БУ ВО Ханты-Мансийского автономного округа – Югры

«Сургутский государственный университет»

Политехнический институт

Кафедра автоматики и компьютерных систем

**ОТЧЁТ**

по лабораторной работе № 2

по дисциплине: Алгоритмы и структуры данных

тема: «Алгоритмы сортировки»

Выполнила: студентка группы №609-11,

Пашина Илона Руслановна

Принял: доцент кафедры АиКС

Даниленко И.Н.

Сургут

2023г.

**Цель работы:**

Изучение алгоритмов сортировки, исследование сложности алгоритмов сортировки при различных исходных данных, закрепление навыков алгоритмизации.

**Задание:**

1. В таблице №1 представлены алгоритмы сортировки (и в некоторых случаях их параметры), которые необходимо реализовать в соответствии с вариантом (таблица №2). Для проверки реализованных функций сортировки разработать функцию, проверяющую упорядоченность элементов в последовательности.
2. Разработать функции, позволяющие производить оценку временной сложности алгоритма T(n) (n – количество элементов сортируемой последовательности) и суммарного количества операций сравнения S(n), выполняемых в ходе его работы.
3. На основе функций формирования последовательностей, реализованных входе предыдущей лабораторной работы, и результатов выполнения пп. 1 и 2, разработать программу, с помощью которой по экспериментальным данным построить зависимости T(n) и S(n) для заданных алгоритмов сортировки по четырём последовательностям (упорядоченная, случайная, упорядоченная в обратном порядке и указанная в варианте). Размеры сортируемых последовательностей следует выбирать самостоятельно в соответствии с характеристиками исследуемого алгоритма. Кроме того, необходимо оценить зависимости T(n) и S(n) для функции qsort из стандартной библиотеки С.
4. Составить отчёт, в котором привести графики T(n) и S(n), результаты анализа полученных экспериментальных данных и теоретических оценок сложности алгоритмов, сравнительную оценку реализованных алгоритмов (по сложности, устойчивости, естественности, требованиям к памяти), выводы по работе.

**1 вариант.**

Алгоритмы сортировки:

* 9 - Модифицированная быстрая сортировака 2 (Выбор разделяющего элемента с помощью медианы)
* 11 - сортировка вставками
* 12 - Модифицированная сортировка вставками 1 (С использованием сигнального ключа)
* 22 - Сортировка подсчётом

4 функция: ступенчатая

Тип данных: int

**Ход работы**

**Таблицы**

Сортировка вставками.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Рандомная | | Упорядоченная | | Упрядоч. В обратном | | Ступенчатая | |
|  | Время | Сравнения | Время | Сравнения | Время | Сравнения | Время | Сравнения |
| 5000 | 31 | 6221189 | 0 | 4999 | 31 | 12379900 | 15 | 1189305 |
| 10000 | 62 | 24966171 | 0 | 9999 | 125 | 49509900 | 16 | 4838319 |
| 15000 | 141 | 56282896 | 0 | 14999 | 297 | 111389900 | 47 | 10683063 |
| 20000 | 266 | 99985278 | 0 | 19999 | 516 | 198019900 | 47 | 19110680 |
| 25000 | 406 | 155642477 | 0 | 24999 | 781 | 309399900 | 78 | 29695817 |
| 30000 | 594 | 223907258 | 0 | 29999 | 1000 | 445529900 | 109 | 42659607 |
| 35000 | 781 | 305530262 | 0 | 34999 | 1266 | 606409900 | 157 | 57862752 |
| 40000 | 860 | 400156697 | 0 | 39999 | 1672 | 792039900 | 203 | 75634825 |
| 45000 | 1062 | 506494422 | 0 | 44999 | 2093 | 1002419900 | 250 | 96953019 |
| 50000 | 1359 | 625777032 | 0 | 49999 | 2610 | 1237549900 | 312 | 118801423 |

Модифицированная сортировка вставками.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Рандомная | | Упорядоченная | | Упрядоч. В обратном | | Ступенчатая | |
|  | Время | Сравнения | Время | Сравнения | Время | Сравнения | Время | Сравнения |
| 5000 | 16 | 6171475 | 0 | 9997 | 47 | 12380047 | 0 | 1189561 |
| 10000 | 47 | 24702110 | 0 | 19997 | 94 | 49510097 | 16 | 4838839 |
| 15000 | 140 | 55998400 | 0 | 29997 | 234 | 111390147 | 15 | 10683821 |
| 20000 | 235 | 98947696 | 0 | 39997 | 469 | 198020197 | 47 | 19111730 |
| 25000 | 375 | 155471601 | 0 | 49997 | 766 | 309400247 | 63 | 29697096 |
| 30000 | 547 | 222537787 | 0 | 59997 | 1015 | 445530297 | 94 | 42661125 |
| 35000 | 734 | 303606338 | 0 | 69997 | 1203 | 606410347 | 109 | 57864534 |
| 40000 | 797 | 396731439 | 0 | 79997 | 1563 | 792040397 | 156 | 75636745 |
| 45000 | 985 | 502065537 | 0 | 89997 | 1969 | 1002420447 | 219 | 96955260 |
| 50000 | 1234 | 615342745 | 0 | 99997 | 2484 | 1237550497 | 297 | 118803911 |

Быстрая сортировка.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Рандомная | | Упорядоченная | | Упрядоч. В обратном | | Ступенчатая | |
|  | Время | Сравнения | Время | Сравнения | Время | Сравнения | Время | Сравнения |
| 500000 | 31 | 13239493 | 47 | 14112740 | 47 | 15357980 | 31 | 15543056 |
| 1000000 | 94 | 26777212 | 78 | 27658067 | 78 | 26766276 | 94 | 32920936 |
| 1500000 | 125 | 41880311 | 141 | 41172814 | 109 | 40534168 | 141 | 49436710 |
| 2000000 | 171 | 58945811 | 171 | 54533971 | 141 | 53688710 | 188 | 71083816 |
| 2500000 | 218 | 69738552 | 219 | 68576680 | 203 | 67936046 | 219 | 83587447 |
| 3000000 | 265 | 85846828 | 265 | 81936665 | 281 | 89487774 | 265 | 101159651 |
| 3500000 | 297 | 96909886 | 328 | 104921650 | 312 | 95579242 | 313 | 125481696 |
| 4000000 | 343 | 114393690 | 375 | 112088311 | 359 | 108742373 | 375 | 146641623 |
| 4500000 | 390 | 133708837 | 422 | 131709294 | 406 | 125711356 | 407 | 159491502 |
| 5000000 | 437 | 146165588 | 484 | 144286465 | 469 | 143097905 | 484 | 190515447 |

Сортировка подсчётом.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Рандомная | | Упорядоченная | | Упрядоч. В обратном | | Ступенчатая | |
|  | Время | Сравнения | Время | Сравнения | Время | Сравнения | Время | Сравнения |
| 20000000 | 78 | 0 | 63 | 0 | 78 | 0 | 62 | 0 |
| 40000000 | 140 | 0 | 125 | 0 | 140 | 0 | 141 | 0 |
| 60000000 | 203 | 0 | 203 | 0 | 203 | 0 | 218 | 0 |
| 80000000 | 313 | 0 | 282 | 0 | 296 | 0 | 281 | 0 |
| 100000000 | 359 | 0 | 344 | 0 | 359 | 0 | 391 | 0 |
| 120000000 | 453 | 0 | 437 | 0 | 422 | 0 | 422 | 0 |
| 140000000 | 515 | 0 | 531 | 0 | 484 | 0 | 500 | 0 |
| 160000000 | 593 | 0 | 562 | 0 | 562 | 0 | 578 | 0 |
| 180000000 | 656 | 0 | 640 | 0 | 656 | 0 | 656 | 0 |
| 200000000 | 750 | 0 | 703 | 0 | 687 | 0 | 719 | 0 |

Qsort – встроенная функция сортировки.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Рандомная | | Упорядоченная | | Упрядоч. В обратном | | Ступенчатая | |
|  | Время | Сравнения | Время | Сравнения | Время | Сравнения | Время | Сравнения |
| 1000000 | 47 | 9210092 | 31 | 6800062 | 16 | 6800061 | 31 | 8828515 |
| 2 000 000 | 94 | 20103984 | 31 | 13600062 | 16 | 13600061 | 69 | 18042849 |
| 3 000 000 | 141 | 27662496 | 47 | 20400062 | 47 | 20400061 | 110 | 27266217 |
| 4 000 000 | 188 | 37958269 | 62 | 27200062 | 63 | 27200061 | 141 | 35641499 |
| 5 000 000 | 234 | 48047747 | 78 | 34000062 | 63 | 34000061 | 172 | 44303051 |
| 6 000 000 | 281 | 58328512 | 93 | 40800062 | 93 | 40800061 | 219 | 52624336 |
| 7 000 000 | 328 | 66564280 | 109 | 47600062 | 125 | 47600061 | 250 | 63976445 |
| 8 000 000 | 375 | 78558169 | 141 | 54400062 | 140 | 54400061 | 297 | 72715242 |
| 9 000 000 | 453 | 84142666 | 141 | 61200062 | 156 | 61200061 | 312 | 80729382 |
| 10 000 000 | 469 | 98203961 | 156 | 68000062 | 172 | 68000061 | 344 | 91001017 |

**Графики по таблицам.**

1 ряд – рандомная

2 ряд – Упорядоченная

3 ряд – Упорядоченная в обратном порядке

4 ряд – Ступенчатая

В данном случае ожидалось, что график будет похож на график алгоритма сортировки “qsort”.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Алгоритм Сортировки | Худший случай | Средний случай | Лучший случай | Естественность | Устойчивость |
| Сортировка вставками |  |  |  | + | + |
| Мод. Сортировка вставками |  |  |  | + | + |
| Быстрая сортировка  (медианой) |  |  |  | - | - |
| Сортировка подсчётом |  |  |  | - | + |
| qsort |  |  |  | - | - |

В ходе лабораторной работы путём исследования были изучены методы сортировки. Самыми быстрыми оказались: сортировка подсчётом и быстрая сортировка (qsort и с медианой). Они за приблизительно то же время сортировали массив больший в 100-4000 раз.

**Вывод:**

Изучила алгоритмы сортировки, исследовала сложности алгоритмов сортировки при различных исходных данных, закрепила навыки алгоритмизации.

**Листинг**

#include <iostream>

#include <cstdlib>

#include <windows.h>

using namespace std;

void print\_array(int \*A, int N);

void FillRand(int \*A, int N, int min, int max, int K);

void FillUp(int \*A, int N, int min, int max, int K);

void FillDown(int \*A, int N, int min, int max, int K);

void FillStep(int \*A, int N, int min, int max, int K);

int CheckSort(int \*A, int N);

void InsertionSort(int \*A, int N);

void InsertionSortKey(int \*A, int N);

void CountingSort(int \*A, int N);

int partition(int \*A, int start, int end);

void quicksort(int \*A, int start, int end);

int Less(int A, int B);

long long Cmp;

int compare(const void \*value, const void \*value2){

int a = \*(int\*)value;

int b = \*(int\*)value2;

Cmp++;

if(a > b) return 1;

if(a < b) return -1;

return 0;

}

int main() {

int max = 100000, min=0;

int N = max;

int s = 5;

double t;

int count = 0;

int Nmax = 200000000;

int \*A;

A = new int[Nmax];

cout << endl;

for(N = Nmax/10; N <= Nmax; N+=Nmax/10){

Cmp = 0;

FillRand(A, N, 0, 100, s);

t = GetTickCount();

CountingSort(A, N);

cout << GetTickCount() - t << " "<< Cmp << endl;

}

cout << Cmp << endl;

cout << endl;

for(N = Nmax/10; N <= Nmax; N+=Nmax/10){

Cmp = 0;

FillUp(A, N, 0, 100, s);

t = GetTickCount();

CountingSort(A, N);

cout << GetTickCount() - t << " "<< Cmp << endl;

}

cout << endl;

cout << endl;

for(N = Nmax/10; N <= Nmax; N+=Nmax/10){

Cmp = 0;

FillDown(A, N, 0, 100, s);

t = GetTickCount();

CountingSort(A, N);

cout << GetTickCount() - t << " "<< Cmp << endl;

}

cout << endl;

cout << endl;

for(N = Nmax/10; N <= Nmax; N+=Nmax/10){

Cmp = 0;

FillStep(A, N, 0, 100, s);

t = GetTickCount();

CountingSort(A, N);

cout << GetTickCount() - t << " "<< Cmp << endl;

}

delete[] A;

return 0;

}

void print\_array(int \*A, int N){

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

cout << A[i] << ' ';

}

cout << endl;

}

void FillRand(int \*A, int N, int min, int max, int K){

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

A[i] = ((rand() << 15) | rand()) % (max - min) + min;

}

}

void FillUp(int \*A, int N, int min, int max, int K){

double delta = ((double)(max - min))/N;

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

A[i] = delta\*i+min;

}

}

void FillDown(int \*A, int N, int min, int max, int K){

double delta = ((double)(max - min))/N;

for(int i = N-1, j = 0; i >=0; i--, j++){

A[j] = delta\*i+min;

}

}

void FillStep(int \*A, int N, int min, int max, int K){ // Ñòóïåí÷àòàÿ

int step = N/K;

int stepbegin = 0;

int mins = min;

int maxs = max/K;

for(int j = 0; j < K; j++){

for(int i = stepbegin; i < stepbegin+step; i++){

A[i] = rand() % (maxs - mins) + mins;

}

mins = maxs;

maxs += max/K;

stepbegin += step;

}

}

int CheckSort(int \*A, int N){

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

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

return 1;

}

}

return 0;

}

int Less(int A, int B) {

Cmp++;

return A < B;

}

void InsertionSort(int \*A, int N){

int V;

int j;

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

V = A[i];

for(j = i; j > 0 && Less(V, A[j-1]); j--){

A[j] = A[j-1];

}

A[j] = V;

}

}

void InsertionSortKey(int \*A, int N){

int V;

int j;

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

if(Less(A[i], A[i-1]))

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

}

for(int i = 2; i < N; i++){

V = A[i];

for(j = i; Less(V, A[j-1]); j--){

A[j] = A[j-1];

}

A[j] = V;

}

}

void CountingSort(int \*A, int N){

int \*Cnt, \*AH, D = 101;

AH = new int[N];

Cnt = new int[D];

for(int i = 0; i < D; i++) Cnt[i] = 0;

for(int i = 0; i < N; i++) Cnt[A[i]+1]++;

for(int i = 1; i < D; i++) Cnt[i] += Cnt[i-1];

for(int i = 0; i < N; i++) AH[Cnt[A[i]]++] = A[i];

for(int i = 0; i < N; i++) A[i] = AH[i];

free(AH);

free(Cnt);

}

int partition(int \*A, int start, int end)

{

int V = A[end];

int i = start-1;

int j = end;

for(;;){

while(Less(A[++i], V));

while(Less(V, A[--j])) if(j==start) break;

if(i >=j) break;

swap(A[i], A[j]);

}

swap(A[end], A[i]);

return i;

}

void quicksort(int \*A, int start, int end)

{

int P;

if (end - start <= 0) return;

swap(A[(start+end)/2], A[end-1]);

if(Less(A[end-1], A[start])) swap(A[start], A[end-1]);

if(Less(A[end], A[start])) swap(A[start], A[end]);

if(Less(A[end], A[end-1])) swap(A[end-1], A[end]);

P = partition(A, start, end);

quicksort(A, start, P-1);

quicksort(A, P+1, end);

}