|  |  |
| --- | --- |
|  | ***Министерство образования и науки Российской Федерации***  *Калужский филиал федерального государственного бюджетного образовательного учреждения высшего профессионального образования*  ***«Московский государственный технический университет  имени Н.Э. Баумана»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | "Фундаментальные науки" |
| **КАФЕДРА** | "Высшая математика" |

**О Т Ч Е Т**

ЛАБОРАТОРНАЯ РАБОТА №4

|  |  |  |
| --- | --- | --- |
| **ДИСЦИПЛИНА:** | | "Дискретная математика" |
| **ТЕМА:** | " Генерация перестановок и сочетаний " | |

|  |  |
| --- | --- |
| Выполнил: студент гр. ИТД.Б-31 | Лысенков А.Е.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Проверил: доцент кафедры ФН3-КФ | Булычев В.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

|  |  |  |
| --- | --- | --- |
| Дата сдачи (защиты) лабораторной работы: | |  |
| Результаты сдачи (защиты):  Количество рейтинговых баллов |  | |
| Оценка | зачтено | |

Калуга, 2015 г.

**Цель работы:**

Реализация алгоритмов полного перебора.  
**Задание:**

1. Прочитайте из файла «data4.txt» значения переменных N и K, строку S (все её символы различны) и матрицу стоимостей A для своего варианта.
2. Подключите пакет комбинаторных вычислений **combinat**.
3. Используя соответствующий алгоритм, составьте программу и выведите на печать все строки, которые можно получить из строки S перестановками её символов, и посчитайте их количество. Проверьте полученный результат с помощью функции **numbperm()**.
4. Используя соответствующий алгоритм, составьте программу и выведите на печать все возможные сочетания по K элементов, которые можно получить из символов строки S, и посчитайте их количество. Проверьте полученный результат с помощью функции **numbcomb()**.
5. Используя программу из пункта 3, составьте программу для решения *задачи коммивояжёра* с матрицей стоимостей A: найдите самый выгодный маршрут, выходящий из города №1, проходящий через все города по одному разу и возвращающийся в город №1. Выведите на печать найденный маршрут и его стоимость.

**Теоретическая часть:**

1. Дайте определение размещения, сочетания, размещения с повторениями и сочетания с повторениями по K элементов из N. Приведите формулы для расчёта числа размещений, сочетаний, размещений с повторениями и сочетаний с повторениями по K элементов из N.

Размещением из N объектов по K называют комбинацию, которая получается в результате последовательного выбора без возвращения любых K объектов из N имеющихся.

Сочетанием из N объектов по K называется комбинация в результате одновременного выбора K объектов из N элементов. Порядок выбора не учитывается.

Размещением с повторениями из N объектов по K называется комбинация, которая получается в результате последовательного выбора с возвращением любых K объектов из N имеющихся. Порядок выбора учитывается, но при этом один и тот же объект можно выбрать много раз.

Сочетанием с повторениями из N объектов по K называют комбинацию, которая получается в результате выбора с возвращением любых k объектов из N имеющихся без учета выбора порядка.

2. Для заданных в вашем варианте N и K вычислите число размещений, сочетаний, размещений с повторениями и сочетаний с повторениями по K элементов из N.

Исходные данные N= 6; K = 3;

; ; ;

**Текст исходной программы:**

read "data.txt":

ArrToStr := proc(word, alphabet)

local str, i:

str := "":

for i from 1 to nops(word) do

str := cat(str, alphabet[floor(word[i])]):

end do:

end proc:

"Вовращает, какие буквы принадлежат слову":

GetSetOfNumber := proc(word, itemsNumb)

local i, answer:

answer := [seq(0, i = 1..itemsNumb)]:

for i from 1 to nops(word) do

if (word[i] <> 0) then

answer[floor(word[i])] := 1:

end if:

end do:

return answer:

end proc:

"Возвращает, какую букву следует изменить и на что":

GetIndexOfNextChangeNumber := proc(word, length, itemsNumb)

local i, j, isBelong, tmp\_word:

tmp\_word := word:

i := length:

while (i > 0) do

isBelong := GetSetOfNumber(tmp\_word, itemsNumb):

for j from word[i] to itemsNumb do

if (isBelong[j] = 0) then

return [i, j]:

end if:

end do:

tmp\_word[i] := 0:

i := i - 1:

end do:

return [0, 0]:

end proc:

"Упорядоченный стартовое слово, с отсутствующими буквами":

GetStartValue := proc(maskBelong, length, itemsNumb)

local i, j, word:

word := [seq(0, i = 1..length)]:

j := 1:

i := 1:

while (i <= itemsNumb) and (j <= length) do

if (maskBelong[i] = 0) then

word[j] := i:

j := j + 1:

end if:

i := i + 1:

end do:

return word:

end proc:

Copy := proc(arr, left, right)

local length, result, i:

length := right - left + 1:

result := [seq(0, i = 1..length)]:

for i from 1 to length do

result[i] := arr[left + i - 1]:

end do:

return result:

end proc:

Combine := proc(arr1, arr2)

local i, length1, length2, result:

length1 := nops(arr1):

length2 := nops(arr2):

result := [seq(0, i = 1..(length1 + length2))]:

for i from 1 to length1 do

result[i] := arr1[i]:

end do:

for i from 1 to length2 do

result[length1 + i] := arr2[i]:

end do:

return result:

end proc:

"Возвращает следующую последовательность":

GetNext := proc(word, length, itemsNumb)

local tmp, index, letter, tmp\_word, i, begin\_word, end\_word:

tmp := GetIndexOfNextChangeNumber(word, length, itemsNumb):

index := tmp[1]:

letter := tmp[2]:

if (index = 0) or (letter = 0) then

return "":

end if:

tmp\_word := word:

tmp\_word[index] := letter:

"Эта часть останется в следующей последовательности":

begin\_word := Copy(tmp\_word, 1, index):

end\_word :=

GetStartValue(

GetSetOfNumber(begin\_word, itemsNumb),

length - index,

itemsNumb

):

tmp\_word := Combine(begin\_word, end\_word):

return tmp\_word:

end proc:

FindAllCombines := proc(str, length, itemsNumb)

local word, i:

i := 0:

word :=

GetStartValue(

[seq(0, i = 1..length)],

length,

itemsNumb

):

while (word <> "") do

"print(ArrToStr(word, str)):":

i := i + 1:

"print(word):":

word := GetNext(word, length, itemsNumb):

end do:

print(i):

end proc:

CalculateWeight := proc (path)

local i, weight:

weight := 0:

for i from 2 to N do

weight := weight + A[path[i - 1], path[i]]:

end do:

weight := weight + A[path[N], path[1]]:

return weight:

end proc:

FindMinPath := proc()

local min\_path, min\_weight, length, i, cur\_path, cur\_weight:

length := N:

cur\_path :=

GetStartValue(

[seq(0, i = 1..length)],

length,

length

):

cur\_weight := CalculateWeight(cur\_path):

min\_path := cur\_path:

min\_weight := cur\_weight:

while (cur\_path <> "") do

cur\_path := GetNext (cur\_path, length, length):

if (cur\_path<>"") then

cur\_weight := CalculateWeight(cur\_path):

if cur\_weight < min\_weight then

min\_weight := cur\_weight:

min\_path := cur\_path:

end if:

end if:

end do:

print("Minimal path: ", min\_path):

print("Minimal weight: ", min\_weight):

end proc:

Summ := proc(setNumber, length)

local i:

answer := 0:

from i from 1 to length do

answer := answer + setNumber[i]:

end do:

return answer:

end proc:

SetOfBase := proc(number, max)

local i, tmp:

basePowered := [seq(1, i = 1..max+1)]:

i := 2:

while (i <= max) do

tmp := basePowered[i - 1]:

basePowered[i] := tmp \* 2:

i = i + 1:

end do:

answer := [seq(0, i = 1..max+1)]:

tmp := number:

i := max + 1:

while (i > 0) do

if (tmp >= basePowered[i]) then

tmp := tmp - basePowered[i]:

answer[i] := 1:

end if:

i := i - 1:

end do:

return answer:

end proc:

FindAllCombines2 := proc(str, length, itemsNumb)

local word, i, max, tmp, tmp2, tmp3, answer:

i := 0:

word := 0:

max := 1:

answer := 0:

from i := 1 to length+1 do

max := max \* 2:

end do:

from i := 0 to max do

tmp := summ(setOfBase(i, length+1), length+1):

if tmp = itemsNumb then

tmp2 := setOfBase(i, length+1):

tmp3 := ArrToStr(tmp2, S):

print(tmp3):

answer = answer + 1:

end if:

end do:

print(answer):

end proc:

print("Number n: ", N):

print("Number k: ", K):

print("String s: ", S):

print("Mtrix A: ", A):

with(combinat):

print("Maple calculation: n po n ", numbperm(N)):

print("My calculation: n po n "):

FindAllCombines(S, N, N):

print("Maple calculation: n po k ", numbcomb(N, K)):

print("My calculation: n po k "):

FindAllCombines2(S, K, N):

FindMinPath():

**Результаты работы программы**

 7

4

"GAWULQY"



5040

"Words " + 5040

"Words " + 840

840

35

"Calculated..."

[1, 4, 3, 5, 7, 6, 2]

435

**Вывод:** в ходе выполнения лабораторной работы составлены программы, осуществляющие полный перебор вариантов для различных типов комбинаций.