Тверской Государственный технический университет

Кафедра: Программного обеспечения и вычислительной

техники

Лабораторная работа № 6

Выполнил: Студент второго курса

Группы Б.ПИН.РИС 18.05

Федотов Ярослав Всеволодович

Тверь 2019

**Постановка задачи 1**

Самолет, находящийся на высоте H 0 и имеющий скорость F 0, должен

подняться на заданную высоту H кон, набрав при этом заданную скорость V кон.

Известен расход горючего, требуемый для подъема самолета с любой высоты H 1 на

любую другую H 2 (H 2>Н 1) при неизменной скорости V ; известен также расход

горючего, требуемый для увеличения скорости от любого значения V 1 до любого

другого значения V 2 (V 2>V 1) при неизменной высоте H .

Требуется найти оптимальный режим набора высоты и скорости, при

котором расход горючего будет минимальным.

**Алгоритм решения задачи 1**

Импортируем numpy as np

n = 6

v0 = 0

h0 = 0

fuel0 = 0

fuelsmash = 0

fuelminus0smash = 0

typetask = 0

inf = 1000000

m = np.array([[inf,1,inf,3,3,inf], [inf,inf,6,inf,inf,inf],[inf,2,inf,inf,inf,9],[inf,inf,inf,4,7,inf],[inf,8,inf,inf,inf,4],[inf,inf,inf,inf,inf,inf]])

Функция inputdataair():

h0 = int(input("введите начальную высоту самолёта: "))

v0 = int(input("введите начальную скорость самолёта: "))

fuel0 = int(input("введите начальное количество топлива самолёта: "))

fuelsmash = int(input("введите конечное количество топлива самолёта: "))

typetask = int(input("задайте тип решения задачи (1/2): "))

print("ну что полетели!"+'\n'+"карта пути:")

print(m)

print('кратчайший путь по карте пути:')

dijkstra(v0,h0,fuel0,fuelsmash,fuelminus0smash,typetask)

Функция dijkstra(v0,h0,fuel0,fuelsmash,fuelminus0smash,typetask):

v0default = v0

h0default = h0

fueldefault = fuel0

distance = []

visited = []

s = getvertex()

if(s == -1):

вернуть

Для i in range(0,n):

distance.append(inf)

visited.append(false)

distance[s] = 0

mind = int()

while(mind < inf):

mind = inf

minv = int(-1)

Для i in range(0,n):

if(distance[i] < mind and visited[i] == false):

mind = distance[i]

minv = i

h0 = h0default

v0 = v0default

fuel0 = fueldefault

if(minv == -1):

выйти

Для i in range(0,n):

if(m[minv,i] < inf and visited[i] == false):

distance[i] = min(distance[i],distance[minv] + m[minv,i])

visited[minv] = истина

Для i in range(0,n):

if(distance[i] > 0 and distance[i] < inf):

t = i

r = ""

while(t != s):

Для j in range(0,n):

if(m[j,t] < inf and distance[j] == distance[t] - m[j,t]):

t = j

r = chr(ord('a') + t) + "-" + r

выйти

print(r + chr(ord('a')+i))

h0 = h0 + 1

if(typetask == 2):

v0 = v0 + 1

fuel0 = fuel0 - 1

fuelminus0smash = fuel0 - fuelsmash

print("текущая высота: " + str(h0))

print("текущая скорость: " + str(v0))

print("количество бензина: " + str(fuel0))

print("остаток бензина до цели:" + str(fuelminus0smash))

backing = истина

Функция getvertex():

errorconst = -1

v = ' '

maxletter = ord('a') + n - 1

m = chr(maxletter)

v = 'a'

if(ord(v) > ord(m)):

вернуть errorconst

else:

вернуть ord(v) - ord('a')

inputdataair()

**Программа для задачи 1**

import numpy as np

N = 6

V0 = 0

H0 = 0

Fuel0 = 0

FuelSmash = 0

FuelMinus0Smash = 0

TypeTask = 0

INF = 1000000

M = np.array([[INF,1,INF,3,3,INF], [INF,INF,6,INF,INF,INF],[INF,2,INF,INF,INF,9],[INF,INF,INF,4,7,INF],[INF,8,INF,INF,INF,4],[INF,INF,INF,INF,INF,INF]])

def inputDataAir():

H0 = int(input("Введите начальную высоту самолёта: "))

V0 = int(input("Введите начальную скорость самолёта: "))

Fuel0 = int(input("Введите начальное количество топлива самолёта: "))

FuelSmash = int(input("Введите конечное количество топлива самолёта: "))

TypeTask = int(input("Задайте тип решения задачи (1/2): "))

print("Ну что полетели!"+'\n'+"Карта пути:")

print(M)

print('КРАТЧАЙШИЙ ПУТЬ ПО КАРТЕ ПУТИ:')

Dijkstra(V0,H0,Fuel0,FuelSmash,FuelMinus0Smash,TypeTask)

def Dijkstra(V0,H0,Fuel0,FuelSmash,FuelMinus0Smash,TypeTask):

V0Default = V0

H0Default = H0

FuelDefault = Fuel0

Distance = []

Visited = []

S = GetVertex()

if(S == -1):

return

for i in range(0,N):

Distance.append(INF)

Visited.append(False)

Distance[S] = 0

MinD = int()

while(MinD < INF):

MinD = INF

MinV = int(-1)

for i in range(0,N):

if(Distance[i] < MinD and Visited[i] == False):

MinD = Distance[i]

MinV = i

H0 = H0Default

V0 = V0Default

Fuel0 = FuelDefault

if(MinV == -1):

break

for i in range(0,N):

if(M[MinV,i] < INF and Visited[i] == False):

Distance[i] = min(Distance[i],Distance[MinV] + M[MinV,i])

Visited[MinV] = True

for i in range(0,N):

if(Distance[i] > 0 and Distance[i] < INF):

T = i

R = ""

while(T != S):

for j in range(0,N):

if(M[j,T] < INF and Distance[j] == Distance[T] - M[j,T]):

T = j

R = chr(ord('A') + T) + "-" + R

break

print(R + chr(ord('A')+i))

H0 = H0 + 1

if(TypeTask == 2):

V0 = V0 + 1

Fuel0 = Fuel0 - 1

FuelMinus0Smash = Fuel0 - FuelSmash

print("Текущая высота: " + str(H0))

print("Текущая скорость: " + str(V0))

print("Количество бензина: " + str(Fuel0))

print("Остаток бензина до цели:" + str(FuelMinus0Smash))

Backing = True

def GetVertex():

ErrorConst = -1

V = ' '

MaxLetter = ord('A') + N - 1

m = chr(MaxLetter)

V = 'A'

if(ord(V) > ord(m)):

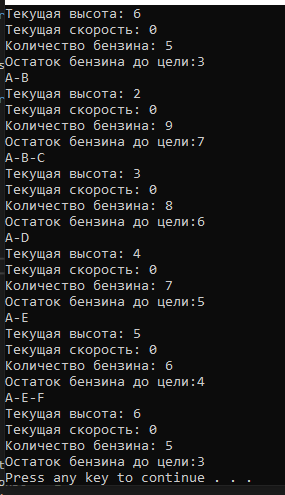
return ErrorConst

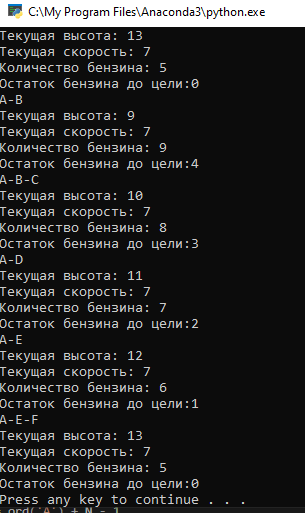
else:

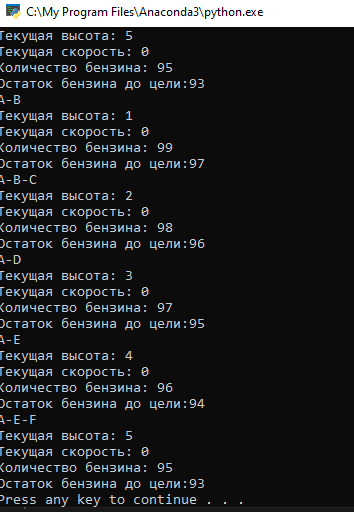
return ord(V) - ord('A')

inputDataAir()

**Скриншоты выполнения программы**







**Постановка задачи 2 с изменёнными условиями**

Решить задачу 1, но с измененным условием2. Самолет может набирать

высоту и скорость одновременно. Требуется найти оптимальный режим набора

высоты и скорости, при котором расход горючего будет минимальным.

**Алгоритм решения задачи 1**

Импортируем numpy as np

n = 6

v0 = 0

h0 = 0

fuel0 = 0

fuelsmash = 0

fuelminus0smash = 0

typetask = 0

inf = 1000000

m = np.array([[inf,1,inf,3,3,inf], [inf,inf,6,inf,inf,inf],[inf,2,inf,inf,inf,9],[inf,inf,inf,4,7,inf],[inf,8,inf,inf,inf,4],[inf,inf,inf,inf,inf,inf]])

Функция inputdataair():

h0 = int(input("введите начальную высоту самолёта: "))

v0 = int(input("введите начальную скорость самолёта: "))

fuel0 = int(input("введите начальное количество топлива самолёта: "))

fuelsmash = int(input("введите конечное количество топлива самолёта: "))

typetask = int(input("задайте тип решения задачи (1/2): "))

print("ну что полетели!"+'\n'+"карта пути:")

print(m)

print('кратчайший путь по карте пути:')

dijkstra(v0,h0,fuel0,fuelsmash,fuelminus0smash,typetask)

Функция dijkstra(v0,h0,fuel0,fuelsmash,fuelminus0smash,typetask):

v0default = v0

h0default = h0

fueldefault = fuel0

distance = []

visited = []

s = getvertex()

if(s == -1):

вернуть

Для i in range(0,n):

distance.append(inf)

visited.append(false)

distance[s] = 0

mind = int()

while(mind < inf):

mind = inf

minv = int(-1)

Для i in range(0,n):

if(distance[i] < mind and visited[i] == false):

mind = distance[i]

minv = i

h0 = h0default

v0 = v0default

fuel0 = fueldefault

if(minv == -1):

выйти

Для i in range(0,n):

if(m[minv,i] < inf and visited[i] == false):

distance[i] = min(distance[i],distance[minv] + m[minv,i])

visited[minv] = истина

Для i in range(0,n):

if(distance[i] > 0 and distance[i] < inf):

t = i

r = ""

while(t != s):

Для j in range(0,n):

if(m[j,t] < inf and distance[j] == distance[t] - m[j,t]):

t = j

r = chr(ord('a') + t) + "-" + r

выйти

print(r + chr(ord('a')+i))

h0 = h0 + 1

if(typetask == 2):

v0 = v0 + 1

fuel0 = fuel0 - 1

fuelminus0smash = fuel0 - fuelsmash

print("текущая высота: " + str(h0))

print("текущая скорость: " + str(v0))

print("количество бензина: " + str(fuel0))

print("остаток бензина до цели:" + str(fuelminus0smash))

backing = истина

Функция getvertex():

errorconst = -1

v = ' '

maxletter = ord('a') + n - 1

m = chr(maxletter)

v = 'a'

if(ord(v) > ord(m)):

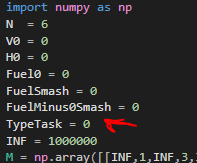
вернуть errorconst

else:

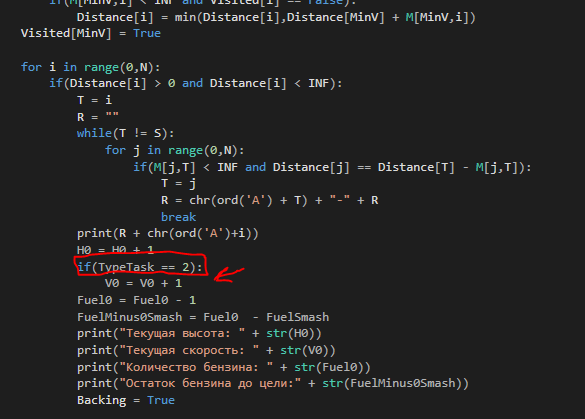
вернуть ord(v) - ord('a')

inputdataair()

**Изменения в программе для задачи 1**

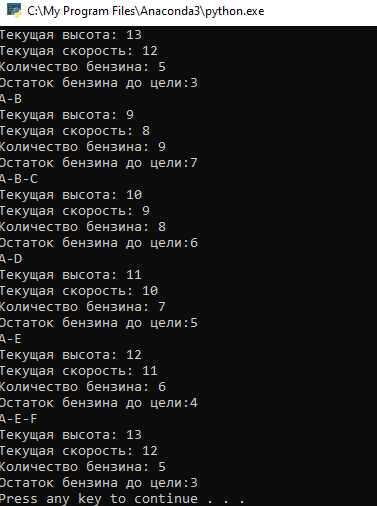


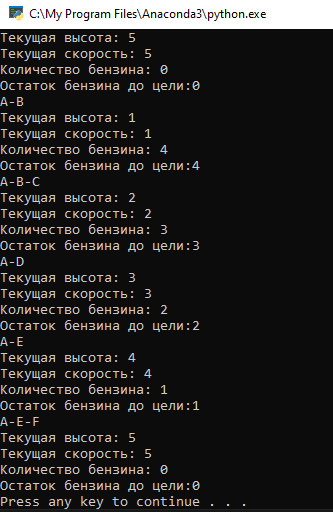
Добавлен тип изменения решения задачи

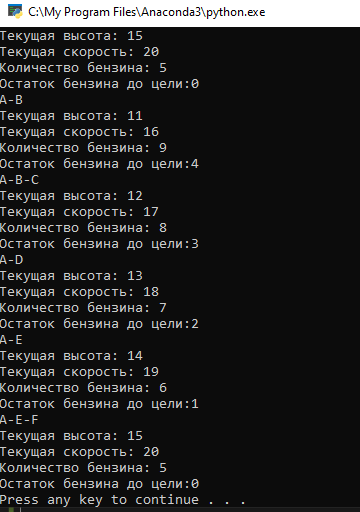


В код посещения вершин добавлено, условие, если тип решения задачи равен 2 ,то тогда увеличиваем скорость при каждом посещение следующей вершины.

**Скриншоты выполнения программы**







**Общий вывод**

Результаты решения задачи 1:

У меня получилось реализовать перемещение самолёта по графу с помощью алгоритма Дейкстры, который ищет кратчайшие пути до указанной цели точки F в данном моём примере матрицы, которая выводиться на экран консольного приложения при полёте самолёта

Каждый раз ,если изменять расстояния матрицы (размеры) ,то по разному тратиться бензин и увеличивается скорость самолёта(отражено на скриншотах выше)

Результаты решения задачи 2:

У меня получилось добавить возможность изменения скорости самолётом, при переходе на следующую вершину, для этого я использовал выбор типа задачи и при входе новую вершину (A-B-C) увеличивается скорость перемещения самолёта 2 – 3 – 4 и т.д

Задача относиться к динамическому программирования, так как является той задачей, в которой нужно применять рекурсию поиск в цикле возврат к начальному состоянию (начальная точка перемещения самолёта) ,а также вывод и поиск кратчайшего пути на матрице смежности. Она задаётся в начале выполнения консольного приложения.