ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Алгоритмы сортировки»

Выполнил работу

Старцев Иван

Академическая группа №J3110

Принято

Практик, Владислав Вершинин

Санкт-Петербург

2024

1. Введение

Цель работы: Реализовать 3 алгоритма сортировки и сравнить их в действии

Задачи:

* Выбрать алгоритма для исследования
* Проанализировать их сложность
* Провести замеры алгоритмов на различных наборах данных
* Понять для каких случаев стоит использовать той или иной алгоритм.

1. Теоретическая подготовка

Во всех сортировках в основном используются обычный vector для хранения входных данных. Но в HeapSort используется такая структура данных как двоичная куча, а в CountingSort мы пользуемся префиксными суммами.

1. Реализация

Первым этапом был поиск алгоритмов для реализации. Вторым шло изучение и понимание работы каждого из них. Третим этапом была разработка данных алгоритмов и тестов к ним для проверки работоспособности. Финальным элементом была аналитика времени работы от разных размеров входных данных.

1. Экспериментальная часть

Временная сложность Cocktailshakingsort – O(N^2)

Временная сложность HeapSort – O(N \* log N)

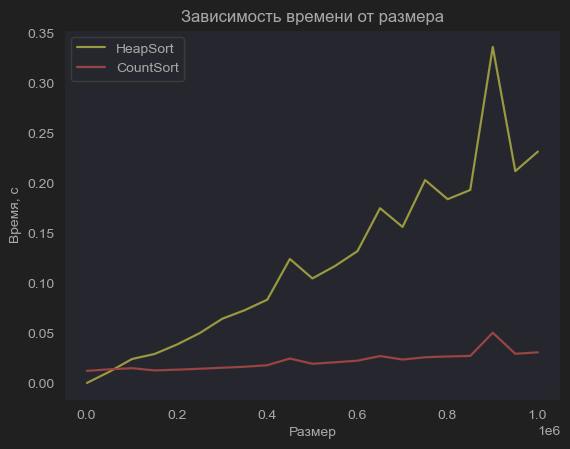
Временная сложность CountingSort – O(N + M)

Пространственная сложность Cocktailshakingsort – O(1)

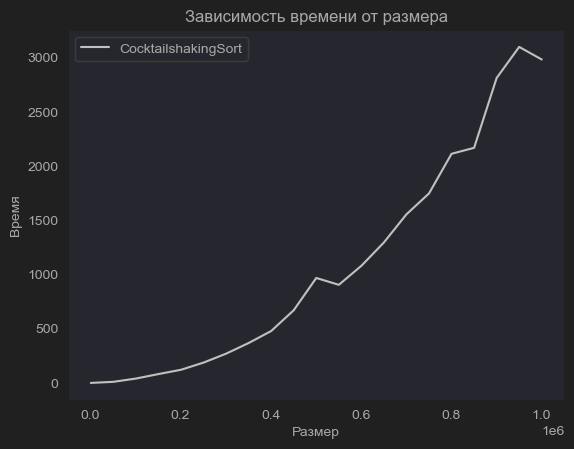
Пространственная сложность HeapSort – O(1)

Пространственная сложность CountingSort – O(N + K)

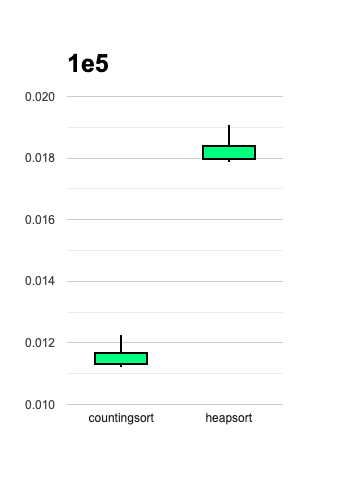
Линейные графики и графики Box plot будут на изображения №1-№6 ниже.



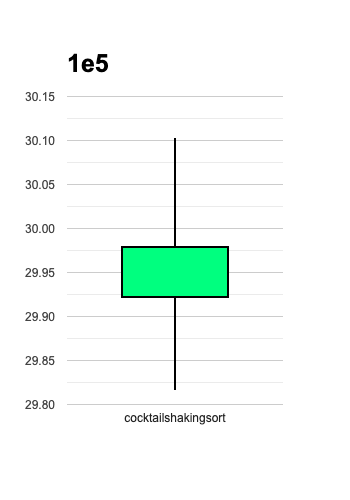
Изображение №1 - График времени для Heap и Counting



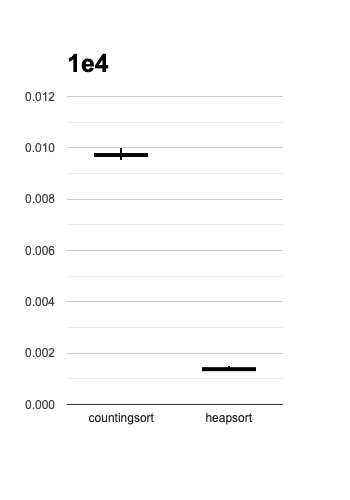
Изображение №2 – график времени для Cocktailshaking



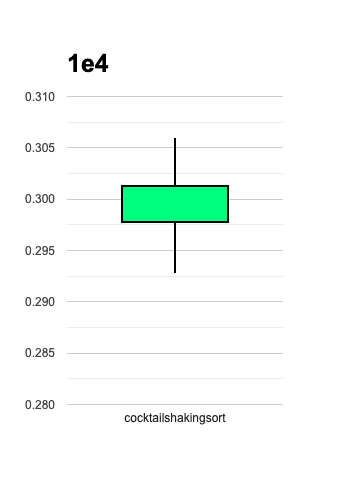
Изображение №3 – box plot для counting и heap c размером 1е5



Изображение №4 – box plot для Cocktailshaking с размером 1е5



Изображение №5 – box plot для counting и heap с размером 1е4



Изображение №6 – box plot для cocktailshaking с размером 1е4

На данных графиках видно что Cocktailshakingsort проигрывает на любом размере входных данных, но это и понятно, ведь сложность O(n^2). А Вот с двумя другими алгоритмами ситуация немного другая, heap опережает counting при маленьком размере входных данных, но потом становится медленее т.к n\*logn > n + k при большом n.

1. Заключение

Мною были реализованы 3 алгоритма сортировки:

* CocktailShakingSort
* HeapSort
* CountingSort

Цель работы была достигнута путем запуска алгоритмов на большом количестве различных входных данных и аналитике полученного времени. Полученные данные совпадают с теоретическими оценками сложности алгоритмов.

Подводя итог, Cocktailshaking проигрывает своим конкурентам на любом размере, а вот heap стоит использовать только при сортировке небольшого кол-ва значений.

ПРИЛОЖЕНИЕ А

Листинг кода файла cocktailshakingsort.cpp

#include <iostream>

#include <fstream>

#include <sstream>

#include <ctime>

using namespace std;

int main(int argc, char\* argv[]) {

ifstream inputFile(argv[1]);

if (!inputFile) {

cerr << "Не удалось открыть файл!" << endl;

return 1;

}

string line;

vector<int> a;

if (getline(inputFile, line)) {

stringstream ss(line);

string cell;

while (getline(ss, cell, '\t')) {

a.push\_back(stoi(cell));

}

}

inputFile.close();

clock\_t begin = clock();

int size = a.size() - 1;

bool finish = false;

while (!finish) {

finish = true;

for (int i = 0; i < size; i++) {

if (a[i] > a[i + 1]) {

swap(a[i], a[i + 1]);

finish = false;

}

}

if (finish) {

break;

}

finish = true;

for (int i = size - 1; i >= 1; i--) {

if (a[i] < a[i - 1]) {

swap(a[i], a[i - 1]);

finish = false;

}

}

}

clock\_t end = clock();

double elapsed = (double) (end - begin) / CLOCKS\_PER\_SEC;

cout << elapsed << endl;

//for (int i = 0; i <= size; i++) {

// cout << a[i] << " ";

//}

//cout << endl;

}

ПРИЛОЖЕНИЕ Б

#include <iostream>

#include <fstream>

#include <sstream>

#include <ctime>

using namespace std;

void heapify(int \*arr, int size, int i){

int l = 2\*i+1;

int r = 2\*i+2;

int maxx = i;

if (l<size && arr[l]>arr[maxx]){

maxx = l;

}

if (r<size && arr[r]>arr[maxx]){

maxx = r;

}

if (maxx != i){

swap(arr[i], arr[maxx]);

heapify(arr, size, maxx);

}

}

int main(int argc, char\* argv[]){

ifstream inputFile(argv[1]);

if (!inputFile) {

cerr << "Не удалось открыть файл!" << endl;

return 1;

}

string line;

if (getline(inputFile, line)) {

stringstream ss(line);

string cell;

int count = 0;

stringstream temp(line);

while (getline(temp, cell, '\t')) {

count++;

}

int arr[count];

int index = 0;

while (getline(ss, cell, '\t')) {

arr[index++] = stoi(cell);

}

inputFile.close();

clock\_t begin = clock();

int size = sizeof(arr) / sizeof(arr[0]);

for (int i = size/2-1; i >= 0; i--){

heapify(arr, size, i);

}

for (int i = size-1; i >= 0; i--){

swap(arr[0], arr[i]);

heapify(arr, i, 0);

}

clock\_t end = clock();

double elapsed = (double) (end - begin) / CLOCKS\_PER\_SEC;

cout << elapsed << endl;

//for (int i = 0; i < size; i++){

// cout << arr[i] << " ";

// }

}

}

ПРИЛОЖЕНИЕ С

#include <iostream>

#include <fstream>

#include <sstream>

#include <ctime>

using namespace std;

int main(int argc, char\* argv[]){

ifstream inputFile(argv[1]);

if (!inputFile) {

cerr << "Не удалось открыть файл!" << endl;

return 1;

}

string line;

vector<int> arr;

if (getline(inputFile, line)) {

stringstream ss(line);

string cell;

while (getline(ss, cell, '\t')) {

arr.push\_back(stoi(cell));

}

}

inputFile.close();

clock\_t begin = clock();

int sizee = arr.size();

int maximum = 0;

for (int i = 0; i < sizee; i++) {

maximum = max(arr[i], maximum);

}

vector<int> pref\_summa(maximum + 1, 0);

for (int i = 0; i < sizee; i++) {

pref\_summa[arr[i]]++;

}

for (int i = 1; i <= maximum; i++) {

pref\_summa[i] += pref\_summa[i - 1];

}

vector<int> answer(sizee);

for (int i = 0; i < sizee; i++) {

answer[pref\_summa[arr[i]] - 1] = arr[i];

pref\_summa[arr[i]]--;

}

clock\_t end = clock();

double time\_spent = (double) (end - begin) / CLOCKS\_PER\_SEC;

cout << time\_spent << endl;

//for (int i = 0; i < sizee; i++) {

// cout << answer[i] << " ";

// }

}