**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Множества

Студенты гр. 5307 Ахмадов Н.М, Самоуков Н.В.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Колинько П.Г.

Санкт-Петербург

2016

Оглавление

[1. ЗАДАНИЕ 3](#_Toc461443032)

[2. ФОРМАЛИЗАЦИЯ ЗАДАНИЯ 3](#_Toc461443033)

[3. КОНТРОЛЬНЫЕ ТЕСТЫ 3](#_Toc461443034)

[4. ВРЕМЕННАЯ СЛОЖНОСТЬ 3](#_Toc461443035)

[5. РЕЗУЛЬТАТ ИЗМЕРЕНИЯ ВРЕМЕНИ ОБРАБОТКИ 4](#_Toc461443036)

[6. ВЫВОД 4](#_Toc461443037)

[7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 4](#_Toc461443038)

[8. ПРИЛОЖЕНИЕ 4](#_Toc461443039)

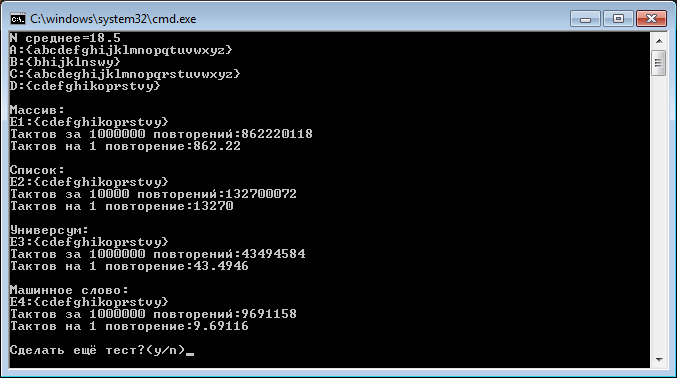
1. ЗАДАНИЕ

На универсуме строчных латинских букв есть 4 множества A, B, C, D. Надо вычислить множество, содержащее все буквы, общие для множеств A и B, за исключением букв, содержащихся в С, а также все буквы с D.

2. ФОРМАЛИЗАЦИЯ ЗАДАНИЯ

**E = A ⋂ B \ C ∪ D**

3. КОНТРОЛЬНЫЙ ТЕСТ



4. ВРЕМЕННАЯ СЛОЖНОСТЬ

Для массивов предполагаемая сложность о(n).

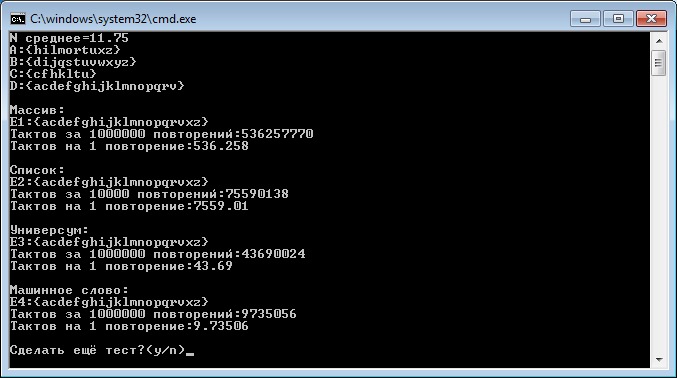
Для достижения данной сложности было решено обрабатывать множество по порядку: сначала обработать A и B потом их результат обработать с C, и данный результат обработать с D. Для уменьшения сложности алгоритма пришлось в функцию обработки подавать отсортированные исходные данные. При этом стало возможно использовать метод снижающий теоретическое время обработки с о(n2) до о(n). Суть метода заключается в том, что мы проходим одновременно оба массива прибавляя индекс у того массива, у которого сейчас меньший или равный элемент. Тот же метод был использован и для списков.

Для списка предполагаемая сложность о(n).

Для универсума предполагаемая сложность о(1).

Для машинное слово предполагаемая сложность о(1).

5. РЕЗУЛЬТАТ ИЗМЕРЕНИЯ ВРЕМЕНИ ОБРАБОТКИ



Для массивов и списков время обработки зависит от N, а для универсума и машинного слова время обработки является константой и не зависит от N. Также было замечено что используемая функция для измерения времени \_\_rtdsc выдаёт разные результаты даже для одинаковых исходных данных, отклонение результатов от среднего при этом около 20%, это свидетельствует о том, что на работу программы влияют некоторые факторы.

6. ВЫВОД

Выполняя данную лабораторную работу, мы получили навыки работы с четырьмя способами представления множеств. А именно в виде массива элементов множества, списка с элементами множества, отображения на универсум в виде массива и с помощью машинного слова. Также был разработан метод, описанный в пункте 4, позволяющий уменьшить сложность обработки первых двух способов с о(n2) до o(n).

По нашему мнению, лучший способ для представления множества – машинное слово. Другие способы более пригодны для крайних случаев.

7. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Колинько П. Г. Алгоритмы и структуры данных: методические указания к лабораторным работам, практическим занятиям и курсовому проектированию. Ч. 1. – СПб.: Изд-во СПбГЭТУ <<ЛЭТИ>>, 2016. – 63 с.

8. ПРИЛОЖЕНИЕ

Файл: code.cpp – Исходный код.

#include <windows.h>

#include <conio.h>

#include <time.h>

#include <iostream>

using namespace std;

#pragma intrinsic(\_\_rdtsc)

const long REP1 = 1000000;

const long REP2 = 10000;

const long REP3 = 1000000;

const long REP4 = 1000000;

const char ST\_CH = 'a';

const char EN\_CH = 'z';

const int N\_CH = 26;

struct DATA1 //массив

{

char a[N\_CH + 1];

};

struct SP\_EL

{

char ch;

SP\_EL \*n;

};

struct DATA2 //список

{

SP\_EL \*spis;

};

void freesp(DATA2 \_sp)

{

while (\_sp.spis != nullptr)

{

SP\_EL\* p;

p = \_sp.spis;

\_sp.spis = \_sp.spis->n;

delete p;

}

\_sp.spis = nullptr;

}

struct DATA3 //юниверсум

{

bool a[N\_CH];

};

struct DATA4 //сжатый юниверсум

{

long a;

};

char\* gen(int \_n = rand() % (N\_CH - 5) + 5)

{

static char s[N\_CH + 1];

bool n[N\_CH];

int r = 0;

if (\_n > N\_CH)

\_n = N\_CH;

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

n[i] = 0;

while (\_n > 0)

{

int i;

i = rand() % N\_CH;

while (n[i])

i = (i + 1) % N\_CH;

n[i] = 1;

\_n--;

}

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

if (n[i])

{

s[r] = ST\_CH + i;

r++;

}

s[r] = 0;

return s;

}

void init(char \*\_s, DATA1 \*\_a)

{

int ais[N\_CH];

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

ais[i] = 0;

while (\*\_s)

{

if (\*\_s >= ST\_CH && \*\_s <= EN\_CH)

if (ais[\*\_s - ST\_CH] == 0)

{

ais[\*\_s - ST\_CH] = 1;

}

\_s++;

}

int r = 0;

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

if (ais[i] == 1)

{

\_a->a[r] = i + ST\_CH;

r++;

}

\_a->a[r] = 0;

}

void init(char \*\_s, DATA2 \*\_a)

{

int ais[N\_CH];

SP\_EL \*p;

\_a->spis = NULL;

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

ais[i] = 0;

while (\*\_s)

{

if (\*\_s >= ST\_CH && \*\_s <= EN\_CH)

if (ais[\*\_s - ST\_CH] == 0)

{

ais[\*\_s - ST\_CH] = 1;

}

\_s++;

}

for (int i = N\_CH-1; i >= 0; i--)

if (ais[i] == 1)

{

p = new SP\_EL;

p->n = \_a->spis;

p->ch = i + ST\_CH;

\_a->spis = p;

}

}

void init(char \*\_s, DATA3 \*\_a)

{

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

\_a->a[i] = 0;

while (\*\_s)

{

if (\*\_s >= ST\_CH && \*\_s <= EN\_CH)

\_a->a[\*\_s - ST\_CH] = 1;

\_s++;

}

}

void init(char \*\_s, DATA4 \*\_a)

{

\_a->a = 0;

while (\*\_s)

{

if (\*\_s >= ST\_CH && \*\_s <= EN\_CH)

{

\_a->a |= 1 << (\*\_s - ST\_CH);

}

\_s++;

}

}

void put(DATA1 \_a)

{

cout << "{" << \_a.a << "}\n";

}

void put(DATA2 \_a)

{

cout << "{";

for (SP\_EL \*p = \_a.spis; p; p = p->n)

cout << p->ch;

cout << "}\n";

}

void put(DATA3 \_a)

{

cout << "{";

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

if (\_a.a[i])

{

cout << (char)(i + ST\_CH);

}

cout << "}\n";

}

void put(DATA4 \_a)

{

cout << "{";

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

if (\_a.a & (1 << i))

{

cout << (char)(i + ST\_CH);

}

cout << "}\n";

}

void processing(DATA1 &\_a, DATA1 &\_b, DATA1 &\_c, DATA1 &\_d, DATA1 &\_e);

void processing(DATA2 &\_a, DATA2 &\_b, DATA2 &\_c, DATA2 &\_d, DATA2 &\_e);

void processing(DATA3 &\_a, DATA3 &\_b, DATA3 &\_c, DATA3 &\_d, DATA3 &\_e);

void processing(DATA4 &\_a, DATA4 &\_b, DATA4 &\_c, DATA4 &\_d, DATA4 &\_e);

unsigned long long time\_req()

{

return \_\_rdtsc();

}

int main()

{

srand(time(nullptr));

setlocale(0, "RU");

char s\_a[30], s\_b[30], s\_c[30], s\_d[30];

int i;

char ch;

do{

system("cls");

DATA1 a1;

DATA1 b1;

DATA1 c1;

DATA1 d1;

DATA1 e1;

DATA2 a2;

DATA2 b2;

DATA2 c2;

DATA2 d2;

DATA2 e2;

DATA3 a3;

DATA3 b3;

DATA3 c3;

DATA3 d3;

DATA3 e3;

DATA4 a4;

DATA4 b4;

DATA4 c4;

DATA4 d4;

DATA4 e4;

strcpy(s\_a, gen());

strcpy(s\_b, gen());

strcpy(s\_c, gen());

strcpy(s\_d, gen());

cout << "N среднее=" << (strlen(s\_a) + strlen(s\_b) + strlen(s\_c) + strlen(s\_d)) / 4.0

<< " \n";

init(s\_a, &a1);

init(s\_b, &b1);

init(s\_c, &c1);

init(s\_d, &d1);

init(s\_a, &a2);

init(s\_b, &b2);

init(s\_c, &c2);

init(s\_d, &d2);

init(s\_a, &a3);

init(s\_b, &b3);

init(s\_c, &c3);

init(s\_d, &d3);

init(s\_a, &a4);

init(s\_b, &b4);

init(s\_c, &c4);

init(s\_d, &d4);

unsigned \_\_int64 t1, t2;

cout << "A:";

put(a1);

cout << "B:";

put(b1);

cout << "C:";

put(c1);

cout << "D:";

put(d1);

cout << "\n";

t1 = time\_req();

for (i = 0; i < REP1; i++)

{

e1.a[0] = 0;

processing(a1, b1, c1, d1, e1);

}

t2 = time\_req();

cout << "Массив:\n";

cout << "E1:";

put(e1);

cout << "Тактов за " << REP1 << " повторений:" << t2 - t1 << "\n";

cout << "Тактов на 1 повторение:" << (t2 - t1) / (float)REP1 << "\n\n";

t1 = time\_req();

for (i = 0; i < REP2; i++)

{

processing(a2, b2, c2, d2, e2);

freesp(e2);

}

t2 = time\_req();

processing(a2, b2, c2, d2, e2);

cout << "Список:\n";

cout << "E2:";

put(e2);

freesp(a2);

freesp(b2);

freesp(c2);

freesp(d2);

freesp(e2);

cout << "Тактов за " << REP2 << " повторений:" << t2 - t1 << "\n";

cout << "Тактов на 1 повторение:" << (t2 - t1) / (float)REP2 << "\n\n";

t1 = time\_req();

for (i = 0; i < REP3; i++)

{

e3.a[0] = 0;

processing(a3, b3, c3, d3, e3);

}

t2 = time\_req();

cout << "Универсум:\n";

cout << "E3:";

put(e3);

cout << "Тактов за " << REP3 << " повторений:" << t2 - t1 << "\n";

cout << "Тактов на 1 повторение:" << (t2 - t1) / (float)REP3 << "\n\n";

t1 = time\_req();

for (i = 0; i < REP4; i++)

{

if (i % (REP4 / 10) == 0)

cout << " \b";

e4.a = 0;

processing(a4, b4, c4, d4, e4);

}

t2 = time\_req();

cout << "Машинное слово:\n";

cout << "E4:";

put(e4);

cout << "Тактов за " << REP4 << " повторений:" << t2 - t1 << "\n";

cout << "Тактов на 1 повторение:" << (t2 - t1) / (float)REP4 << "\n\n";

cout << "Сделать ещё тест?(y/n)";

do

{

ch = getch();

} while (ch != 'y' && ch != 'Y' &&ch != 'n' &&ch != 'N');

} while (ch == 'y' || ch == 'Y');

}

void processing(DATA1 &\_a, DATA1 &\_b, DATA1 &\_c, DATA1 &\_d, DATA1 &\_e)

{

// 1 2 3

//a & b / c | d

int r = 0;

DATA1 f;

//1

for (char \*s1 = \_a.a, \*s2 = \_b.a; \*s1 && \*s2; \*s1 > \*s2 ? s2++ : s1++)

if (\*s1 == \*s2)

{

f.a[r] = \*s1;

r++;

}

f.a[r] = 0;

//2

for (char \*s1 = f.a, \*s2 = \_c.a; \*s1 && \*s2; \*s1 > \*s2 ? s2++ : s1++)

if (\*s1 == \*s2)

{

\*s1 = '-';

r++;

}

r = 0;

for (char \*s1 = f.a; \*s1; s1++)

if (\*s1 != '-')

{

f.a[r] = \*s1;

r++;

}

f.a[r] = 0;

//3

r = 0;

char \*s1 = f.a, \*s2 = \_d.a;

while (\*s1 || \*s2)

{

if (\*s1 > \*s2 && \*s2 != 0 || \*s1 == 0)

{

if ((r == 0 || \*s2 != \_e.a[r - 1]) && \*s2 != 0)

{

\_e.a[r] = \*s2;

r++;

}

s2++;

}

else

{

if ((r == 0 || \*s1 != \_e.a[r - 1]) && \*s1 != 0)

{

\_e.a[r] = \*s1;

r++;

}

s1++;

}

}

\_e.a[r] = 0;

}

void processing(DATA2 &\_a, DATA2 &\_b, DATA2 &\_c, DATA2 &\_d, DATA2 &\_e)

{

// 1 2 3

//a & b / c | d

DATA2 f;

f.spis = nullptr;

//1

SP\_EL \*p;

for (SP\_EL \*p1 = \_a.spis, \*p2 = \_b.spis; p1 != nullptr && p2 != nullptr;

p1->ch > p2->ch ? p2 = p2->n : p1 = p1->n)

if (p1->ch == p2->ch)

{

if (f.spis == nullptr)

{

p = new SP\_EL;

f.spis = p;

p->n = nullptr;

}

else

{

p->n = new SP\_EL;

p = p->n;

p->n = nullptr;

}

p->ch = p1->ch;

}

//2

SP\_EL \*\*g = &f.spis;

for (SP\_EL \*p1 = f.spis, \*p2 = \_c.spis; p1 && p2;)

{

if (p1->ch == p2->ch)

{

p = p1->n;

delete (\*g);

\*g = p;

p1 = (\*g);

}

if (p1)

if (p1->ch > p2->ch)

{

p2 = p2->n;

}

else

{

p1 = p1->n;

g = &((\*g)->n);

}

}

//3

SP\_EL \*p1 = f.spis, \*p2 = \_d.spis;

p = 0;

\_e.spis = 0;

while (p1 || p2)

{

if (p1 == 0 || p2 != 0 && p1->ch > p2->ch)

{

if (p == nullptr || p2->ch != p->ch)

{

if (p == nullptr)

{

p = new SP\_EL;

\_e.spis = p;

}

else

{

p->n = new SP\_EL;

p = p->n;

}

p->n = nullptr;

p->ch = p2->ch;

}

p2 = p2->n;

}

else

{

if (p == nullptr || p1->ch != p->ch)

{

if (p == nullptr)

{

p = new SP\_EL;

\_e.spis = p;

}

else

{

p->n = new SP\_EL;

p = p->n;

}

p->n = nullptr;

p->ch = p1->ch;

}

p1 = p1->n;

}

}

freesp(f);

}

void processing(DATA3 &\_a, DATA3 &\_b, DATA3 &\_c, DATA3 &\_d, DATA3 &\_e)

{

//a & b / c | d

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

\_e.a[i] = ((\_a.a[i] & \_b.a[i]) & (~\_c.a[i])) | \_d.a[i];

}

void processing(DATA4 &\_a, DATA4 &\_b, DATA4 &\_c, DATA4 &\_d, DATA4 &\_e)

{

//a & b / c | d

\_e.a = ((\_a.a & \_b.a) & (~\_c.a)) | \_d.a;

}