Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"

профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

**Лабораторная работа №3  
 по дисциплине  
 Алгоритмы и вычислительные методы оптимизации**

Бригада №3

Выполнил:

студент гр. ИП-012 Николаев А.Д. \_

ФИО студента

«13» февраля 2023 г.

Новосибирск 2023 г.

**Оглавление**

[Постановка задачи 3](https://d.docs.live.net/546c55b31644a344/Рабочий%20стол/Лабы%20по%20АВМО%20(Kotlin)/Отчёт2.docx#_Toc127393577)

[Примеры работы программы 4](https://d.docs.live.net/546c55b31644a344/Рабочий%20стол/Лабы%20по%20АВМО%20(Kotlin)/Отчёт2.docx#_Toc127393578)

[Исходный код программы 6](https://d.docs.live.net/546c55b31644a344/Рабочий%20стол/Лабы%20по%20АВМО%20(Kotlin)/Отчёт2.docx#_Toc127393579)

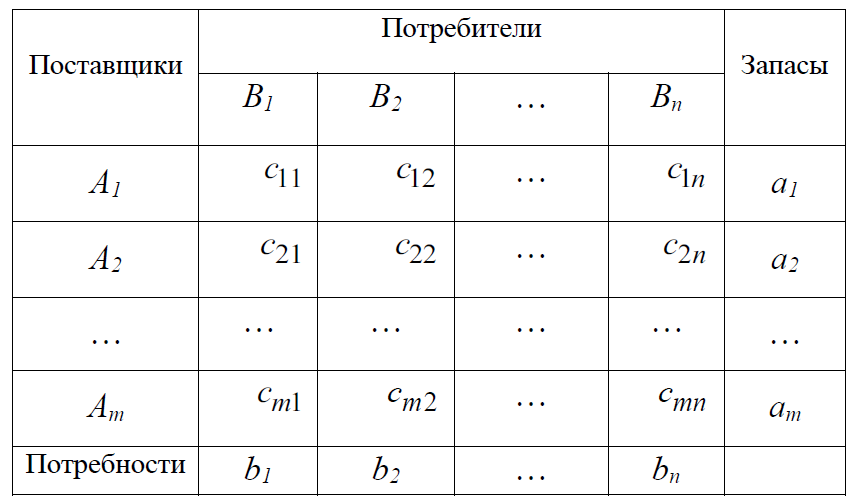
# **Постановка задачи**

Написать программу, находящую начальный опорный план транспортной задачи одним из указанных методов:

1. Метод северо-западного угла.
2. Метод минимальной стоимости.
3. Метод Фогеля.

Номер задания определяется по формуле: N mod 3 = X, где N – номер бригады, а X – номер метода. (3 mod 3 = 0. Метод северо-западного угла). В комментариях кода программы можно дополнительно реализовать метод минимальной стоимости и метод потенциалов.

На вход программе подаются начальные данные распределительной таблицы: тарифы перевозок, запасы поставщиков и потребности потребителей (данные считываются из файла в виде матрицы размера (m+1)X(n+1)):



# **Примеры работы программы**

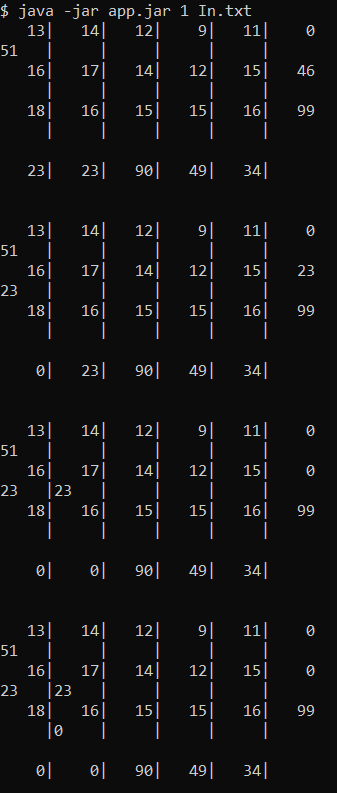
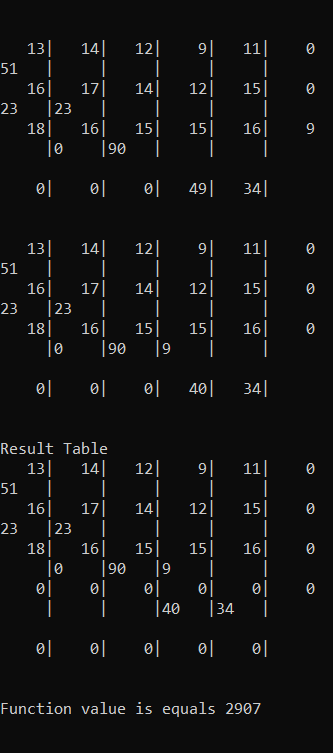
 

Рис. 1 Пример работы программы с заданием 1.б занятия №10

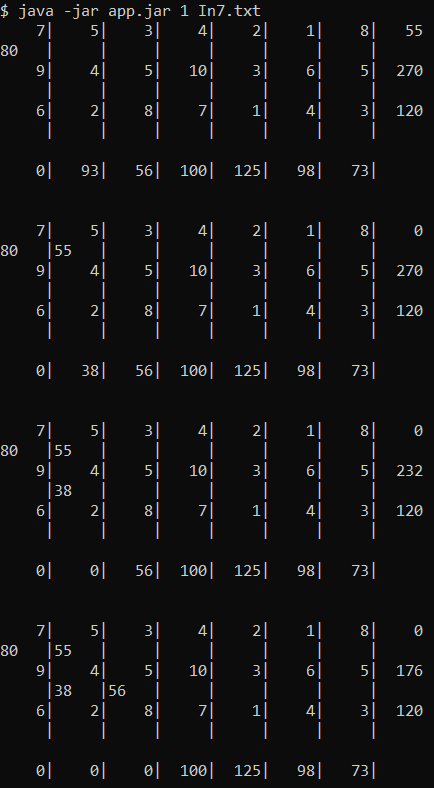
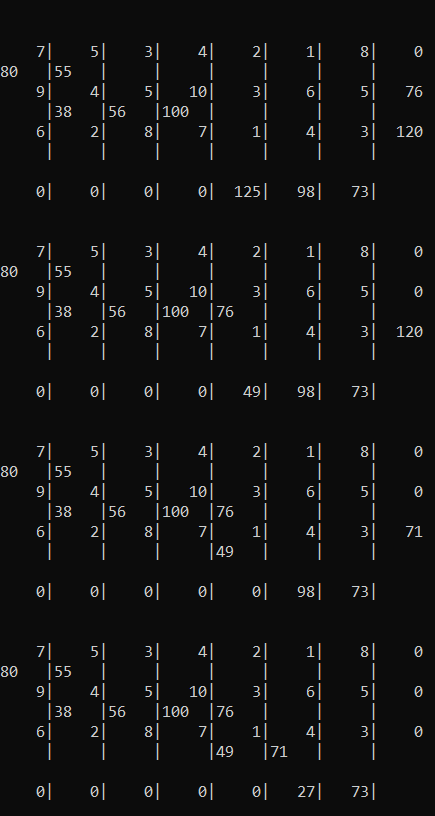
 

Рис. 2 Пример работы программы с заданием 2.ж занятия №10 без результата

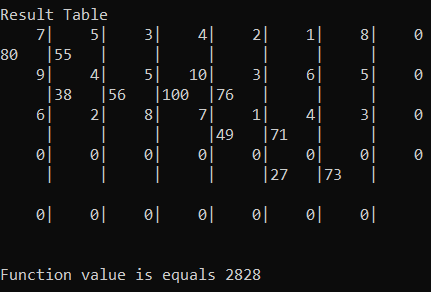


Рис. 3 Пример работы программы с заданием 2.ж занятия №10 с результатом

# **Исходный код программы**

import java.io.File

import kotlin.math.abs

import java.util.Scanner

typealias MutMatr<T> = MutableList< MutableList<T> >

typealias MutVect<T> = MutableList<T>

fun Min(x:Int, y:Int): Int

{

return if(x<y) x else y

}

class Index(val x: Int, val i: Int, val j: Int){}

fun printTable(A:MutVect<Int>, B:MutVect<Int>, C: MutMatr<Int>, X: MutMatr<Int?>)

{

val n=A.size

val m=B.size

for(i in 0 until n)

{

for(j in 0 until m) print(String.format("%5d|",C[i][j]))

println(String.format("%5d",A[i]))

for(j in 0 until m)

{

if(X[i][j]!=null) print(String.format("%-5d|",X[i][j]))

else print(" |")

}

println("")

}

println("")

for(j in 0 until m) print(String.format("%5d|",B[j]))

println("\n\n")

}

fun printTablePotential(A:MutVect<Int>, B:MutVect<Int>, C: MutMatr<Int>, X: MutMatr<Int?>, Rows: MutVect<Int?>, Columns: MutVect<Int?>)

{

val n=A.size

val m=B.size

for(i in 0 until n)

{

for(j in 0 until m) print(String.format("%5d|",C[i][j]))

print(String.format("%5d ",A[i]))

println("(${Rows[i]!!})")

for(j in 0 until m)

{

if(X[i][j]!=null) print(String.format("%-5d|",X[i][j]))

else print(String.format("%5s|","(${C[i][j]-Rows[i]!!-Columns[j]!!})"))

}

println("")

}

println("")

for(j in 0 until m) print(String.format("%5d|",B[j]))

println("")

for(j in 0 until m) print(String.format("%6s",String.format("(%d)",Columns[j]!!)))

println("\n\n")

}

fun printValue(A:MutVect<Int>, B:MutVect<Int>, C: MutMatr<Int>, X: MutMatr<Int?>)

{

val n=A.size

val m=B.size

var sum=0L

for(i in 0 until n) for(j in 0 until m) if(X[i][j]!=null) sum+=X[i][j]!!.toLong()\*C[i][j].toLong()

println("Function value is equals $sum\n")

}

fun countNotNull(X: MutMatr<Int?>,n: Int, m: Int): Int

{

var res=0

for(i in 0 until n) for(j in 0 until m) if(X[i][j]!=null) res++

return res

}

fun northWestAngle(A:MutVect<Int>, B:MutVect<Int>, C: MutMatr<Int>, X: MutMatr<Int?>)

{

var i=0

var j=0

val n=A.size

val m=B.size

val count=n+m

for(k in 1 until count)

{

if(i==n || j==m) break

val min=Min(A[i],B[j])

A[i]-=min

B[j]-=min

X[i][j]=min

if(A[i]==0) i++ else j++

printTable(A,B,C,X)

}

}

/\* Метод минимальной стоимости

fun minCast(A:MutVect<Int>, B:MutVect<Int>, C: MutMatr<Int>, X: MutMatr<Int?>)

{

val n=A.size

val m=B.size

var i:Int

var j:Int

var min:Int

var AvailableRow=(0 until n).toMutableList()

var AvailableColumn=(0 until m).toMutableList()

var SortArray=mutableListOf<Index>()

for(i in 0 until n) for(j in 0 until m) SortArray.add(Index(C[i][j],i,j))

SortArray.sortBy{it.x}

for(ind in SortArray)

{

i=ind.i

j=ind.j

if((i in AvailableRow) && (j in AvailableColumn))

{

min=Min(A[i],B[j])

A[i]-=min

B[j]-=min

X[i][j]=min

if(A[i]==0) AvailableRow.remove(i)

else AvailableColumn.remove(j)

printTable(A,B,C,X)

}

}

}

\*/

/\* Проверка оптимальности решения и метод потенциалов

fun checkSolution(A:MutVect<Int>, B:MutVect<Int>, C: MutMatr<Int>, X: MutMatr<Int?>): Int?

{

val n=A.size

val m=B.size

var count=n+m-1

var switch=0

var rowInQueue=mutableListOf<Int>()

var columnInQueue=mutableListOf<Int>()

var rowPotentials=MutableList<Int?>(n, {null})

var columnPotentials=MutableList<Int?>(m, {null})

rowPotentials[0]=0

rowInQueue.add(0)

while(count>0)

{

if(switch==0)

{

for(i in rowInQueue) for(j in 0 until m) if(X[i][j]!=null && columnPotentials[j]==null)

{

columnPotentials[j]=C[i][j]-rowPotentials[i]!!.toInt()

count--

columnInQueue.add(j)

}

rowInQueue.clear()

}

else

{

for(j in columnInQueue) for(i in 0 until n) if(X[i][j]!=null && rowPotentials[i]==null)

{

rowPotentials[i]=C[i][j]-columnPotentials[j]!!.toInt()

count--

rowInQueue.add(i)

}

columnInQueue.clear()

}

switch=(1-switch)

}

printTablePotential(A,B,C,X,rowPotentials,columnPotentials)

var min=Index(C[0][0]-(rowPotentials[0]!!.toInt()+columnPotentials[0]!!.toInt()),0,0)

var temp: Index

for(i in 0 until n) for(j in 0 until m)

{

temp=Index(C[i][j]-(rowPotentials[i]!!.toInt()+columnPotentials[j]!!.toInt()),i,j)

if(min.x>temp.x) min=temp

}

if(min.x<0) return min.i\*m+min.j

return null

}

fun newBasisVariable(A:MutVect<Int>, B:MutVect<Int>, C: MutMatr<Int>, X: MutMatr<Int?>, \_i: Int, \_j: Int)

{

val n=A.size

val m=B.size

var VertexList=mutableMapOf<Int, Int?>()

var VertEdgeMapList=mutableMapOf<Int, MutVect<Int> >()

var GorEdgeMapList=mutableMapOf<Int, MutVect<Int> >()

var switch: Int

var VertexQueue=mutableListOf<Int>()

var VertexQueueNext=mutableListOf<Int>()

var flag=true

X[\_i][\_j]=0

for(i in 0 until n) for(j in 0 until m) if(X[i][j]!=null)

{

VertexList.put(i\*m+j, null)

VertEdgeMapList.put(i\*m+j, mutableListOf<Int>())

GorEdgeMapList.put(i\*m+j, mutableListOf<Int>())

}

for(k in VertexList.keys) for(i in 0 until n)

{

for(i in 0 until n)

{

if(i==k/m) continue

if(X[i][k%m]!=null) VertEdgeMapList[k]!!.add(i\*m+k%m)

}

for(j in 0 until m)

{

if(j==k%m) continue

if(X[k/m][j]!=null) GorEdgeMapList[k]!!.add(k-k%m+j)

}

}

//Место для поиска в глубину

switch=0

var \_ind=\_i\*m+\_j

VertexList[\_ind]=-1

VertexQueue.add(\_ind)

while(flag)

{

for(vert in VertexQueue)

{

if(switch==0)

{

for(ind in (GorEdgeMapList[vert]!!))

{

if(VertexList[ind]==-1)

{

VertexList[ind]=vert

flag=false

}

VertexList[ind]=vert

VertexQueueNext.add(ind)

}

}

else

{

for(ind in (VertEdgeMapList[vert]!!))

{

if(VertexList[ind]==-1)

{

VertexList[ind]=vert

flag=false

}

VertexList[ind]=vert

VertexQueueNext.add(ind)

}

}

}

switch=1-switch

VertexQueue.clear()

VertexQueue.addAll(VertexQueueNext)

VertexQueueNext.clear()

}

if(switch==1)

{

println("What the...")

return

}

switch=0

var vertNow=\_ind

var vertNext=VertexList[\_ind]!!

var vertMin=vertNext

print("(${vertNow/m},${vertNow%m})")

while(true)

{

if(switch==1 && (X[vertNow/m][vertNow%m]!!<X[vertMin/m][vertMin%m]!!)) vertMin=vertNow

vertNow=vertNext

vertNext=VertexList[vertNow]!!

switch=(1-switch)

print(" -> (${vertNow/m},${vertNow%m})")

if(vertNow==\_ind) break

}

println("\n")

switch=0

vertNext=VertexList[\_ind]!!

var minValue=X[vertMin/m][vertMin%m]!!

while(true)

{

if(switch==0) X[vertNow/m][vertNow%m]=X[vertNow/m][vertNow%m]!!+minValue

else X[vertNow/m][vertNow%m]=X[vertNow/m][vertNow%m]!!-minValue

vertNow=vertNext

vertNext=VertexList[vertNow]!!

switch=(1-switch)

if(vertNow==\_ind) break

}

X[vertMin/m][vertMin%m]=null

}

fun findBestSolution(A:MutVect<Int>, B:MutVect<Int>, C: MutMatr<Int>, X: MutMatr<Int?>)

{

var position: Int?=checkSolution(A,B,C,X)

var i: Int

var j: Int

val n=A.size

val m=B.size

while(position != null)

{

i=position.toInt()/m

j=position.toInt()%m

newBasisVariable(A,B,C,X,i,j)

printTable(A,B,C,X)

printValue(A,B,C,X)

position=checkSolution(A,B,C,X)

}

}

\*/

fun main(args: Array<String>){

val func=if(args.size>0) 1 else 1

val fileName=if(args.size>1) args[1] else "In.txt"

val scan=Scanner(File(fileName))

val n=scan.nextInt()

val m=scan.nextInt()

var Matr\_C=mutableListOf< MutableList<Int> >()

var Matr\_X=mutableListOf< MutableList<Int?> >()

var Vect\_A=mutableListOf<Int>()

var Vect\_B=mutableListOf<Int>()

var sum\_A=0

var sum\_B=0

for(i in 0 until n)

{

Matr\_C.add(mutableListOf<Int>())

Matr\_X.add(mutableListOf<Int?>())

for(j in 0 until m)

{

Matr\_C[i].add(scan.nextInt())

Matr\_X[i].add(null)

}

Vect\_A.add(scan.nextInt())

sum\_A+=Vect\_A[i]

}

for(j in 0 until m)

{

Vect\_B.add(scan.nextInt())

sum\_B+=Vect\_B[j]

}

when(func)

{

1 -> northWestAngle(Vect\_A,Vect\_B,Matr\_C,Matr\_X)

//2 -> minCast(Vect\_A,Vect\_B,Matr\_C,Matr\_X)

}

var count: Int

if(sum\_A<sum\_B)

{

Vect\_A.add(sum\_B-sum\_A)

val i0=Vect\_A.size-1

Matr\_C.add(mutableListOf<Int>())

Matr\_X.add(mutableListOf<Int?>())

for(j in 0 until m)

{

Matr\_C[i0].add(0)

Matr\_X[i0].add(if(Vect\_B[j]!=0) Vect\_B[j] else null)

Vect\_A[i0]-=Vect\_B[j]

Vect\_B[j]=0

}

count=countNotNull(Matr\_X,n+1,m)

for(j in 0 until m)

{

if(count==(n+m)) break

if(Matr\_X[i0][j]==null)

{

Matr\_X[i0][j]=0

count++

}

}

}

else if(sum\_A>sum\_B)

{

Vect\_B.add(sum\_A-sum\_B)

val j0=Vect\_B.size-1

for(i in 0 until n)

{

Matr\_C[i].add(0)

Matr\_X[i].add(if(Vect\_A[i]!=0) Vect\_A[i] else null)

Vect\_B[Vect\_B.size-1]-=Vect\_A[i]

Vect\_A[i]=0

}

count=countNotNull(Matr\_X,n,m+1)

for(i in 0 until n)

{

if(count==(n+m)) break

if(Matr\_X[i][j0]==null)

{

Matr\_X[i][j0]=0

count++

}

}

}

println("Result Table")

printTable(Vect\_A,Vect\_B,Matr\_C,Matr\_X)

printValue(Vect\_A,Vect\_B,Matr\_C,Matr\_X)

//findBestSolution(Vect\_A,Vect\_B,Matr\_C,Matr\_X)

}