Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №2**

Комбинаторные алгоритмы

Выполнил:

Студент 2 курса 1 группы ФИТ

Шумова Елизавета Игоревна

**2022 г.**

**Цель работы:** приобрести навыки разработки генераторов подмножеств, перестановок, сочетаний и размещений на С++; научиться применять разработанные генераторы для решения задач о рюкзаке (упрощенную, коммивояжера, об оптимальной загрузке судна и об оптимальной загрузке судна с центровкой).

**Вариант 14**

Решить упрощенную задачу о рюкзаке (веса предметов и их стоимость сгенерировать случайным образом: вместимость рюкзака 300 кг, веса предметов 10 – 300 кг, стоимость предметов 5 – 55 у.е.; количество предметов – 18 шт.)



Рис. 6. Схема решения задачи о рюкзаке с применением генератора множества всех подмножеств

#include "pch.h"

#include <iostream>

#include "Combi.h"

#include "Knapsack.h"

#include <time.h>

#include <iomanip>

#define NN (sizeof(c)/sizeof(int))

int main()

{

setlocale(LC\_ALL, "rus");

int V = 600, // вместимость рюкзака

v[] = { 25, 56, 67, 40, 20, 27, 37, 33, 33, 44, 53, 12,

60, 75, 12, 55, 54, 42, 43, 14, 30, 37, 31, 12 },

c[] = { 15, 26, 27, 43, 16, 26, 42, 22, 34, 12, 33, 30,

12, 45, 60, 41, 33, 11, 14, 12, 25, 41, 30, 40 };

short m[NN];

int maxcc = 0;

clock\_t t1, t2;

std::cout << std::endl << "-------- Задача о рюкзаке --------- ";

std::cout << std::endl << "- вместимость рюкзака : " << V;

std::cout << std::endl << "-- количество ------ продолжительность -- ";

std::cout << std::endl << " предметов вычисления ";

for (int i = 12; i <= 20; i++)

{

t1 = clock();

maxcc = knapsack\_s(V, i, v, c, m);

t2 = clock();

std::cout << std::endl << " " << std::setw(2) << i

<< " " << std::setw(5) << (t2 - t1);

}

std::cout << std::endl << std::endl;

system("pause");

return 0;

}

Knapsack.h:

#pragma once

#include "Combi.h"

int knapsack\_s(

int V, // [in] вместимость рюкзака

short n, // [in] количество типов предметов

const int v[], // [in] размер предмета каждого типа

const int c[], // [in] стоимость предмета каждого типа

short m[] // [out] количество предметов каждого типа

);

Combi.h:

#pragma once

namespace combi

{

struct subset // генератор множества всех подмножеств

{

short n, // количество элементов исходного множества < 64

sn, // количество элементов текущего подмножества

\*sset; // массив индексов текущего подмножества

unsigned \_\_int64 mask; // битовая маска

subset(short n = 1); // конструктор(количество элементов исходного множества)

short getfirst(); // сформормировать массив индексов по битовой маске

short getnext(); // ++маска и сформировать массив индексов

short ntx(short i); // получить i-й элемент массива индексов

unsigned \_\_int64 count(); // вычислить общее количество подмножеств

void reset(); // сбросить генератор, начать сначала

};

};

Knapsack.cpp:

#include "pch.h"

#include "Knapsack.h"

#define NINF 0x80000000 // самое малое int-число

int calcv(combi::subset s, const int v[]) // объем в рюкзаке

{

int rc = 0;

for (int i = 0; i < s.sn; i++) rc += v[s.ntx(i)];

return rc;

};

int calcc(combi::subset s, const int v[], const int c[]) //стоимость в рюкзаке

{

int rc = 0;

for (int i = 0; i < s.sn; i++) rc += (v[s.ntx(i)] \* c[s.ntx(i)]);

return rc;

};

void setm(combi::subset s, short m[]) //отметить выбранные предметы

{

for (int i = 0; i < s.n; i++) m[i] = 0;

for (int i = 0; i < s.sn; i++) m[s.ntx(i)] = 1;

};

int knapsack\_s(

int V, // [in] вместимость рюкзака

short n, // [in] количество типов предметов

const int v[], // [in] размер предмета каждого типа

const int c[], // [in] стоимость предмета каждого типа

short m[] // [out] количество предметов каждого типа {0,1}

)

{

combi::subset s(n);

int maxc = NINF, cc = 0;

short ns = s.getfirst();

while (ns >= 0)

{

if (calcv(s, v) <= V)

if ((cc = calcc(s, v, c)) > maxc)

{

maxc = cc;

setm(s, m);

}

ns = s.getnext();

};

return maxc;

};

Combi.cpp:

#include "pch.h"

#include "Combi.h"

#include <algorithm>

namespace combi

{

subset::subset(short n)

{

this->n = n;

this->sset = new short[n];

this->reset();

};

void subset::reset()

{

this->sn = 0;

this->mask = 0;

};

short subset::getfirst()

{

\_\_int64 buf = this->mask;

this->sn = 0;

for (short i = 0; i < n; i++)

{

if (buf & 0x1) this->sset[this->sn++] = i;

buf >>= 1;

}

return this->sn;

};

short subset::getnext()

{

int rc = -1;

this->sn = 0;

if (++this->mask < this->count()) rc = getfirst();

return rc;

};

short subset::ntx(short i)

{

return this->sset[i];

};

unsigned \_\_int64 subset::count()

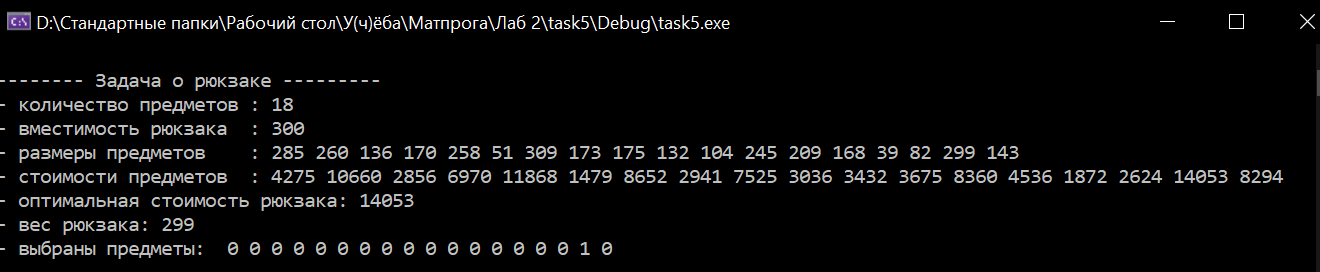
{

return (unsigned \_\_int64)(1 << this->n);

};

};

**Задание №5. Результат:**



**Результат:**

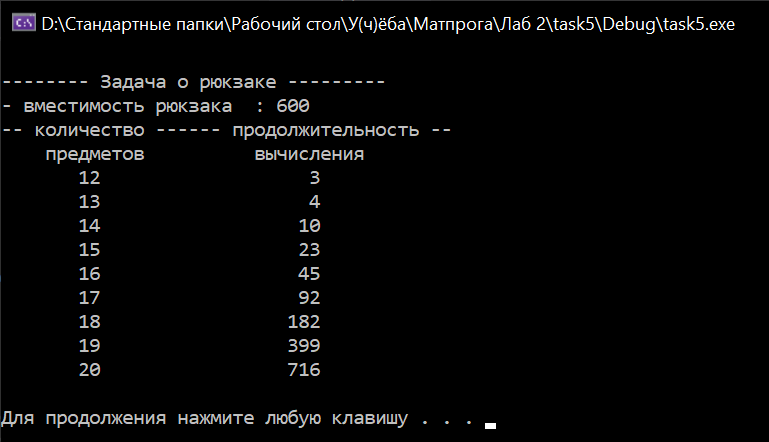


График Excel, показывающий зависимость времени вычисления от размерности задачи:

|  |  |
| --- | --- |
| **Кол-во предметов** | **Продолжительность  вычисления** |
| **12** | **3** |
| **13** | **4** |
| **14** | **10** |
| **15** | **23** |
| **16** | **45** |
| **17** | **92** |
| **18** | **182** |
| **19** | **399** |
| **20** | **716** |