КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

**ЛАБОРАТОРНА РОБОТА №1,2,3**

Роботу виконав:

**Пядик Любомир Сергійович**

студент групи МІ-3, факультету кібернетики КНУ

Київ – 2018

**№ 1.** Алгоритм перетворення розрідженої матриці з двовимірного масиву у мультисписок з вузлами з ненульових значень.

У алгоритмі використовується матриця, яка заповнена на незначий відсоток випадковими

елементами та мультисписок. Шляхом проходження по матриці заповнюємо відповідний список у мультисписку, зберігаючи значення елемента та положення у матриці. Складність виконання такого алгоритму квадратична по часу виконання O(N^2), де N - розмір матриці, та використовувана пам'ять залежить від кількості елементів у вхідній матриці.

constexpr int N = 7, M = 20;

constexpr int MOD = 50;

struct cell

{

int\* data = nullptr;

cell\* next\_row = nullptr;

cell\* next\_col = nullptr;

};

int table[N][N];

cell multilist[N][N];

cell\* last\_row = nullptr;

cell\* last\_cols[N];

int main()

{

std::srand(std::time(nullptr));

for (size\_t i = 0; i < M; ++i)

{

size\_t x = std::rand() % N;

size\_t y = std::rand() % N;

int value = std::rand() % MOD;

table[x][y] = value;

}

for (size\_t i = 0; i < N; ++i)

{

for (size\_t j = 0; j < N; ++j)

{

std::cout << "\t" << table[i][j] << " ";

}

std::cout << std::endl;

}

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

{

last\_row = nullptr;

for (int j = N - 1; j >= 0; --j)

{

if (!table[i][j]) continue;

multilist[i][j].data = &table[i][j];

multilist[i][j].next\_row = last\_row;

multilist[i][j].next\_col = last\_cols[j];

last\_row = &multilist[i][j];

last\_cols[j] = &multilist[i][j];

}

}

std::cout << std::endl;

for (size\_t i = 0; i < N; ++i)

{

for (size\_t j = 0; j < N; ++j)

{

if (!multilist[i][j].data) continue;

std::cout << "\t(" << (multilist[i][j].next\_row ? std::to\_string(\*multilist[i][j].next\_row->data) : "null") << "," << (multilist[i][j].next\_col ? std::to\_string(\*multilist[i][j].next\_col->data) :

"null") << ")" << " ";

}

std::cout << std::endl;

}

return EXIT\_SUCCESS;

}

**№ 2.** Алгоритм, що з використанням роздільного зв'язування вставляє N випадкових чисел у таблицю розміром N/100,а потім визначає довжину найдовшого та найкоротшого списків,

N = 10^3, 10^4, 10^5, 10^6.

Алгоритм базується на таблиці розміром N/100, у якій зберігаються списки із числовими

елементами згенерованими випадковим чином у програмі. Кожному випадковому числу ставиться у відповідність індекс (id) у таблиці за допомогою деякої наперед заданої фунції. Після добавляння усіх чисел до таблиці проходом по ній визначається найбільший та найменший списки. Складність агоритму лінійна по кількості чисел у плані часу виконання O(N+M) і також лінійна у плані додаткової пам'яті O(N), де M - кількість списків.

const int N = 100000;

const int M = N / 100;

std::vector< std::list<int> > table;

std::size\_t hash(int x)

{

return x % M;

}

int main()

{

std::srand(std::time(nullptr));

table.resize(M);

for (std::size\_t i = 0; i < N; ++i)

{

int value = std::rand();

table[hash(value)].push\_back(value);

}

std::size\_t longest = 0;

std::size\_t shortest = 0;

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

{

if (table[i] < table[shortest])

{

shortest = i;

}

if (table[i] > table[longest])

{

longest = i;

}

}

std::cout << "Shortest: value=" << shortest << " cnt=" << table[shortest].size() << std::endl;

std::cout << "Longest: value=" << longest << " cnt=" << table[longest].size() << std::endl;

return EXIT\_SUCCESS;

}

**№ 3.** Реалізувати структуру данних типу "множина рядків".

Рядки непусті послідовності довжиною до 15 символів рядкових латинських літер.

Структура данних повинна підтримувати операції додавання рідку до множини, перевіри належності, вилучення.

Максимальна кількість елементів у множині, що зберігається - 106.

"Ідеальна асимптотика" O(m+n), насправді - O(mn) за рахунок колізій

Вх.данні

Кожен рядок вхідних данних задає одну операцію над множиною.

Запис операції - тип операції та наступний за ним рядку над яким проводить операцію.

+ - insert

? - check

- - delete

К-сть операцій у вх.файлі до 106.

Список операцій завершується рядком із символом #.

При додавання елемента не гарантується, що він відсутній у множині.

При вилученні не гарантується присутність.

Вих.данні

Для кожної операції ? - рядок Yes/No.

Для віднімання - повідомлення про успішність операції.

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <list>

using namespace std;

class string\_set

{

public:

string\_set(size\_t capacity)

: m\_data(capacity)

{}

void insert(const string& s)

{

auto& list = m\_data[hash(s)];

list.push\_back(s);

}

bool contains(const string& s)

{

auto& list = m\_data[hash(s)];

return find(list.begin(), list.end(), s) != list.end();

}

bool erase(const string& s)

{

auto& list = m\_data[hash(s)];

auto it = find(list.begin(), list.end(), s);

if (it != list.end())

{

list.erase(it);

return true;

}

return false;

}

private:

vector<list<string>> m\_data;

constexpr static int MOD = 1e9 + 7;

int hash(const string& s)

{

int result = 0;

for (size\_t i = 0; i < s.size(); ++i)

{

result = (result \* 2 + s[i]) % MOD;

}

return result % m\_data.size();

}

};

int main()

{

string\_set s(1000);

ifstream file("input.txt");

string type;

while (file >> type)

{

string value;

if (type == "+")

{

file >> value;

s.insert(value);

}

else if (type == "?")

{

file >> value;

cout << "Contains '" << value << "'? = " << (s.contains(value) ? "OK" : "FAIL") << endl;

}

else if (type == "-")

{

file >> value;

cout << "Erase '" << value << "' = " << (s.erase(value) ? "OK" : "FAIL") << endl;

}

else

{

break;

}

}

file.close();

return 0;

}