Курсов проект по Системен Анализ

Изработен от Николай Иванов Ф-нм. 17621301

Тема: решение на транспортната задача по Метода на минималния елемент

Задание за курсов проект 20

по дисциплината „Системен анализ“
на студента: Николай Иванов, ф.н. 17621301

Задача: Да се състави програма за получаване на началното решение на транспортната задача по Метода на минималния елемент със следните входни данни:

	Склад 1	Склад 2	Склад3	Капацитет
Завод1	34	25	15	1800
Завод2	23	15	9	1200
Завод3	9	34	15	1500
Капацитет	2200	1300	1000	

	Склад1	Склад2	Капацитет
Завод1	2	4	2300
Завод2	5	3	1900
Завод3	3	2	2600
Капацитет	4000	2300	

Изискванията към описанието на проекта:

1. Записката да съдържа файл на MS Word и презентация за защита на проекта.
2. Файлът да съдържа:
   - заглавна страница;
   - заданието;
   - теоретична част: постановка на транспортната задача, каноничен вид, метод на минималния елемент за намиране на началното решение, метод на потенциалите. Теорията да се илюстрира с подходящ пример;
   - практическа част: описание на алгоритъма чрез формули и блокова схема (да се използват функционалностите на MS Word), входните данни да се прочитат от файл или таблица, изходните резултати да се записват във файл или таблица;
   - изводи и заключения;
   - използвана литература;
   - приложение – сорсът на програмата

3. Дата на предаване и защита не по-късно от 18.12.2019 г.

Метода на минималния елемент

При всяка стъпка се удовлетворява транспорта на количество от клетката с минимална цена за единица превоз. Запълването на таблицата с началните стойности започва от клетката с минимална тарифа. В нея се записва по-малката от стойностите на производителя или консуматора. От разглеждане се изключва реда или колоната, чиито количества са напълно изчерпани. След това от оставащите клетки отново се избира тази с най-малката тарифа и процесът продължава до разпределяне на всички количества.

Пример.

	Склад 1	Склад 2	Склад 3	капацитет
Завод 1	34	25	15	1800
Завод 2	23	15	9	1200
Завод 3	9	34	15	1500
капацитет	2200	1300	1000	

Стъпка - 1

	Склад 1	Склад 2	Склад 3	капацитет
Завод 1	34	25	15	1800
Завод 2	23	15	9	1200
Завод 3	9 1500	0	0	0
капацитет	700	1300	1000	

Стъпка - 2

	Склад 1	Склад 2	Склад 3	капацитет
Завод 1	34	25	0	1800
Завод 2	23	15	9 1000	200
Завод 3	9 1500	0	0	0
капацитет	700	1300	0	

	Склад 1	Склад 2	Склад 3	капацитет
Завод 1	34	25	0	1800
Завод 2	0	15 200	9 1000	0
Завод 3	9 1500	0	0	0
капацитет	700	1100	0	

Стъпка - 4

	Склад 1	Склад 2	Склад 3	капацитет
Завод 1	34	25 1100	0	700
Завод 2	23	15 200	9 1000	0
Завод 3	9 1500	0	0	0
капацитет	700	0	0	

Стъпка - 5

	Склад 1	Склад 2	Склад 3	капацитет
Завод 1	34 700	25 1100	0	0
Завод 2	0	15 200	9 1000	0
Завод 3	9 1500	0	0	0
капацитет	0	0	0	

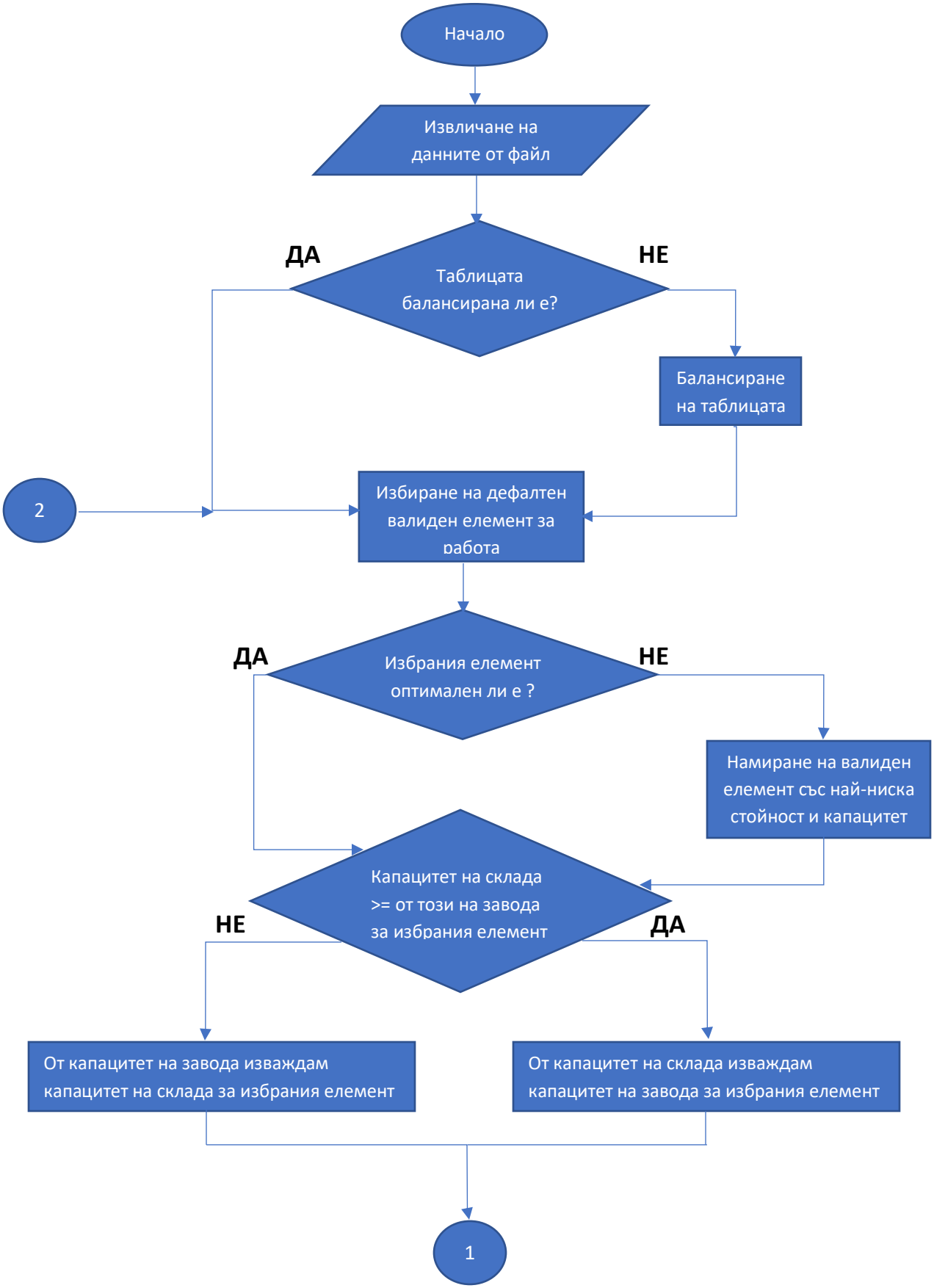
Zmin = 34 * 700 + 25 *1100 + 15 *200 + 9 * 1000 + 9 * 1500

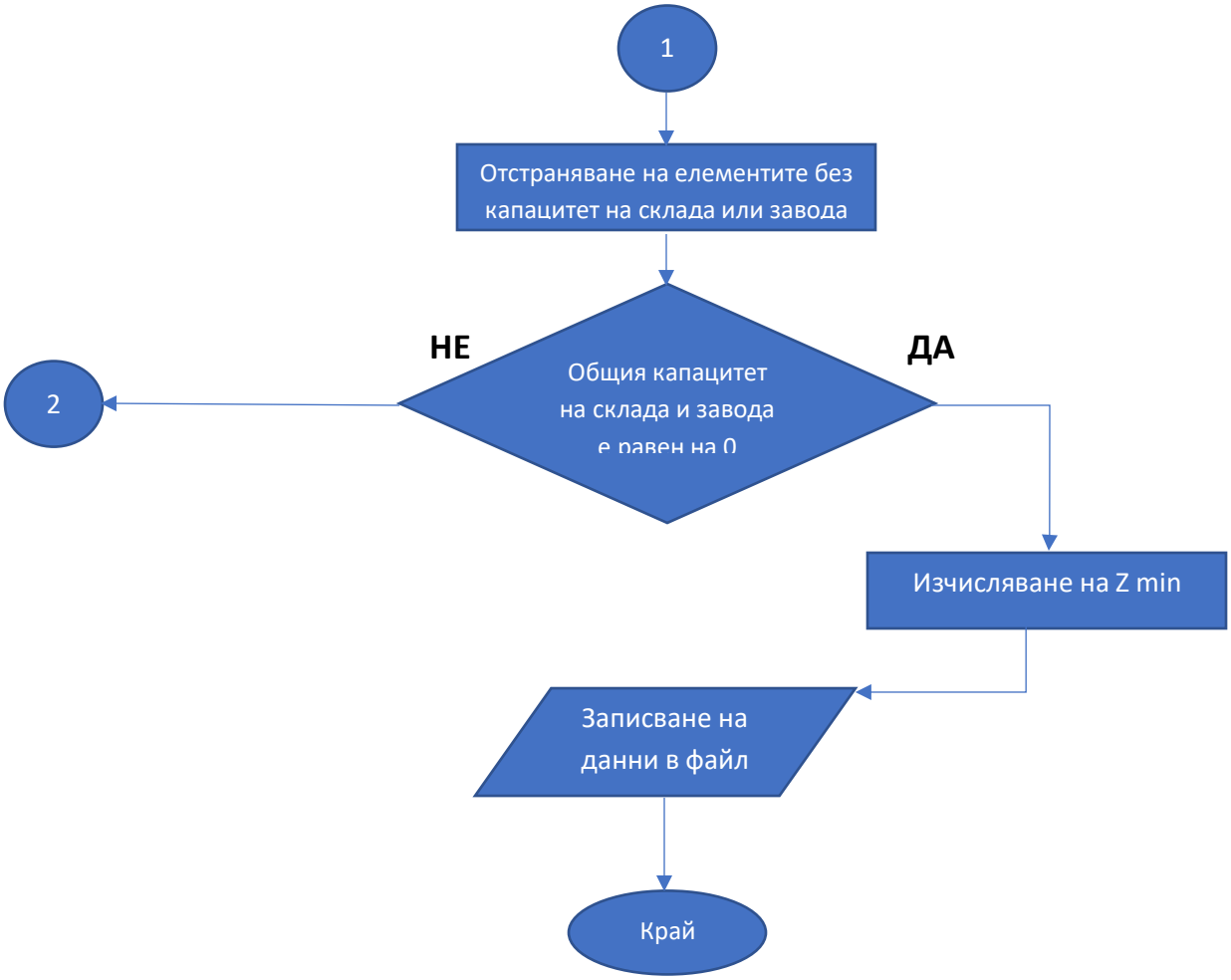
Zmin = 76800

Алгоритъма за постигане на този резултат е със сложност:

F(n) = O(n)

Блок схема


Начало
Извличане на данните от файл
Таблицата балансирана ли е?
ДА
НЕ
Балансиране на таблицата
2
Избиране на дефалтен валиден елемент за работа
Избрания елемент оптимален ли е ?
ДА
НЕ
Намиране на валиден елемент със най-ниска стойност и капацитет
Капацитет на склада >= от този на завода за избрания елемент
НЕ
ДА
От капацитет на завода изваждам капацитет на склада за избрания елемент
От капацитет на склада изваждам капацитет на завода за избрания елемент
1



Използвана литература:

Системен анализ – Ръководство за лабораторни упражнения
Н. Николов, Д. Генов, А. Иванов, М. Тодорова

Програмата е написана на Python 3.7
Редовете в зелено са коментари
Сорс код:
GUI.py – Графичен интерфейс

```
from tkinter import * # импортирам библиотеката tkinter за създаване на графичен интерфайс
from tkinter.filedialog import askopenfilename
import balkp as BA #импортирам бакенда на програмата
def searchfile(): #функцията seravhfile си използва за търсене на файлове когато бутон 1 е актевиран
   window.fileName = askopenfilename(filetypes = ( ("textfile" ,".txt"), ("All files","*") ) )
   e1.insert(0, str(window.fileName)) # записвам адреса на файла в текстовата кутия
   l1['text']="File selected" #променям индекатора за наличност на файл
   l1['fg']="Green"

def start(): #тази функция стартира бакенда на програмата
   if str(window.fileName)!= "": # проверка дали има избран файл
      BA.fun(window.fileName) #викане на бакенда
   else:
      l1['text']="Chose file"

window=Tk()#създавам tk обект за графичен интерфайс
window.title(" Least cost method problem solver")
window.resizable(width=False, height=False)

#бутон за избране на файл който повиква searchfile()
b1=Button(window,text="Chose file", width=16,command=searchfile)
b1.grid(row=1, column=1)

#бутон за стартиране бакенда на програмата start()
b2=Button(window,text="Generate solution", width=16,command=start)
b2.grid(row=2, column=1)

textVar="Please select a file" #индикатор за валидност
l1=Label(text = textVar , fg='Red' ,bd=5)
l1.grid(row=2, column=2)
e1=Entry(window,width=40) # entry за въвеждане на дайл
e1.grid(row=1, column=2)
window.mainloop()
```

balkp.py – бакенд на програмата

```
from datetime import datetime  #импортирам библиотека datetime за генериране на фиме за изходен файл

def fun(Filename): # главната фукция за рещаване на задача по метода на минималния елемент
########################Файлов OUTPUT
  # стздавам временни променливи Bdemand, Bsupply и lista за обработка на соровите данни от тесктовия дойумент
  Bdemand=[]
  Bsupply=[]
  lista=[]
  with open(Filename, 'r') as file: # отварчм текстов документ чиито име местоположение е записано в променливата Filename под формата на string
    index_counter=0 # боряч който следи на кой ред се намира filepointera
    for i in file.readlines(): # визима 1 необработен ред от файла и го разделя на индивидуални елементи, който после вкарва в съответните листи
      if index_counter == 0: #ако file pointer е в позиция 0 то данните ще влязат в Bdemand
        Bdemand=i.split()

      elif index_counter == 1: #ако file pointer е в позиция 1 то данните ще влязат в Bsupply
        Bsupply=i.split()

      elif index_counter>1: #ако file pointer е в позиция по голяма от 1 то данните ще влязат в Bsupply
        lista.append(i.split())
      index_counter+=1
  # demand и supply са трайните листове които държат обработините данни от текстовия документ
  demand=[]
  supply=[]
  for i in Bdemand: # вкарвам уформените от demand
    demand.append(int(i))

  for i in Bsupply: # вкарвам уформените от supply
    supply.append(int(i))
```

```
   S1=[] # В S1,S2 и S3 се съхраняват редовете с транспортните цени
   S2=[]
   S3=[]
   unBalList=[] # unBalList съдържа обработините цени на транспорта
   for i in lista:# В този фор цикъл се обработват данните от листа и се вкарват ред по ред S1,S2 и S3, които после влизат в unBalList
      for j in i:
         if lista.index(i) == 0:
            S1.append([int(j),True])
         elif lista.index(i) == 1:
            S2.append([int(j),True])
         elif lista.index(i) == 2:
            S3.append([int(j),True])
   unBalList.append(S1) # вакарвам вече уформените данни в финалния лист
   unBalList.append(S2)
   unBalList.append(S3)
         # Тук преключва файловата чат от програмата

########################Блансиране на листа ако вече не е балансиран
   if int(len(demand)) < int(len(supply) ): #ако в supply има повече елементи от demand, листа е не балансиран
      for i in unBalList: #добавям празен елемент с цел баланс
         i.append([0, True])

      demand.append(sum(supply)-sum(demand))
########################избиране на елемент за работа
   while sum(supply)+sum(demand) != 0: #този цикъл се повтаря докато ресурсите в supply и demand не се изразходят
      chosen_element=[] # Този лист запазва стойноста на избрания елемент
      chosen_element_demand=0 #запазвам demand стойноста отговаряща за избрания елемент
      chosen_element_supply_index=0 #запазвам индейса на supply отговарящ за избрания елемент
      chosen_element_demand_index=0 #запазвам индейса на demand отговарящ за избрания елемент

      exitflag=False # ако този флаг е True значе вече имаме избран елемент,този флаг се ресетва всяка иерация на цикъла
```

```
    for i in unBalList: # С този двумерен фор цикъл се избира валиден дефалтен елемент за работа
        if exitflag == True: # ако имаме избран елемент излизаме от цикъла
            break
        first_index=unBalList.index(i) #брои първия индекс за еленент
        for j in i:
            second_index=i.index(j) #брои врория индекс за еленент
            if j[1]==True: # ако елемента е валиден/свободен ибва избран за дефалтен работен елемент, независимо дъли е оптимален
                exitflag = True #флага става True и следващата иерация на цикъла излизаме от него
                chosen_element=unBalList[first_index][second_index]
                chosen_element_demand=demand[second_index]
                chosen_element_supply_index=first_index
                chosen_element_demand_index=second_index
                break

    for i in unBalList: # В този двумерен цикъл се избира оптималния елемент за работа ако вече не е избран от предходната съпка
        first_index=unBalList.index(i) #брои първия индекс за еленент
        for j in i:
            second_index=i.index(j) #брои врория индекс за еленент
            if j[1] == True and j[0] <=chosen_element[0]: #ако елемента е свободен/валиден и стойноста на транспорта е по малка от тази на вече избрания елемент влизам продължавам напред
                if j[0] == chosen_element[0]: # ако елементите имат еднакви транспортни стойности се избира този със по-големия demand
                    if demand[second_index] > chosen_element_demand:

chosen_element=chosen_element=unBalList[first_index][second_index]
                        chosen_element_demand=demand[second_index]
                        chosen_element_supply_index=first_index
                        chosen_element_demand_index=second_index
                    continue # пропускам тази итерация на цикъла
                chosen_element=chosen_element=unBalList[first_index][second_index]
                chosen_element_demand=demand[second_index]
                chosen_element_supply_index=first_index
                chosen_element_demand_index=second_index
```

```
    # Избирам коя е по голямата стойност измежду supply и demand ,и от нея изваждам по-малката
    If supply[chosen_element_supply_index] <= demand[chosen_element_demand_index]: # ако supply е по малка ще я извадя от demand и ще я зануля
        demand[chosen_element_demand_index]-=supply[chosen_element_supply_index] # изваждам

unBalList[chosen_element_supply_index][chosen_element_demand_index][1]=supply[chosen_element_supply_index] # замествам индекатора за свободност на клетката с количеството ресурси което ще се транспортира
        supply[chosen_element_supply_index]=0 #занулявам

        for i in unBalList: #прави елементите, чийто ресурси са изчерпани, невалидни
            first_index=unBalList.index(i) #брои първия индекс за еленент
            for j in i:
                second_index=i.index(j) #брои второя индекс за еленент
                if j[1] == True and first_index==chosen_element_supply_index: #проверка по елемент
                    j[1]=False

    elif supply[chosen_element_supply_index] > demand[chosen_element_demand_index]: # ако demand е по малка ще я извадя от supply и ще я зануля
        supply[chosen_element_supply_index]-=demand[chosen_element_demand_index] # изваждам

unBalList[chosen_element_supply_index][chosen_element_demand_index][1]=demand[chosen_element_demand_index] # замествам индекатора за свободност на клетката с количеството ресурси което ще се транспортира
        demand[chosen_element_demand_index]=0 #занулявам

        for i in unBalList: #прави елементите, чийто ресурси са изчерпани, невалидни
            first_index=unBalList.index(i) #брои първия индекс за еленент
            for j in i:
                second_index=i.index(j) #брои второя индекс за еленент
```

```
            if j[1] == True and second_index==chosen_element_demand_index: #проверка по елемент
                j[1]=False

    #for i in unBalList : #изкарва таблицата на конзола с цел дебъгване , само трябва да се откоментира кода
    #    print(i,end='')
    #    print("      "+str(supply[unBalList.index(i)]))
    #print(demand)

    if sum(demand) + sum(supply)  == 0: #ако ресурсите са изчерпани и програмата е на проследната си итерация ще се пресметне цената 'Z'
        Z=0 # минималната цена
        for i in unBalList: # иерира през всички елементи на лста
            for j in i:
                if j[1] !=False and j[1] !=0: # ако в летката има ресурси, към Z ще се добави сбора на цената на транспорта и ресурса
                    Z+=(j[1]*j[0])
        #print(Z) #изкарва минималната цена на конзола с цел дебъгване , само трябва да се откоментира кода
######################Файлов INPUT
        with open("minicost - "+str(datetime.now().strftime("%H-%M-%S"))+".txt", 'w') as file: #създавам фал със автоматично генериращо име (сегашната дата и време)
            file.write(str(Z)) # записвам минималната цена в файла
            for i in unBalList:
                file.write("\n") # записвам нов ред
                for j in i:
                    if(j[1]==False): # записвам цните на транспорта и ресурсите в файл
                        file.write(" ("+str(j[0])+" 0)")
                    else:
                        file.write(" ("+str(j[0])+" "+str(j[1])+")")
```