Учреждение образования

«Белорусский государственный технологический университет»

**Лабораторная работа №2**

**Комбинаторные алгоритмы решения оптимизационных задач**

Выполнил:

Студент 2 курса 7 группы ФИТ

Кореневский Кирилл Русланович

**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.**

***Задание 1-5***. Разобрать и разработать генератор подмножеств заданного множества, сочетаний, перестановок, размещений, решить задачу о оптимальной загрузке судна с условием центровки.



Файл 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

void reset();

unsigned \_\_int64 count();

short ntx(short i);

};

struct xcombination // генератор сочетаний (эвристика)

{

short n, // количество элементов исходного множества

m, nc, // количество элементов в сочетаниях

\* sset; // массив индексов текущего сочетания

xcombination(

short n = 1, // количество элементов исходного множества

short m = 1 // количество элементов в сочетаниях

);

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

short getfirst(); // сформировать первый массив индексов

short getnext(); // сформировать следующий массив индексов

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

unsigned \_\_int64 count() const;

};

struct permutation // генератор перестановок

{

const static bool L = true; // левая стрелка

const static bool R = false; // правая стрелка

short n, // количество элементов исходного множества

\* sset; // массив индексов текущей перестановки

bool\* dart; // массив стрелок (левых-L и правых-R)

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

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

\_\_int64 getfirst(); // сформировать первый массив индексов

\_\_int64 getnext(); // сформировать случайный массив индексов

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

unsigned \_\_int64 np; // номер перествновки 0,... count()-1

unsigned \_\_int64 count() const; // вычислить общее кол. перестановок

};

struct accomodation // генератор размещений

{

short n, // количество элементов исходного множества

m, // количество элементов в размещении

\* sset; // массив индесов текущего размещения

xcombination\* cgen; // указатель на генератор сочетаний

permutation\* pgen; // указатель на генератор перестановок

accomodation(short n = 1, short m = 1); // конструктор

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

short getfirst(); // сформировать первый массив индексов

short getnext(); // сформировать следующий массив индексов

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

unsigned \_\_int64 na; // номер размещения 0, ..., count()-1

unsigned \_\_int64 count() const; // общее количество размещений

};

}

Файл Combi.cpp

#include "Combi.h"

#include <algorithm>

#define NINF ((short)0x8000)

namespace combi

{

// подмножества множества

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);

};

// сочетания

xcombination::xcombination(short n, short m)

{

this->n = n;

this->m = m;

this->sset = new short[m + 2];

this->reset();

}

void xcombination::reset() // сбросить генератор, начать сначала

{

this->nc = 0;

for (int i = 0; i < this->m; i++)

this->sset[i] = i;

this->sset[m] = this->n;

this->sset[m + 1] = 0;

};

short xcombination::getfirst()

{

return (this->n >= this->m) ? this->m : -1;

};

short xcombination::getnext() // сформировать следующий массив индексов

{

short rc = getfirst();

if (rc > 0)

{

short j;

for (j = 0; this->sset[j] + 1 == this->sset[j + 1]; ++j)

this->sset[j] = j;

if (j >= this->m)

rc = -1;

else {

this->sset[j]++;

this->nc++;

};

}

return rc;

};

short xcombination::ntx(short i)

{

return this->sset[i];

};

unsigned \_\_int64 fact(unsigned \_\_int64 x) {

return(x == 0) ? 1 : (x \* fact(x - 1));

};

unsigned \_\_int64 xcombination::count() const

{

return (this->n >= this->m) ? fact(this->n) / (fact(this->n - this->m) \* fact(this->m)) : 0;

};

// перестановки

permutation::permutation(short n)

{

this->n = n;

this->sset = new short[n];

this->dart = new bool[n];

this->reset();

};

void permutation::reset()

{

this->getfirst();

};

\_\_int64 permutation::getfirst()

{

this->np = 0;

for (int i = 0; i < this->n; i++)

{

this->sset[i] = i;

this->dart[i] = L;

};

return (this->n > 0) ? this->np : -1;

};

\_\_int64 permutation::getnext() //

{

\_\_int64 rc = -1;

short maxm = NINF, idx = -1;

for (int i = 0; i < this->n; i++)

{

if (i > 0 &&

this->dart[i] == L &&

this->sset[i] > this->sset[i - 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

if (i < (this->n - 1) &&

this->dart[i] == R &&

this->sset[i] > this->sset[i + 1] &&

maxm < this->sset[i]) maxm = this->sset[idx = i];

};

if (idx >= 0)

{

std::swap(this->sset[idx], this->sset[idx + (this->dart[idx] == L ? -1 : 1)]);

std::swap(this->dart[idx], this->dart[idx + (this->dart[idx] == L ? -1 : 1)]);

for (int i = 0; i < this->n; i++)

if (this->sset[i] > maxm)

this->dart[i] = !this->dart[i];

rc = ++this->np;

}

return rc;

};

short permutation::ntx(short i) {

return this->sset[i];

};

unsigned \_\_int64 factorial(unsigned \_\_int64 x) {

return (x == 0) ? 1 : (x \* factorial(x - 1));

};

unsigned \_\_int64 permutation::count() const {

return factorial(this->n);

}

accomodation::accomodation(short n, short m)

{

this->n = n;

this->m = m;

this->cgen = new xcombination(n, m);

this->pgen = new permutation(m);

this->sset = new short[m];

this->reset();

}

void accomodation::reset()

{

this->na = 0;

this->cgen->reset();

this->pgen->reset();

this->cgen->getfirst();

};

short accomodation::getfirst()

{

short rc = (this->n >= this->m) ? this->m : -1;

if (rc > 0)

{

for (int i = 0; i <= this->m; i++)

this->sset[i] = this->cgen->sset[this->pgen->ntx(i)];

};

return rc;

};

short accomodation::getnext()

{

short rc;

this->na++;

if ((this->pgen->getnext()) > 0) rc = this->getfirst();

else if ((rc = this->cgen->getnext()) > 0)

{

this->pgen->reset();

rc = this->getfirst();

};

return rc;

};

short accomodation::ntx(short i)

{

return this->sset[i];

};

unsigned \_\_int64 factor(unsigned \_\_int64 x) {

return (x == 0) ? 1 : (x \* factor(x - 1));

};

unsigned \_\_int64 accomodation::count() const

{

return (this->n >= this->m) ? fact(this->n) / fact(this->n - this->m) : 0;

};

};

Файл Boat.h

#pragma once

#include "Combi.h"

int boat(

int V,

short m,

short n,

const int v[],

const int c[],

short r[]

);

Файл Boat.cpp

#include "Boat.h"

namespace boatfnc

{

int calcv(combi::xcombination s, const int v[])

{

int rc = 0;

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

return rc;

};

int calcc(combi::xcombination s, const int c[])

{

int rc = 0;

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

return rc;

};

void copycomb(short m, short\* r1, const short\* r2)

{

for (int i = 0; i < m; i++) r1[i] = r2[i];

};

}

int boat(

int V,

short m,

short n,

const int v[],

const int c[],

short r[]

)

{

combi::xcombination xc(n, m);

int rc = 0, i = xc.getfirst(), cc = 0;

while (i > 0)

{

if (boatfnc::calcv(xc, v) <= V)

{

if ((cc = boatfnc::calcc(xc, c)) > rc)

{

rc = cc; boatfnc::copycomb(m, r, xc.sset);

}

}

i = xc.getnext();

};

return rc;

Файл main.cpp

#include <iostream>

#include <iomanip>

#include "Combi.h"

#include "Boat.h"

#include "Auxil.h"

#define N (sizeof(AA4)/2)

#define M 3

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

int main()

{

// генерация подмножеств

setlocale(LC\_ALL, "rus");

char AA[][2] = { "A", "B", "C", "D" };

std::cout << " - Генератор множества всех подмножеств -" << std::endl;

std::cout << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA) / 2; i++)

std::cout << AA[i] << ((i < sizeof(AA) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация всех подмножеств ";

combi::subset s1(sizeof(AA) / 2); // создание генератора

int n = s1.getfirst(); // первое (пустое) подмножество

while (n >= 0) // пока есть подмножества

{

std::cout << std::endl << "{ ";

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

std::cout << AA[s1.ntx(i)] << ((i < n - 1) ? ", " : " ");

std::cout << "}";

n = s1.getnext(); // cледующее подмножество

};

std::cout << std::endl << "всего: " << s1.count() << std::endl;

// генерация сочетаний

char AA2[][2] = { "A", "B", "C", "D", "E" };

std::cout << std::endl << " --- Генератор сочетаний ---" << std::endl;

std::cout << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA2) / 2; i++)

std::cout << AA2[i] << ((i < sizeof(AA2) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация сочетаний ";

combi::xcombination xc(sizeof(AA2) / 2, 3);

std::cout << "из " << xc.n << " по " << xc.m;

int n2 = xc.getfirst();

while (n2 >= 0)

{

std::cout << std::endl << xc.nc << ": { ";

for (int i = 0; i < n2; i++)

std::cout << AA2[xc.ntx(i)] << ((i < n2 - 1) ? ", " : " ");

std::cout << "}";

n2 = xc.getnext();

};

std::cout << std::endl << "всего: " << xc.count() << std::endl;

// генерация перестановок

char AA3[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " --- Генератор перестановок ---" << std::endl;

std::cout << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < sizeof(AA3) / 2; i++)

std::cout << AA3[i] << ((i < sizeof(AA3) / 2 - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация перестановок ";

combi::permutation p(sizeof(AA3) / 2);

\_\_int64 n3 = p.getfirst();

while (n3 >= 0)

{

std::cout << std::endl << std::setw(4) << p.np << ": { ";

for (int i = 0; i < p.n; i++)

std::cout << AA3[p.ntx(i)] << ((i < p.n - 1) ? ", " : " ");

std::cout << "}";

n3 = p.getnext();

};

std::cout << std::endl << "всего: " << p.count() << std::endl;

// генерация размещений

char AA4[][2] = { "A", "B", "C", "D" };

std::cout << std::endl << " --- Генератор размещений ---";

std::cout << std::endl << "Исходное множество: ";

std::cout << "{ ";

for (int i = 0; i < N; i++)

std::cout << AA4[i] << ((i < N - 1) ? ", " : " ");

std::cout << "}";

std::cout << std::endl << "Генерация размещений из " << N << " по " << M;

combi::accomodation s(N, M);

int n4 = s.getfirst();

while (n4 >= 0)

{

std::cout << std::endl << std::setw(2) << s.na << ": { ";

for (int i = 0; i < 3; i++)

std::cout << AA4[s.ntx(i)] << ((i < n4 - 1) ? ", " : " ");

std::cout << "}";

n4 = s.getnext();

};

std::cout << std::endl << "всего: " << s.count() << std::endl;

//задача об оптимальной загрузке судна

clock\_t t1 = 0, t2 = 0;

auxil::start();

const int MM = 10;

const int CA = 10;

int v[CA]; // вес

for (int i = 0; i < CA; i++)

{

v[i] = auxil::iget(100, 200);

}

int c[CA]; // доход

for (int i = 0; i < CA; i++)

{

c[i] = auxil::iget(10, 100);

}

int minv[MM]; // минимальный вес

for (int i = 0; i < MM; i++)

{

minv[i] = auxil::iget(50, 120);

}

int maxv[MM]; // максимальный вес

for (int i = 0; i < MM; i++)

{

maxv[i] = auxil::iget(150, 850);

}

short r[MM];

t1 = clock();

int cc = boat\_c(

MM, // количество мест для контейнеров

minv, // максимальный вес контейнера на каждом месте

maxv, // минимальный вес контейнера на каждом месте

NN, // всего контейнеров

v, // вес каждого контейнера

c, // доход от перевозки каждого контейнера

r // номера выбранных контейнеров

);

t2 = clock();

std::cout << std::endl << "- Задача о размещении контейнеров на судне -";

std::cout << std::endl << "- общее количество контейнеров : " << NN;

std::cout << std::endl << "- количество мест для контейнеров : " << MM;

std::cout << std::endl << "- минимальный вес контейнера : ";

for (int i = 0; i < MM; i++) std::cout << std::setw(3) << minv[i] << " ";

std::cout << std::endl << "- максимальный вес контейнера : ";

for (int i = 0; i < MM; i++) std::cout << std::setw(3) << maxv[i] << " ";

std::cout << std::endl << "- вес контейнеров : ";

for (int i = 0; i < NN; i++) std::cout << std::setw(3) << v[i] << " ";

std::cout << std::endl << "- доход от перевозки : ";

for (int i = 0; i < NN; i++) std::cout << std::setw(3) << c[i] << " ";

std::cout << std::endl << "- выбраны контейнеры (0,1,...,m-1) : ";

for (int i = 0; i < MM; i++) std::cout << r[i] << " ";

std::cout << std::endl << "- доход от перевозки : " << cc;

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

std::cout << std::endl << "количество мест: " << MM;

std::cout << std::endl << "продолжительность (у.е): " << (t2 - t1);

std::cout << std::endl << " (сек): "

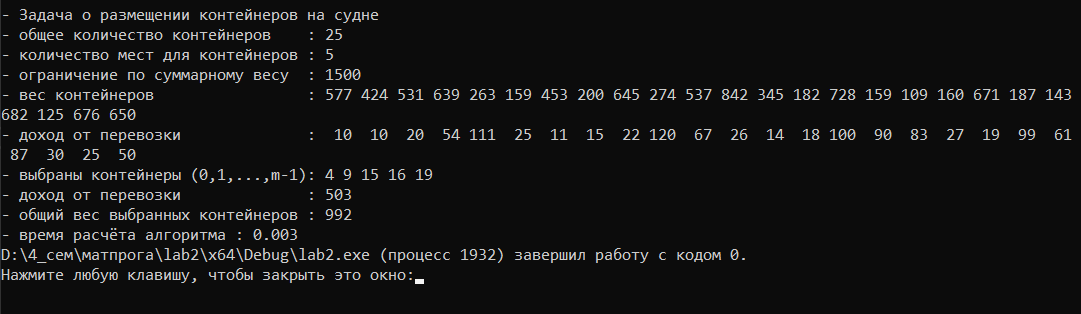
<< ((double)(t2 - t1)) / ((double)CLOCKS\_PER\_SEC);

std::cout << std::endl;

system("pause");

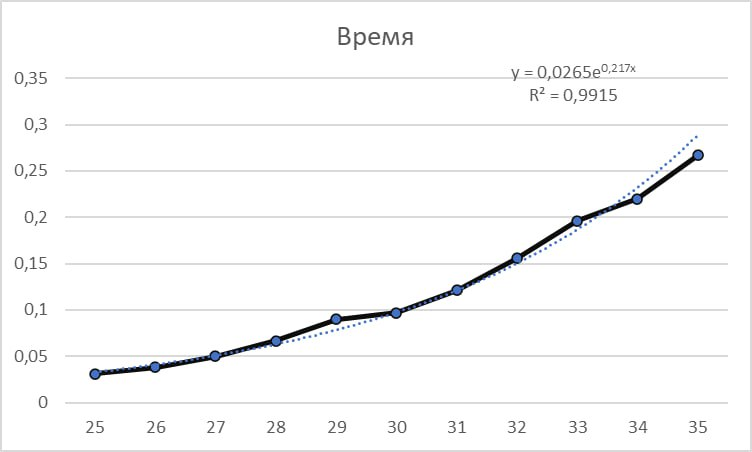
return 0;

}

****

Решение задачи

***Задание 6***. Исследовать зависимость времени вычисления необходимое для решения задачи.



Зависимость времени вычисления от размерности задачи

**ВЫВОД: В ходе выполнения лабораторной работы были приобретены навыки разработки генераторов подмножеств, сочетаний, перестановок, размещений, а также решения задач с их помощью. Была исследована зависимость времени вычисления от размерности задачи.**