МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«Харківський Політехнічний Інститут»

Кафедра «Стратегічне управління»

Звіт з лабораторної роботи №4

«Алгоритми сортування та асимптотичний аналіз алгоритмів»

з дисципліни

"Алгоритми та структури даних"

Варіант № 6

Перевірив:

ст. викл. каф. стратегічного

управління Мошко Є.О.

Виконав: Колій Дмитро

ст. гр. КН-320б

Харків – 2022

Мета: вивчити основні алгоритми сортування масивів і освоїти їх на практиці. Перевірити роботу алгоритмів на різних наборах даних, провести асимптотичний аналіз алгоритмів сортування.

Індивідуальне завдання:

1. Створити функцію сортування масиву алгоритмами сортування, які треба обрати у таблиці 3 відповідно до свого варіанту. Варіант завдання вибирається за номером студента в списку групи.

2. Будь-яким способом заповнити елементи масиву значеннями.

3. Виконати сортування масиву першим алгоритмом і проаналізувати отримані результати.

4. Перевірити всі варіанти вихідного заповнення масиву: випадковим чином, відсортованого у порядку зростання, відсортованого у порядку спадання. Переконатися в правильності сортування у всіх випадках. Зробити висновки.

5. Повторити пункти 2-4 для другого та третьего алгоритму сортування.

6. Дослідити складність алгоритмів. Провести асимптотичний аналіз алгоритмів сортування та зробити висновки.

7. Здійснити порівняння алгоритмів сортування для n = 10, 100, 1000, 10000 і в наступному порядку вхідних елементів:

− елементи вже впорядковані;

− елементи в зворотному порядку;

− розстановка елементів випадкова.

8. Виконання порівняння алгоритмів сортування на основі наступних кроків:

− створити таблицю асимптотичних оцінок трудомісткості алгоритмів в кращому, середньому, гіршому випадках;

− розставити лічильники операцій у функціях угруповань;

− провести експеримент, визначити середню кількість операцій для різних угруповань, побудувати графіки;

− для кожного виклику сортування генерувати новий масив;

− створити таблиці і представити графіки експериментальних оцінок алгоритмів.

Необхідно враховувати, що порівняння алгоритмів проводиться на однакових вхідних даних. Зберегти файл з тестом програми для подальших робіт.

Варіант завдання вибирається за номером студента в списку групи:

|  |  |
| --- | --- |
| № варіанту | Дані |
| 6 | 1. Алгоритм підрахунком  2. Алгоритм пірамідального сортування  3. Алгоритм сортування Шелла |

Сортування підрахунком



Рисунок 1 – Блок-схема для сортування підрахунком

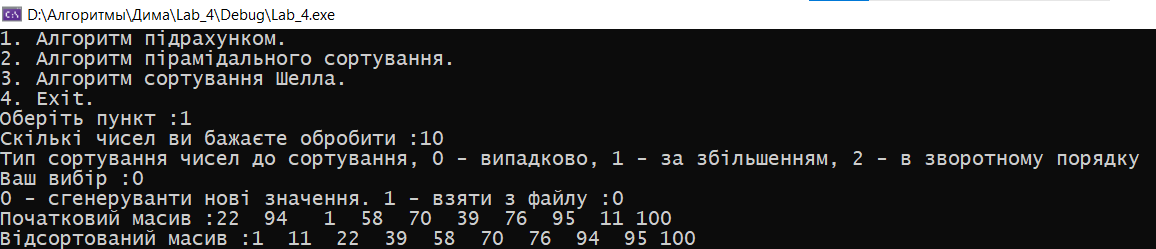


Рисунок 2 – Результат підрахункового сортування

Програмна реалізація підрахункового сортування:

int sort\_count(vector<int> arr) {

int max = \*max\_element(arr.begin(), arr.end());

int\* c = new int[++max] {0};

for (int i = 0; i < arr.size(); i++)

{

c[arr[i]] = ++c[arr[i]];

}

int b(0);

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

{

for (int j = 0; j < c[i]; j++)

{

arr[b++] = i;

}

}

cout << "Відсортований масив :";

print\_mass(arr);

return 0;

}

Сортування пірамідально

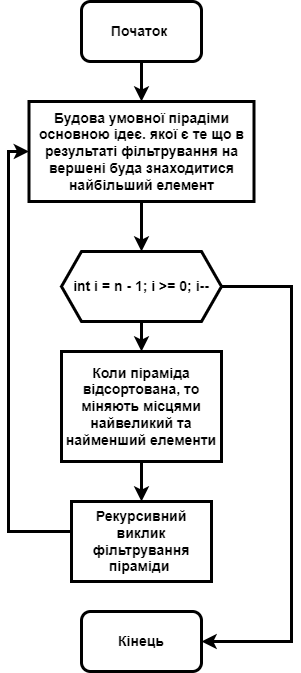


Рисунок 3 – Блок-схема для пірамідального сортування

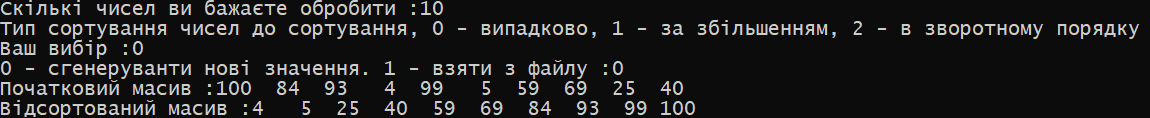


Рисунок 4 – Результат пірамідального сортування

Програмна реалізація пірамідального сортування:

void heapify(vector<int>& arr, int n , int i)

{

int largest = i;

int l = 2 \* i + 1;

int r = 2 \* i + 2;

if (l < n && arr[l] > arr[largest])

largest = l;

if (r < n && arr[r] > arr[largest])

largest = r;

if (largest != i)

{

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

heapify(arr, n, largest);

}

}

int heapSort(vector<int> arr)

{

int n = arr.size();

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

heapify(arr, n, i);

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

{

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

heapify(arr, i, 0);

}

cout << "Відсортований масив :";

print\_mass(arr);

return 0;

}

int sort\_count(vector<int> arr) {

int max = \*max\_element(arr.begin(), arr.end());

int\* c = new int[++max] {0};

for (int i = 0; i < arr.size(); i++)

{

c[arr[i]] = ++c[arr[i]];

}

int b(0);

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

{

for (int j = 0; j < c[i]; j++)

{

arr[b++] = i;

}

}

cout << "Відсортований масив :";

print\_mass(arr);

return 0;

}

Сортування Шелла

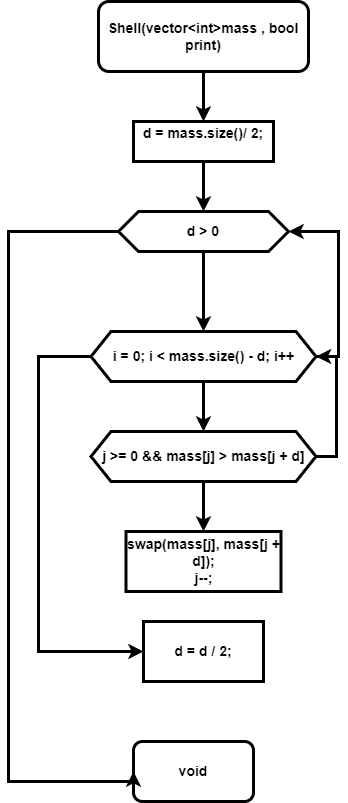


Рисунок 5 – Алгоритм сортування Шелла

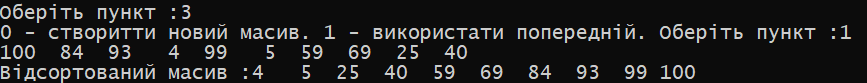


Рисунок 6 – Результат сортування Шелла

Програмна реалізація сортування Шелла:

int Shell(vector<int>mass) //сортировка Шелла

{

int i, j, d, per\_for\_count\_iter(0);

d = mass.size() / 2;

while (per\_for\_count\_iter++, d > 0)

{

for (i = 0; i < mass.size() - d; i++)

{

j = i;

per\_for\_count\_iter += 2;

while (per\_for\_count\_iter += 2, j >= 0 && mass[j] > mass[j + d])

{

swap(mass[j], mass[j + d]);

j--;

per\_for\_count\_iter += 4;

}

}

per\_for\_count\_iter += 2; // перед выходом из for произошла проверка на выход

d = d / 2;

}

per\_for\_count\_iter += 2;

cout << "Відсортований масив :";

print\_mass(mass);

return per\_for\_count\_iter;

}

Аналіз алгоритму підрахункового сортування

Для аналізу швидкості алгоритму використовується функція, яка називається тимчасова складність алгоритму ( T(n) ). Тимчасова складність алгоритму підрахункового сортування в найгіршому випадку виглядає наступним чином:

n – кількість елементів; k – діапазон елементів

Таким чином найгірший випадок – . Найкращий випадок – Середній випадок – .

Аналіз алгоритму пірамідального сортування

Для пірамідального сортування тимчасова складність алгоритму T(n) в найгіршому випадку виглядає наступним чином:

Таким чином найгірший випадок –. Найкращий випадок – . Середній випадок – .

Аналіз алгоритму сортування Шелла

Для сортування Шелла тимчасова складність алгоритму в найгіршому випадку виглядає наступним чином:

Таким чином найгірший випадок – . Найкращий випадок – . В середньому випадку – .

Після аналізу всіх алгоритмів сортування, які були наведені вище, приведено результати їх аналізу в табл. 1 в якій позначено оцінки асимптотичної складності. В даній таблиці зеленим кольором позначено ту тимчасову складність алгоритму сортування в якій алгоритм сортування впорався з поставленою задачею за найкоротший час, а рожевим кольором позначено ту тимчасову складність алгоритму в якій алгоритм справляться з задачею за найгірший час.

Таблиця 1 – Оцінки асимптотичної складності для всіх алгоритмів сортування



Після побудови таблиці в якій наведено результати оцінки асимптотичної складності для всіх алгоритмів сортування необхідно перевірити роботу даних видів сортувань.

Насамперед було виконано перевірку роботи видів сортувань на масиві, який вже був впорядкований по збільшенню. Для цього було виконано тестування даних видів сортувань на масивах розміром 10, 100, 1000, 10000 елементів. В табл. 2 наведено результати роботи алгоритмів сортувань з впорядкованим масивом по збільшенню в якій наведено скільки операцій потрібно для того чи іншого алгоритму, щоб виконати сортування певної кількості елементів масиву.

Таблиця 2 – Результати алгоритмів сортування при сортуванні впорядкованого масиву по збільшенню.



Наступною перевіркою була перевірка роботи даних алгоритмів сортувань на сортуванні масиву, який був впорядкований по спаданню.

Результати наведені в табл. 3.

Таблиця 3 – Результати алгоритмів сортування при сортуванні впорядкованого масиву по спаданню



Таблиця 4 – Результат алгоритмів сортування при сортуванні рандомного масиву.



Після проведення перевірок над алгоритмами сортування було побудовано табл. 5 в якій наведено середні значення операцій для кожного алгоритму сортування.

Таблиця 5 – Середні значення операцій для алгоритмів сортування



На основі табл. 5 було побудовано графік, котрий уособлює асимптотичний аналіз алгоритмів сортування.



Отже після проведеного асимптотичного аналізу алгоритмів сортування було досліджено, що при невеличних об’ємів масиву краще впорюються пірамідальна та Шелла сортування. При збільшенні елементів підрахункове сортування є статистичною і не змінюються, одночасно з цим пірамідальна гірша ніж Шелла. Але при великих об’ємів елементів Шелла програє пірамідальній.

Висновок: Під час виконання даної лабораторної роботи були отриманні навички роботи з алгоритмами сортування та проведення асимптотичного аналізу алгоритмів. Я ознайомився із основними алгоритмами сортування масивів і освоїв їх на практиці. Закріпив навички перевірки роботи алгоритмів на різних наборах даних та проводити асимптотичний аналіз алгоритмів сортування.

Алгоритм сортування підрахунком – це алгоритм сортування, який використовую діапазон чисел масива, який сортують за для підрахунку схожих елементів.

Алгоритм пірамідального сортування – це покращена версія алгоритму сортування вибором: він ділить вихідні дані на відсортовані та невідсортовані області, потім покроково зменшує невідсортовану область, витягуючи надвеликий елемент і пересуваючи його до відсортованої області.

Сортування Шелла – один з різновидів сортування, основна ідея якого полягає в порівнюванні елементів масиву розташованих на деякому кроці один від одного. Потім крок стає все меншим і меншим та в кінці виконання сортування буде виконане прохід по масиву з кроком один.