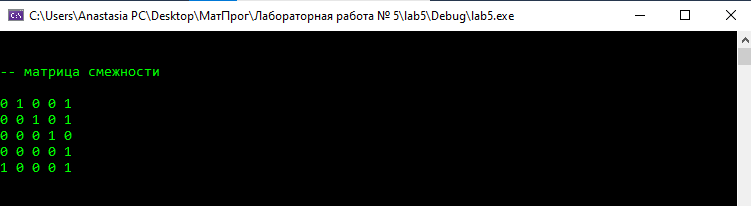
Шаг 1

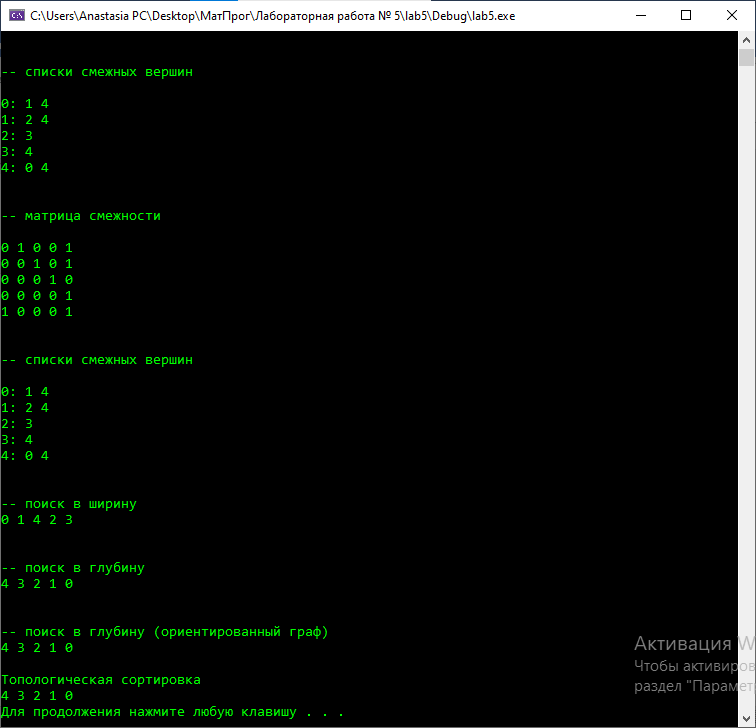
Шаг 2

Шаг 3

Шаг 4

Шаг 4



****

**Лабораторная работа 6. ТРАНСПОРТНАЯ ЗАДАЧА**

**ЦЕЛЬ РАБОТЫ:** Приобретение навыков решения открытой транспортной задачи

**Задание для выполнения:**

**Задание.** Решить транспортную задачу. Имеется 5 поставщиков продукции и 6 потребителей. Величина запасов, потребностей и стоимость затрат на перевозку продукции взять в соответствии с вариантом (*N*). Оформить отчет.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | запасы |
| 1 | **n+12** | **N+2** | **N+6** | **N+3** | **N+11** | **N+1** | **168+N** |
| 2 | **n+10** | **N** | **N+8** | **N+5** | **N+7** | **N+13** | **113+N** |
| 3 | **n+1** | **N+5** | **N+11** | **N+8** | **N+2** | **N+11** | **150+N** |
| 4 | **N+4** | **N+10** | **N+10** | **N+3** | **N+13** | **N+2** | **159+N** |
| 5 | **N+3** | **N+11** | **N+9** | **N** | **N+10** | **N+4** | **100+N** |
| Потребности | **143+N** | **107+N** | **131+N** | **193+N** | **95+N** | **163+N** |  |

Исходя из номера варианта получается следующая таблица:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | запасы |
| 1 | **20** | **10** | **14** | **11** | **19** | **9** | **176** |
| 2 | **18** | **8** | **16** | **13** | **15** | **21** | **121** |
| 3 | **9** | **13** | **19** | **16** | **10** | **19** | **158** |
| 4 | **12** | **18** | **18** | **11** | **21** | **10** | **167** |
| 5 | **11** | **19** | **17** | **8** | **18** | **12** | **108** |
| Потребности | **151** | **115** | **139** | **201** | **103** | **171** |  |

Для разрешимости транспортной задачи необходимо, чтобы суммарные запасы продукции у поставщиков равнялись суммарной потребности потребителей. Проверим это условие .Суммарная потребность потребителя 880, а поставщика 730. Разница составляет 150 единиц. Чтоб выполнялось условие введем фиктивного поставщика с запасом 150. Стоимость доставки единицы продукции от данного поставщика ко всем потребителям примем равной нулю.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | запасы |
| 1 | **20** | **10** | **14** | **11** | **19** | **9** | **176** |
| 2 | **18** | **8** | **16** | **13** | **15** | **21** | **121** |
| 3 | **9** | **13** | **19** | **16** | **10** | **19** | **158** |
| 4 | **12** | **18** | **18** | **11** | **21** | **10** | **167** |
| 5 | **11** | **19** | **17** | **8** | **18** | **12** | **108** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **150** |
| Потребности | **151** | **115** | **139** | **201** | **103** | **171** |  |

И теперь условие выполняется и можно рассматривать минимальные элементы тарифа, 0 не берем в расчет!!!

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | запасы |
| 1 | **20** | **10** | **14** | **11** | **19** | **9** | **176** |
| 2 | **18** | **8(115)** | **16** | **13** | **15** | **21** | **6** |
| 3 | **9** | **13** | **19** | **16** | **10** | **19** | **158** |
| 4 | **12** | **18** | **18** | **11** | **21** | **10** | **167** |
| 5 | **11** | **19** | **17** | **8** | **18** | **12** | **108** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **150** |
| Потребности | **151** | **0** | **139** | **201** | **103** | **171** |  |

Минимальный элемент матрицы тарифов находится в ячейке 2-2 и равен 8, т.е. из незадействованных маршрутов, маршрут доставки продукции от поставщика 2 к потребителю 2 наиболее рентабельный.

Запасы поставщика 2 составляют 121 единиц продукции. Потребность потребителя 2 составляет 115 единиц продукции.

От поставщика 2 к потребителю 2 будем доставлять min = {121 , 115 } = 115 единиц продукции.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | запасы |
| 1 | **20** | **10** | **14** | **11** | **19** | **9** | **176** |
| 2 | **18** | **8(115)** | **16** | **13** | **15** | **21** | **6** |
| 3 | **9** | **13** | **19** | **16** | **10** | **19** | **158** |
| 4 | **12** | **18** | **18** | **11** | **21** | **10** | **167** |
| 5 | **11** | **19** | **17** | **8(108)** | **18** | **12** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **150** |
| Потребности | **151** | **0** | **139** | **93** | **103** | **171** |  |

Следующий минимальный элемент 8 в ячейках 5-4. От поставщика 5 к потребителю 4 будем доставлять min = { 108 , 201 } = 108 единиц продукции.

Следующий минимальный элемент 9 в ячейках 3-1. От поставщика 3 к потребителю 1 будем доставлять min = {151 , 158 } = 151 единиц продукции.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | запасы |
| 1 | **20** | **10** | **14** | **11(5)** | **19** | **9(171)** | **0** |
| 2 | **18** | **8(115)** | **16** | **13** | **15** | **21** | **6** |
| 3 | **9(151)** | **13** | **19** | **16** | **10(7)** | **19** | **0** |
| 4 | **12** | **18** | **18** | **11** | **21** | **10** | **167** |
| 5 | **11** | **19** | **17** | **8(108)** | **18** | **12** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **150** |
| Потребности | **0** | **0** | **139** | **88** | **96** | **0** |  |

Следующий минимальный элемент 9 в ячейках 1-6. От поставщика 1 к потребителю 6 будем доставлять min = {176 , 171 } = 171 единиц продукции.

Следующий минимальный элемент 10 в ячейках 3-5. От поставщика 3 к потребителю 5 будем доставлять min = {7 , 103 } = 7 единиц продукции.

Следующий минимальный элемент 11 в ячейках 1-4. От поставщика 1 к потребителю 4 будем доставлять min = {5 , 93 } = 5 единиц продукции.

Следующий минимальный элемент 11 в ячейках 4-4. От поставщика 4 к потребителю 4 будем доставлять min = {167 , 88 } =88 единиц продукции.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | запасы |
| 1 | **20** | **10** | **14** | **11(5)** | **19** | **9(171)** | **0** |
| 2 | **18** | **8(115)** | **16** | **13** | **15(6)** | **21** | **0** |
| 3 | **9(151)** | **13** | **19** | **16** | **10(7)** | **19** | **0** |
| 4 | **12** | **18** | **18(79)** | **11(88)** | **21** | **10** | **0** |
| 5 | **11** | **19** | **17** | **8(108)** | **18** | **12** | **0** |
| 6 | **0** | **0** | **0** | **0** | **0** | **0** | **150** |
| Потребности | **0** | **0** | **60** | **0** | **90** | **0** |  |

Следующий минимальный элемент 15 в ячейках 2-5. От поставщика 2 к потребителю 5 будем доставлять min = {96 , 6 } =6 единиц продукции.

Следующий минимальный элемент 18 в ячейках 4-3. От поставщика 4 к потребителю 3 будем доставлять min = {79, 139 } =79 единиц продукции.

Следующий минимальный элемент 0 в ячейках 6-3. От поставщика 6 к потребителю 3 будем доставлять min = {150, 60 } =60 единиц продукции.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | 1 | 2 | 3 | 4 | 5 | 6 | запасы |
| 1 | **20** | **10** | **14** | **11(5)** | **19** | **9(171)** | **0** |
| 2 | **18** | **8(115)** | **16** | **13** | **15(6)** | **21** | **0** |
| 3 | **9(151)** | **13** | **19** | **16** | **10(7)** | **19** | **0** |
| 4 | **12** | **18** | **18(79)** | **11(88)** | **21** | **10** | **0** |
| 5 | **11** | **19** | **17** | **8(108)** | **18** | **12** | **0** |
| 6 | **0** | **0** | **0(60)** | **0** | **0(90)** | **0** | **0** |
| Потребности | **0** | **0** | **0** | **0** | **0** | **0** |  |

Следующий минимальный элемент 0 в ячейках 6-5. От поставщика 6 к потребителю 5 будем доставлять min = {90, 90} =90 единиц продукции.

Заполненные нами ячейки будем называть базисными, остальные - свободными.

Для решения задачи методом потенциалов, количество базисных ячеек (задействованных маршрутов) должно равняться m + n - 1, где m - количество строк в таблице, n - количество столбцов в таблице (6+6-1=11). Количество базисных ячеек равно 11, а это значит, что мы нашли начальное решение, то есть истратили все запасы поставщиков и удовлетворили все потребности покупателей.

## S = 151\*9 + 115\*8+ 18\*79 + 60\*0 + 108\*8 + 11\*88 + 11\*5 + 15\*6 +9\*171+70+0= 7287 ден. ед.

Общие затраты на доставку всей продукции, для начального решения, составляют **7287** ден. ед. .

*Улучшим наше решение*

Проверим оптимальность опорного плана. Найдем *предварительные потенциалы* ui, vj. по занятым клеткам таблицы, в которых ui + vj = cij, полагая, что u1 = 0.

|  |
| --- |
| u1 + v4 = 11; v4 = 11 u4 + v4 = 11; u4 = 0 u4 + v3 = 18; v3 = 18 u6 + v3 = 0; u6 = -18 u6 + v5 = 0; v5 = 18 u2 + v5 = 15; u2 = -3 u2 + v2 = 8; v2 = 11 u3 + v5 = 10; u3 = -8 u3 + v1 = 9; v1 = 17 u5 + v4 = 8; u5 = -3 u1 + v6 = 9; v6 = 9 |
|  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | B 1 | B 2 | B 3 | B 4 | B 5 | B 6 | U |
| A 1 | **20** | **10** | **14** | **11(5)** | **19** | **9(171)** | **0** |
| A 2 | **18** | **8(115)** | **16** | **13** | **15(6)** | **21** | **-3** |
| A 3 | **9(151)** | **13** | **19** | **16** | **10(7)** | **19** | **-8** |
| A 4 | **12** | **18** | **18(79)** | **11(88)** | **21** | **10** | **0** |
| A 5 | **11** | **19** | **17** | **8(108)** | **18** | **12** | **-3** |
| A 6 | **0** | **0** | **0(60)** | **0** | **0(90)** | **0** | **-18** |
| V | **17** | **11** | **18** | **11** | **18** | **9** | **880** |

Опорный план не является оптимальным, так как существуют оценки свободных клеток, для которых ui + vj > cij  
(1;2): 0 + 11 > 10; ∆12 = 0 + 11 - 10 = 1 > 0  
(1;3): 0 + 18 > 14; ∆13 = 0 + 18 - 14 = 4 > 0  
(4;1): 0 + 17 > 12; ∆41 = 0 + 17 - 12 = 5 > 0  
(5;1): -3 + 17 > 11; ∆51 = -3 + 17 - 11 = 3 > 0  
max(1,4,5,3) = 5

Выбираем максимальную оценку свободной клетки (4;1): 12  
Для этого в перспективную клетку (4;1) поставим знак «+», а в остальных вершинах многоугольника чередующиеся знаки «-», «+», «-».

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | B 1 | B 2 | B 3 | B 4 | B 5 | B 6 | U |
| A 1 | **20** | **10** | **14** | **11(5)** | **19** | **9(171)** | **0** |
| A 2 | **18** | **8(115)** | **16** | **13** | **15(6)** | **21** | **-3** |
| A 3 | **9(151)(-)** | **13** | **19** | **16** | **10(7)(+)** | **19** | **-8** |
| A 4 | **12(+)** | **18** | **18(79)(-)** | **11(88)** | **21** | **10** | **0** |
| A 5 | **11** | **19** | **17** | **8(108)** | **18** | **12** | **-3** |
| A 6 | **0** | **0** | **0(60)**  **(+)** | **0** | **0(90)(-)** | **0** | **-18** |
| V | **17** | **11** | **18** | **11** | **18** | **9** | **880** |

Цикл приведен в таблице (4,1 → 4,3 → 6,3 → 6,5 → 3,5 → 3,1).  
Из грузов хij стоящих в минусовых клетках, выбираем наименьшее, т.е. у = min (4, 3) = 79. Прибавляем 79 к объемам грузов, стоящих в плюсовых клетках и вычитаем 79 из Хij, стоящих в минусовых клетках. В результате получим новый опорный план.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | B 1 | B 2 | B 3 | B 4 | B 5 | B 6 | U |
| A 1 | **20** | **10** | **14** | **11(5)** | **19** | **9(171)** | **0** |
| A 2 | **18** | **8(115)** | **16** | **13** | **15(6)** | **21** | **-3** |
| A 3 | **9(71) (-)** | **13** | **19** | **16** | **10(86) (+)** | **19** | **-8** |
| A 4 | **12(79)(+)** | **18** | **18(0)(-)** | **11(88)** | **21** | **10** | **0** |
| A 5 | **11** | **19** | **17** | **8(108)** | **18** | **12** | **-3** |
| A 6 | **0** | **0** | **0(139)**  **(+)** | **0** | **0(11)(-)** | **0** | **-18** |
| V | **17** | **11** | **18** | **11** | **18** | **9** | **880** |

То есть мы получили новый план

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | B 1 | B 2 | B 3 | B 4 | B 5 | B 6 | запасы |
| A 1 | **20** | **10** | **14** | **11(5)** | **19** | **9(171)** | **176** |
| A 2 | **18** | **8(115)** | **16** | **13** | **15(6)** | **21** | **121** |
| A 3 | **9(72)** | **13** | **19** | **16** | **10 (86)** | **19** | **158** |
| A 4 | **12(79)** | **18** | **18** | **11(88)** | **21** | **10** | **167** |
| A 5 | **11** | **19** | **17** | **8(108)** | **18** | **12** | **108** |
| A 6 | **0** | **0** | **0(139)** | **0** | **0(11)** | **0** | **150** |
| Потребности | **151** | **115** | **139** | **201** | **103** | **171** | **880** |

Проверим оптимальность опорного плана. Найдем *предварительные потенциалы* ui, vj. по занятым клеткам таблицы, в которых ui + vj = cij, полагая, что u1 = 0.

|  |
| --- |
| u1 + v4 = 11; 0 + v4 = 11; v4 = 11 u4 + v4 = 11; 11 + u4 = 11; u4 = 0 u4 + v1 = 12; 0 + v1 = 12; v1 = 12 u3 + v1 = 9; 12 + u3 = 9; u3 = -3 u3 + v5 = 10; -3 + v5 = 10; v5 = 13 u2 + v5 = 15; 13 + u2 = 15; u2 = 2 u2 + v2 = 8; 2 + v2 = 8; v2 = 6 u6 + v5 = 0; 13 + u6 = 0; u6 = -13 u6 + v3 = 0; -13 + v3 = 0; v3 = 13 u5 + v4 = 8; 11 + u5 = 8; u5 = -3 u1 + v6 = 9; 0 + v6 = 9; v6 = 9 |
|  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| потребители  Поставщики | B 1 | B 2 | B 3 | B 4 | B 5 | B 6 | U |
| A 1 | **20** | **10** | **14** | **11(5)** | **19** | **9(171)** | **0** |
| A 2 | **18** | **8(115)** | **16** | **13** | **15(6)** | **21** | **2** |
| A 3 | **9(71) (-)** | **13** | **19** | **16** | **10(86) (+)** | **19** | **-3** |
| A 4 | **12(79)(+)** | **18** | **18(0)(-)** | **11(88)** | **21** | **10** | **0** |
| A 5 | **11** | **19** | **17** | **8(108)** | **18** | **12** | **-3** |
| A 6 | **0** | **0** | **0(139)**  **(+)** | **0** | **0(11)(-)** | **0** | **13** |
| V | **12** | **6** | **13** | **11** | **13** | **9** | **880** |

Опорный план является оптимальным, так все оценки свободных клеток удовлетворяют условию ui + vj ≤ cij.  
Минимальные затраты составят: F(x) = 11\*5 + 9\*171 + 8\*115 + 15\*6 + 9\*72 + 10\*86 + 12\*79 + 11\*88 + 8\*108 + 0\*139 + 0\*11 = 6892