**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и

автоматизированныхсистем

Дисциплина: Дискретная математика

Лабораторная работа №2.1

Тема: «Алгоритмы порождения комбинаторных объектов»

Выполнил:

студент группы ВТ-12

Воскобойников Илья Сергеевич

Проверил: Рязанов Ю. Д.

Белгород 2019

Задание 1

Реализовать алгоритм порождения подмножеств:

**void** SetMN(**int** \* r, **int** i){  
 **for** (**int** x = 0; x <= 1 ; x++) {  
 r[i] = x;  
 **if**( i == **U**-1 ) {  
 output2MN(r);  
 } **else**{  
 SetMN(r,i+1);  
 }  
 }  
}



Задание 2

Построить график зависимости количества всех подмножеств от мощности множества:

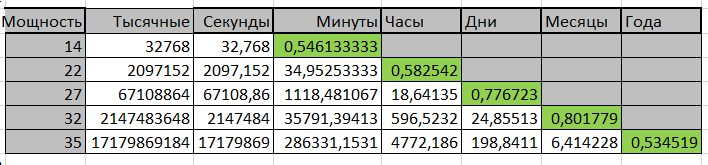
График имеет вид y=2^n где n мощность заданного множества, а y количество всех различных подмножеств этого множества.

Задание 3

Построить графики зависимости времени выполнения алгоритма п.1 на ЭВМ от мощности множества:

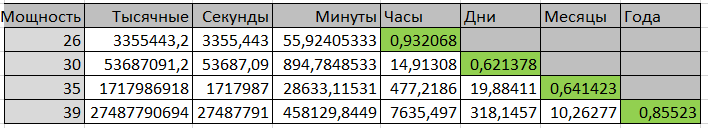
Задание 4

Определить максимальную мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на ЭВМ:

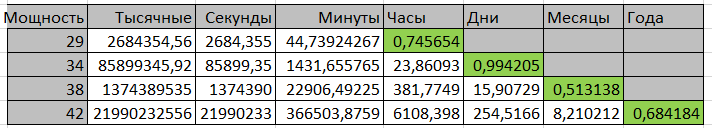


Задание 5

Определить максимальную мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на ЭВМ в 10, 100 раз быстрее:

В 10 раз: 

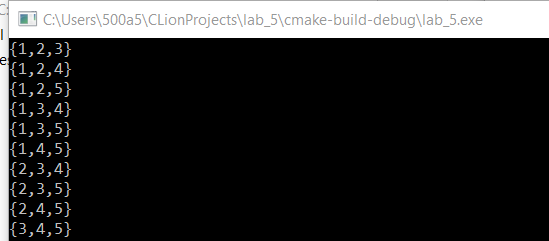
В 100 раз:



Задание 6

Реализовать алгоритм порождения сочетаний:

**void** combination(**int** \*a,**int** i,**int** b){  
 **for**(**int** x=b;x<=(**U**-**K**+i+1);x++){  
 a[i]=x;  
 **if** (i==**K**-1) {  
 outputMN(a,**K**);  
 }**else** {  
 combination(a, i + 1, x + 1);  
 }  
 }  
}



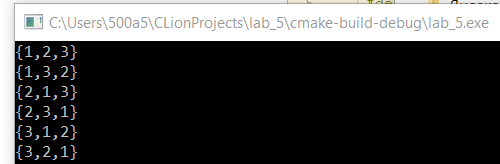
Задание 7

Построить графики зависимости количества всех сочетаний из n по k от k при n = (5,7,9)

Задание 8

Реализовать алгоритм порождения перестановок

**void** transposition(**int** \*a, **int** \*M, **int** i){  
 **for** (**int** j = 0; j < **U**; ++j) {  
 **if**(M[j]){  
 a[i] = M[j];  
 M[j] = 0;  
 **if**( i == **U** - 1) {  
 outputMN(a);  
 }**else** {  
 transposition(a, M, i + 1);  
 }  
 M[j] = a[i];  
 }  
 }  
}



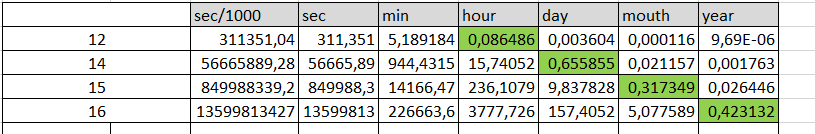
Задание 9

Построить график зависимости количества всех перестановок от мощности множества:

Задание 10

Построить графики зависимости времени выполнения алгоритма п.8 на ЭВМ от мощности множества:

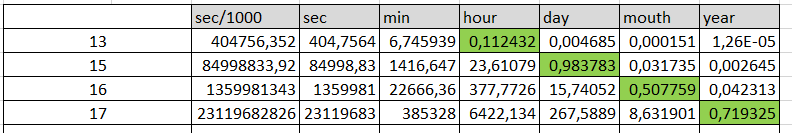
Задание 11

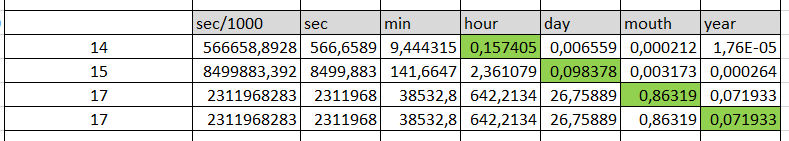
Определить максимальную мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на вашей ЭВМ. 

Задание 12

Определить максимальную мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на ЭВМ, в 10 и в 100 раз быстрее вашей.

В 10 раз:

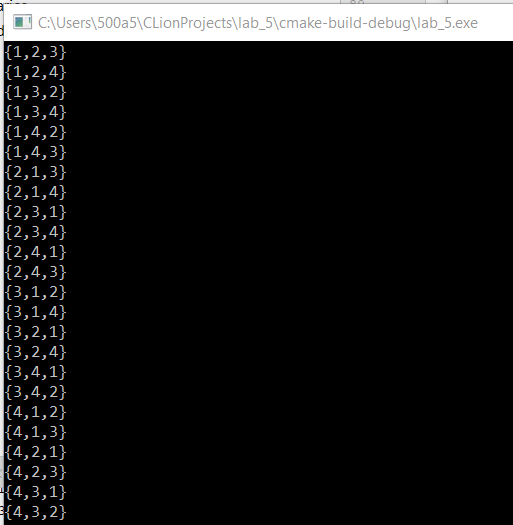


В 100 раз: 

Задание 13

Реализовать алгоритм порождения размещений.

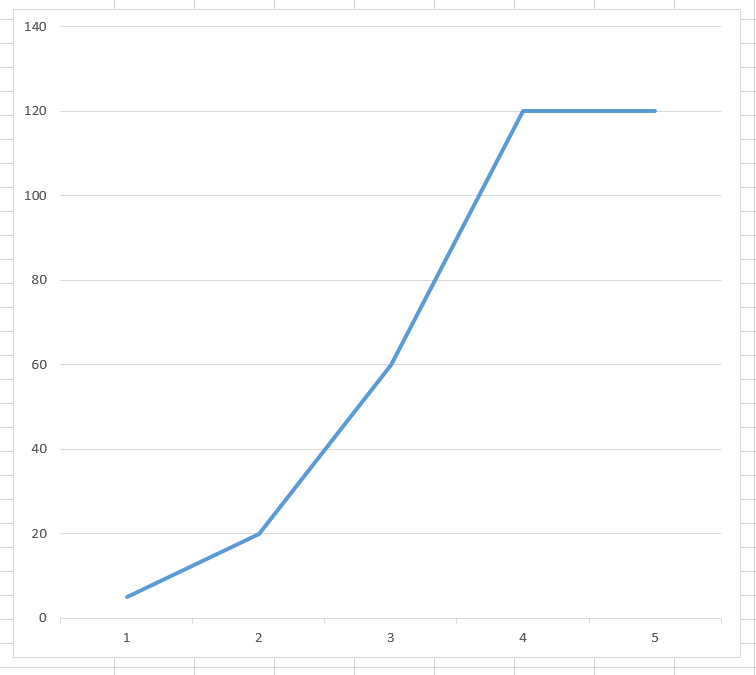
**void** location(**int** \*a,**int** \*M,**int** i){  
 **for** (**int** j = 0; j < **U**; ++j){  
 **if**(M[j]!=0){  
 a[i] = M[j];  
 M[j]=0;  
 **if** (i == **K** - 1) {  
 outputMN(a,**K**);  
 }**else** {  
 location(a, M, i + 1);  
 }  
 M[j]=a[i];  
 }  
 }  
}



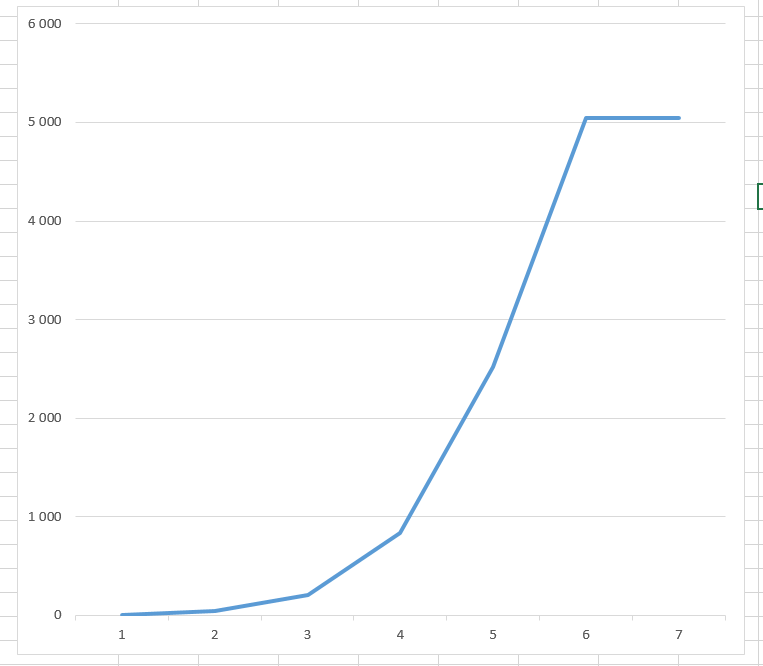
Задание 14

Построить графики зависимости количества всех размещений из n по k от k при n = (5, 7, 9).

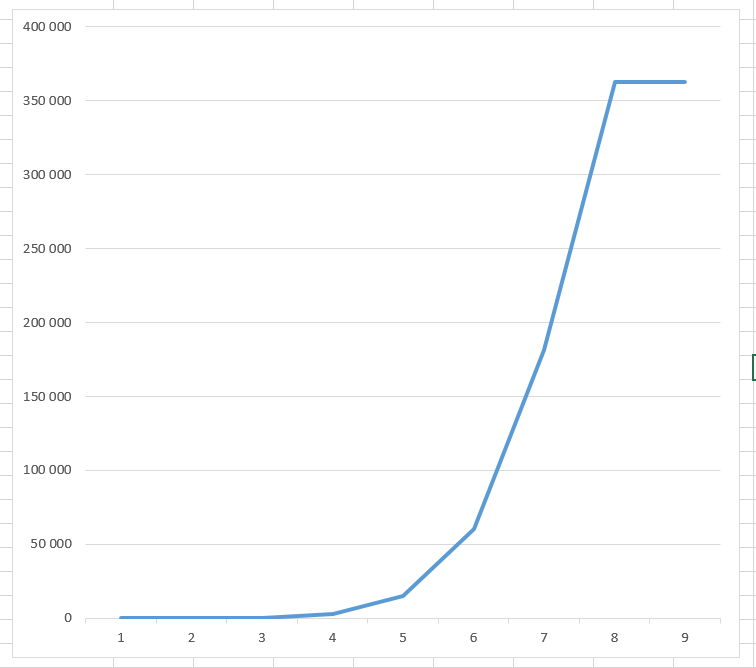
N=5



N=7



N=9



#include **<stdio.h>**#include **<Time.h>**#include **<process.h>**#define **U** 2  
#define **K** 3  
**void** output2MN(**const int** \*a) {  
 **int** j,i;  
 printf(**"{"**);  
 **for**(i=0;i<**U**;i++){  
 **if** (a[i]==1){  
 printf(**"%d"**, i+1);  
 **break**;  
 }  
 }  
 i++;  
 **for** ( j=i; j < **U**; ++j){  
 **if**(a[j]==1) {  
 printf(**",%d"**, j + 1);  
 }  
 }  
 printf(**"}\n"**);  
}  
  
**void** outputMN(**int** \*a,**int** n){  
 printf(**"{%d"**,a[0]);  
 **for**(**int** i=1;i<n;i++) {  
 printf(**",%d"**, a[i]);  
 }  
 printf(**"}\n"**);  
}  
  
**void** SetMN(**int** \* r, **int** i){  
 **for** (**int** x = 0; x <= 1 ; x++) {  
 r[i] = x;  
 **if**( i == **U**-1 ) {  
 output2MN(r);  
 } **else**{  
 SetMN(r,i+1);  
 }  
 }  
}  
  
  
**void** combination(**int** \*a,**int** i,**int** b){  
 **for**(**int** x=b;x<=(**U**-**K**+i+1);x++){  
 a[i]=x;  
 **if** (i==**K**-1) {  
 outputMN(a,**K**);  
 }**else** {  
 combination(a, i + 1, x + 1);  
 }  
 }  
}  
  
  
**void** transposition(**int** \*a, **int** \*M, **int** i){  
 **for** (**int** j = 0; j < **U**; ++j) {  
 **if**(M[j]){  
 a[i] = M[j];  
 M[j] = 0;  
 **if**( i == **U** - 1) {  
 outputMN(a,**U**);  
 }**else** {  
 transposition(a, M, i + 1);  
 }  
 M[j] = a[i];  
 }  
 }  
}  
  
**void** location(**int** \*a,**int** \*M,**int** i){  
 **for** (**int** j = 0; j < **U**; ++j){  
 **if**(M[j]!=0){  
 a[i] = M[j];  
 M[j]=0;  
 **if** (i == **K** - 1) {  
 outputMN(a,**K**);  
 }**else** {  
 location(a, M, i + 1);  
 }  
 M[j]=a[i];  
 }  
 }  
}  
  
  
**int** main() {  
 **int** a[**U**],  
 M[]={1,2};  
 *// SetMN(a,0);  
// combination(a,0,1);* transposition (a,M,0);  
 *//location(a,M,0);* printf(**"%f\n"**,((**float**)clock())/**CLOCKS\_PER\_SEC**);  
 system(**"pause"**);  
 **return** 0;  
}