Тверской Государственный технический университет

Кафедра: Программного обеспечения и вычислительной

техники

Лабораторная работа № 3

Выполнил: Студент второго курса

Группы Б.ПИН.РИС 18.05

Федотов Ярослав Всеволодович

Тверь 2019

**Задача о черепашке**

**Формулировка задачи:**

Реализовать решение задачи с полным перебором вариантов.

Алгоритм решения:

Функция fact(принимаем N):

Если(N < 0):

Return 0;

Всё\_если

Если(N == 0):

Return 1

Всё\_если

Иначе:

Return N \* fact(N-1)

Всё\_иначе

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask1.txt', 'запись')

Ввод n,n

Up = n+m - 2

Down = n- 1

FactUp = fact(Up)

FactDown = fact(Down)

Result = FactUp/(FactDown\*FactDown)

Запись в файл(str(Result))

Закрываем запись чтения с файла()

Печать('При полном переборе вариантов получившийся ответ равен: ')

Печать(Result)

Текст программы:

def fact(N):

if(N < 0):

return 0

if(N == 0):

return 1

else:

return N \* fact(N-1)

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask1.txt', 'w')

print('Задача о черепашке\n Требуется используя движение только вправо и вверх\n достигнуть верхнего правого края доски\nТип решения задачи: Полный перебор вариантов\n Введите кол-во столбцов доски')

n = int(input())

print('Введите количество строк доски')

m = int(input())

#Расчёт вверха функции

Up = n+m - 2

#Расчёт низа функции

Down = n- 1

#Вычисление факториала верха и низа

FactUp = fact(Up)

FactDown = fact(Down)

#Вычисление результата

Result = FactUp/(FactDown\*FactDown)

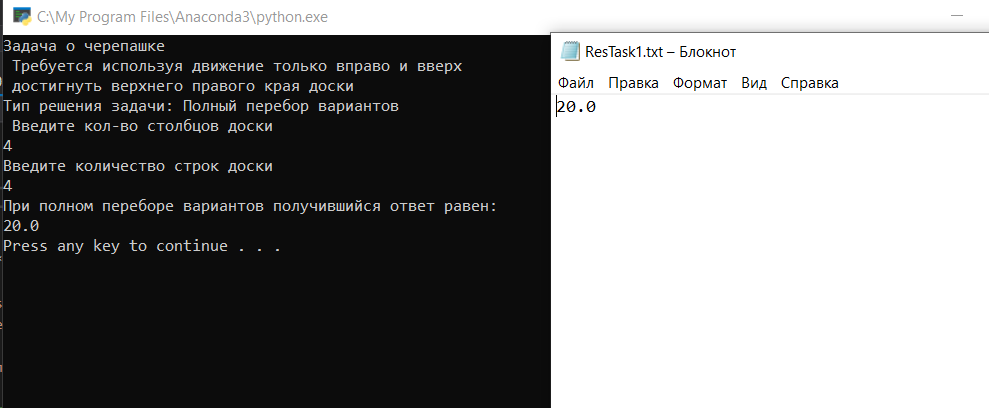
f.write(str(Result))

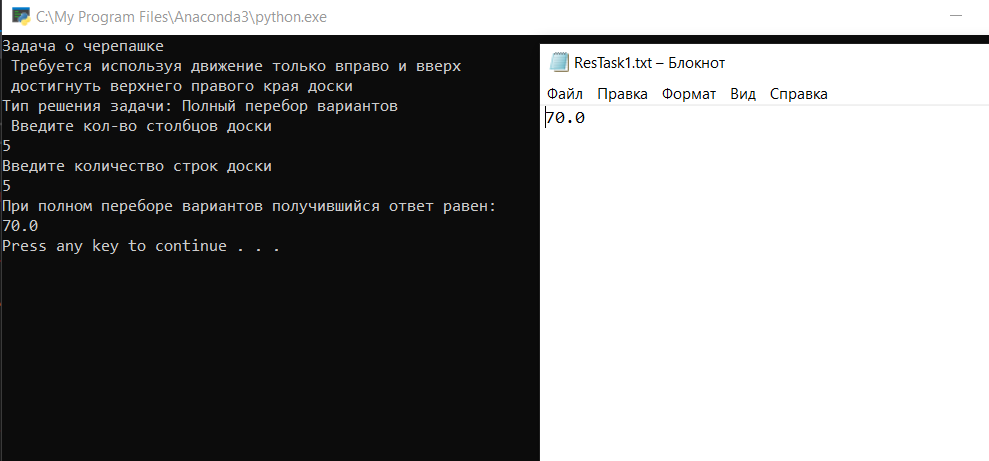
f.close()

print('При полном переборе вариантов получившийся ответ равен: ')

print(Result)

Скриншоты выполнения программы:





Вывод по решению задачи:

Благодаря общему случаю, для движения черепашки вправо и вверх мы можем с легкостью перебрать все варианты, через функцию факториала исходя из общего случая и вывести ответ в файл при помощи f.write(ответ)

**Формулировка задачи:**

Реализовать принцип решения с подзадачей.

Алгоритм решения:

Функция max(принимаем a , принимаем b):

Если(а>b):

Return a

Всё\_если

Иначе:

Return b

Всё\_иначе

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask2.txt', 'запись')

A = np.array([[5,14,27,36],[8,21,39,47],[18,32,52,63],[27,40,58,65]])

B = np.array([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])

Объявление целого I,j,N

N = 4

B[1,1] = A[1,1]

Для I от 0 до N:

B[I,1] = B[I-1,1]+A[I,1]

Всё\_для\_I

Для J от 0 до N:

B[1,J] = B[1,J-1]+A[1,J]

Всё\_для\_J

Для I от 0 до N:

Для J от 2 до N

B[I,j] = Max(B[I-1,j],B[I,J-1])+A[I,j]

Всё\_для\_J

Все\_для\_I

Запись в файл B[N-1,N-1]

Закрыть файл

Печать(B[N-1,N-1])

Текст программы:

print('Задача о черепашке\n Требуется используя движение только вправо и вверх\n достигнуть верхнего правого края доски\nТип принцип задачи: решения с подзадачей')

import numpy as np

def Max(a,b):

if a>b:

return a

else:

return b

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask2.txt', 'w')

A = np.array([[5,14,27,36],[8,21,39,47],[18,32,52,63],[27,40,58,65]])

B = np.array([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])

i = int(); j = int(); N = int();

N = 4

B[0,0] = A[0,0]

for i in range(1,N):

B[i,0] = B[i-1,0]+A[i,0]

for j in range(1,N):

B[0,j] = B[0,j-1]+A[0,j]

for i in range(1,N):

for j in range(1,N):

B[i,j] = Max( B[i-1,j], B[i,j-1] ) + A[i,j];

f.write(str(B))

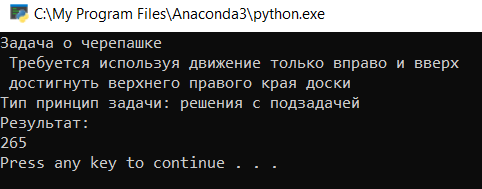
print('Результат:')

f.write(str(B[N-1,N-1]))

f.close()

print(B[N-1,N-1])

Скриншоты выполнения программы:



Вывод по решению задачи:

Благодаря методу решения задачи о черепашке с подзадачей времени на решение и поиск пути уходит меньше, так как время решения подзадачи не возрастает линейно слишком быстро, в отличие от факториала.

**Формулировка задачи:**

Реализовать рекурсивный вариант решения (с элемента , либо ).

Алгоритм решения:

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask3.txt', 'запись')

A = np.array([[5,14,27,36],[8,21,39,47],[18,32,52,63],[27,40,58,65]])

B = np.array([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])

Объявление целого числа N

N = 4

Функция Way(принимаем I,принимаем j):

Если(I = 1) и (j = 1):

Return

Всё\_если

Если(I = 1) и (j > 1):

Way(I,j-1)

Всё\_если

Иначе если(I > 1) и (j = 1):

Way(i-1,j)

Всё\_иначе\_если

Иначе:

Если(B[I,j]-A[I,j] == B[i-1,j]):

Way(i-1,j)

Иначе:

Way(I,j-1)

Записать в файл(I = str(i))

Записать в файл(J = str(j))

Печать(I,j)

Всё\_иначе

Way(N-1,N-1)

Закрыть файл

Текст программы:

import numpy as np

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask3.txt', 'w')

print('Задача о черепашке\n Требуется используя движение только вправо и вверх\n достигнуть верхнего правого края доски\nТип решения задачи: рекурсивный вариант')

A = np.array([[5,14,27,36],[8,21,39,47],[18,32,52,63],[27,40,58,65]])

B = np.array([[5,19,46,82],[13,40,85,132],[31,72,137,200],[58,112,195,265]])

N = int();

N = 4

def Way(i,j):

f.write('i = ')

f.write(str(i))

f.write(' j = ')

f.write(str(j))

f.write(' ;\n')

print(i, ' ',j, ' ;')

if(i == 0) and (j == 0):

return

if(i == 0) and (j > 1): Way(i,j-1)

elif(i > 1) and(j == 1): Way(i-1,j)

else:

if (B[i,j]-A[i,j] == B[i-1,j]):

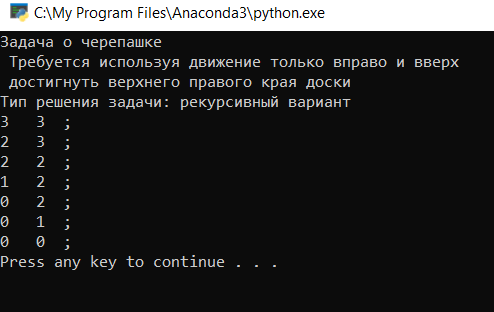
Way(i-1,j)

else: Way(i,j-1)

Way(N-1,N-1)

f.close()

Скриншоты выполнения программы:



Вывод по решению задачи:

Метод Way(I,j) позволяет нам определить, какие координаты клеток задействует черепашка при своём движении к цели и выводит его на экран и записывает в файл.

**Формулировка задачи:**

У Черепашки количество способов достижения конечной клетки при размере таблицы равно . Предположим, что Черепашке разрешено двигаться не только вверх и вправо, но и по диагонали (естественно, на одну клетку). Подсчитать количество способов достижения Черепашкой конечной клетки.

Алгоритм решения:

Функция fact(принимаем N):

Если(N < 0):

Return 0;

Всё\_если

Если(N == 0):

Return 1

Всё\_если

Иначе:

Return N \* fact(N-1)

Всё\_иначе

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask1.txt', 'запись')

Ввод n,n

Up = n+m + 1

Down = n- 1

FactUp = fact(Up)

FactDown = fact(Down)

Result = FactUp/(FactDown\*FactDown)

Запись в файл(str(Result))

Закрываем запись чтения с файла()

Печать('При полном переборе вариантов получившийся ответ равен: ')

Печать(Result)

Текст программы:

def fact(N):

if(N < 0):

return 0

if(N == 0):

return 1

else:

return N \* fact(N-1)

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask4.txt', 'w')

print('Найти количество способов достижения Черепашкой конечной клетки\n При условии , что та может идти:\nВправо,Вверх,По диагонали\nВведите кол-во столбцов')

n = int(input())

print('Введите кол-во строк')

m = int(input())

#Расчёт вверха функции

Up = n+m + 1

#Расчёт низа функции

Down = n- 1

#Вычисление факториала верха и низа

FactUp = fact(Up)

FactDown = fact(Down)

#Вычисление результата

Result = FactUp/(FactDown\*FactDown)

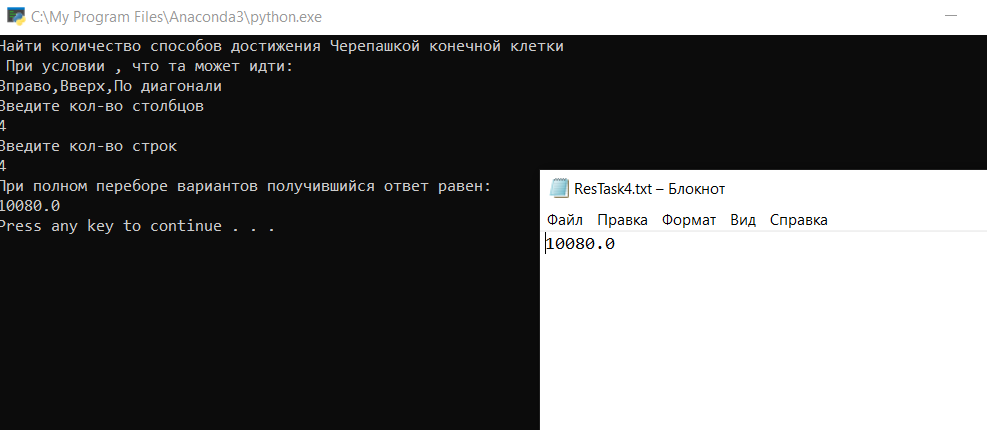
f.write(str(Result))

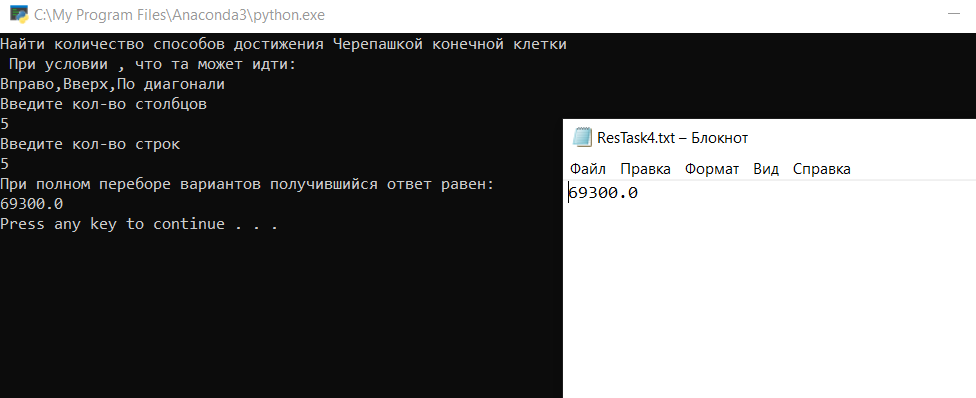
f.close()

print('При полном переборе вариантов получившийся ответ равен: ')

print(Result)

Скриншоты выполнения программы:





Вывод по решению задачи:

Исходя из общего решения ,мы получаем, что для матрицы из 4 столбцов и 4 строк мы можем делать 3 шага вправо и вверх итого 6,а если по диагонали, то прибавляется ещё 3 шага и итого получается следовательно, подставляем получившуюся новую общую формулу в решение задачи и получаем результат.

**Формулировка задачи:**

Какой будет стоимость пути и кратчайший путь Черепашки для условия в пункте 4, если возможен переход по диагонали вправо вверх?

Алгоритм решения:

Функция fact(N):

Если(N < 0):

Return 0

Если(N == 0):

Return 1

Иначе:

Return N \* fact(N-1)

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask5.txt', 'запись')

Ввод n,m

Up = n+m + 7

Down = n- 1

FactUp = fact(Up)

FactDown = fact(Down)

Result = FactUp/(FactDown\*FactDown)

Запись в файл (str(Result))

Закрыть файл

Печать ('При полном переборе вариантов получившийся ответ равен: ')

Печать(Result)

Текст программы:

import numpy as np

def Way(i,j):

f.write('i = ')

f.write(str(i))

f.write(' j = ')

f.write(str(j))

f.write(' ;\n')

print(i, ' ',j, ' ;')

if(i != 3 ) and (j != 3):

if(i == 1) and (j == 1): return

if(i == 0) and (j > 1): Way(i,j-1)

elif(i > 1) and(j == 1): Way(i-1,j)

else:

if (B[i,j]-A[i,j] == B[i-1,j]):

Way(i-1,j)

else: Way(i,j-1)

def Min(a,b):

if a<b:

return a

else:

return b

def fact(N):

if(N < 0):

return 0

if(N == 0):

return 1

else:

return N \* fact(N-1)

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask5.txt', 'w')

print('Найти количество способов достижения Черепашкой конечной клетки\n При условии , что та может идти:\nВправо,Вверх,По диагонали,А также по диагонали вправо и вверх\nВведите кол-во столбцов')

A = np.array([[5,14,27,36],[8,21,39,47],[18,32,52,63],[27,40,58,65]])

B = np.array([[5,19,46,82],[13,40,85,132],[31,72,137,200],[58,112,195,265]])

N = 4

B[0,0] = A[0,0]

for i in range(1,N):

B[i,0] = B[i-1,0]+B[0,i-1]+B[i-1,0]+A[i,0]+A[i-1,0]+A[i,0];

for j in range(1,N):

B[0,j] = B[0,j-1]+B[j-1,0]+B[0,j-1]+A[0,j]+A[0,j-1]+A[0,j]

for i in range(1,N):

for j in range(1,N):

B[i,j] = Min( B[i-1,j], B[i,j-1] ) + A[i,j];

Way(i,j)

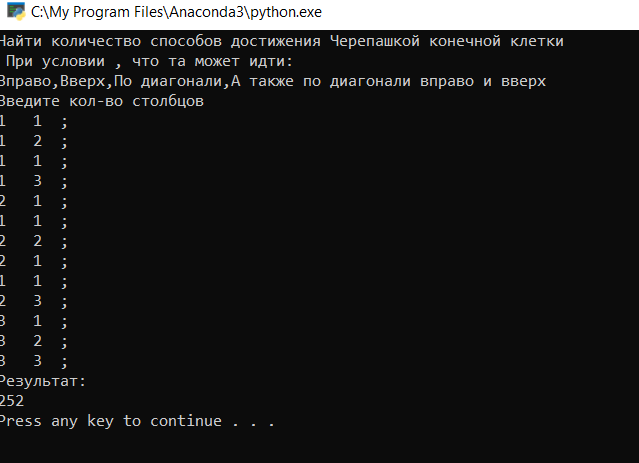
print('Результат:')

f.write(str(B[N-1,N-1]))

f.close()

print(B[N-1,N-1])

Скриншоты выполнения программы:



Вывод по решению задачи:

Исходя из общего решения ,мы получаем, что для матрицы из 4 столбцов и 4 строк мы можем делать 3 шага вправо и вверх итого 6,а если по диагонали, то прибавляется ещё 3 шага и итого получается следовательно, подставляем получившуюся новую общую формулу в решение задачи и получаем результат. Но ещё так как мы можем двигаться по диагонали вправо и вверх мы ещё получаем 6 шагов и получаем ,что общая формула будет изменена на следовательно, подставляем получившуюся новую общую формулу в решение задачи и получаем результат.

**Формулировка задачи:**

1. Предположим, что некоторые клетки поля являются запрещенными для Черепашки. К исходным данным добавляется массив запрещенных клеток. Решить задачу при этом ограничении. Например, входные данные задачи представлены на рисунке 1.

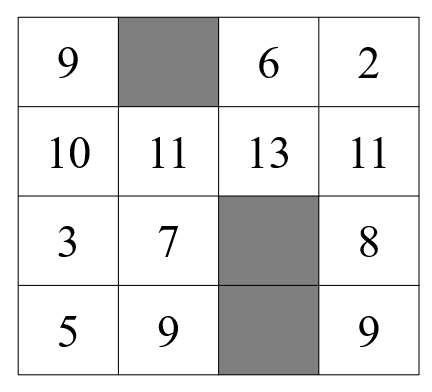


Рисунок 1. Матрица входных данных задачи о Черепашке с ограничениями

Алгоритм программы

Функция Max(принимаем a, принимаем b):

Если(a>b):

Return a

Иначе:

Return b

A = np.loadtxt('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/InputTask6Lab03.txt')

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask6.txt', 'запись')

B = np.array([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])

Объявляем целые числа I,j,N

N = 4

B[1,1] = A[1,1]

Для I от 0 до N:

Если(A[I,1] != -1):

B[I,1] = B[I-1,1]+A[I,1]

Всё\_если

Всё\_для\_I

Для J от 0 до N:

Если(A[1,j] != -1):

B[1,J] = B[1,J-1]+A[1,J]

Всё\_если

Всё\_для\_J

Для I от 0 до N:

Для J от 0 до N:

B[I,j] = Max(B[i-1,j],B[I,j-1]) + A[I,j]

Все\_для\_J

Все\_для\_I

Запись в файл(B[N-1,N-1])

Закрыть файл

Печать(B[N-1,N-1])

Текст программы:

print('Задача о черепашке\n Дана матрица с закрашенными значениями\n Требуется определить максимальный путь\n Движения: Вправо Вверх')

import numpy as np

def Max(a,b):

if a>b:

return a

else:

return b

A = np.loadtxt('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/InputTask6Lab03.txt')

f = open('C:/Users/HP/source/repos/PyhtonProject/PyhtonProject/ResTask6.txt', 'w')

B = np.array([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])

i = int(); j = int(); N = int();

N = 4

B[0,0] = A[0,0]

for i in range(1,N):

B[i,0] = B[i-1,0]+A[i,0]

for j in range(1,N):

B[0,j] = B[0,j-1]+A[0,j]

for i in range(1,N):

for j in range(1,N):

B[i,j] = Max( B[i-1,j], B[i,j-1] ) + A[i,j];

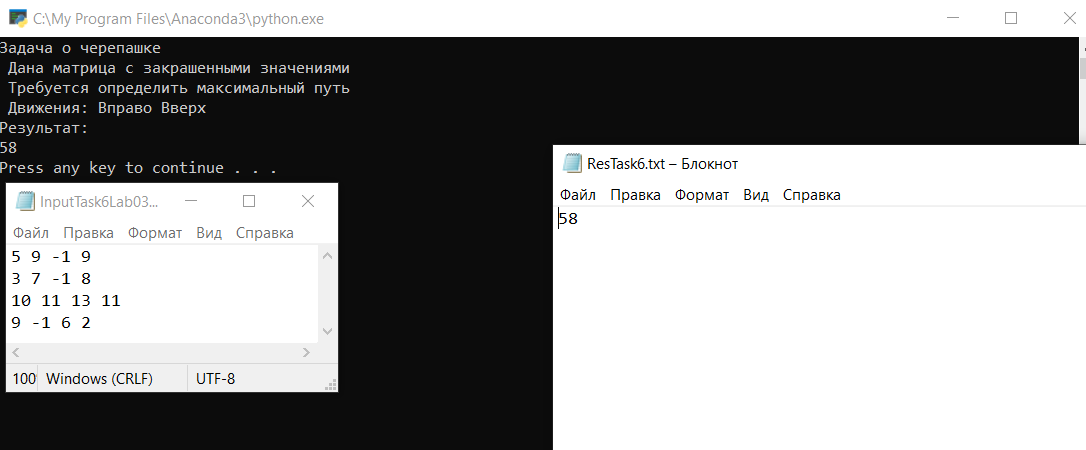
print('Результат:')

f.write(str(B[N-1,N-1]))

f.close()

print(B[N-1,N-1])

Скриншоты выполнения программы:



Вывод по решению задачи:

Через функцию np.loadtxt мы считаем матрицу из текстового файла и написываем её в массив np.array в A. Затем используя проход по массиву, вправо и вверх при этом игнорируя клетки равные -1 получаем результат из складывания пройденных клеток.

Общий вывод:

Данная задача относиться к динамическому программированию потому, что её можно решить различными способами, при этом, используя, каждый способ мы получаем различное время решения задачи.