Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання №4

“Сортування з двома характеристиками”

Виконав студент 2-го курсу

Групи К-28

Гуща Дмитро Сергійович

2020

**Завдання**

Нехай маємо масив, що містить n записів з даними для сортування, і що ключ кожного запису приймає значення 0 або 1. Алгоритм для сортування такого набору записів повинен мати деякі з трьох наступних характеристик: 1) час роботи алгоритму О(n); 2) алгоритм має бути стійким; 3) сортування проводиться на місці, тобто крім вихідного масиву використовується додаткова пам’ять, що не перевищує деякої постійної величини. Розробіть і реалізуйте алгоритм, що задовольняє

1. критеріям 1 і 2
2. критеріям 1 і 3
3. критеріям 2 і 3 (бажано з найкращим часом)

**Теорія**

Для реалізації алгоритму для критеріїв 1 і 2 візьмемо алгоритм сортування підрахунком, в якому не порівнюються вхідні дані між собою, а отже він є стійким і праює за лінійний часа, при його виконанні використовується додаткова пам’ять. Всього є 2 ключа, звідси алгоритм можно модифікувати таким: спочатку у допоміжний масив записуються усі елементи з першим ключем а потім записати усі елементи з другим ключем новоутворений масив і буде результатом алгоритму.

Для критеріїв 1,3 можно взяти алгоритм вставками але модифікувати його, тому-шо значень ключів всього два, то можно просто запам’ятовувати останній елемент з другим ключем, а потім якщо після нього буде елемент з ключем 1 то міняємо їх місцями, це виконується за одне проходження масиву тобто складність буде лінійною, додаткова пам’ять не виділяється, а значить сортування проводиться на місті і алгоритм не є стійким бо можливий обмін місцями елементів з однаковими ключами

Для критеріїв 2,3 можно взяти алгоритм сортування бульбашкою в якому ми виділимо всього одну змінну, цей алгоритм є стійким

**Алгоритм 1( критерії 1 і 2)**

• проходимо весь масив та шукаємо 1-ий ключ, записуємо його у допоміжний масив, якщо 2-ий то ігноруємо його.

•також само як і в минулому пункті, тільки потрібно 2 ключа змінити місцями.

Допоміжний масив і є результатом роботи алгоритму

**Алгоритм 2 ( критерії 1 і 3)**

• проходимо по масиву, якщо елемент який в даний момент зчитується містить ключ – 0 то міняємо його місцями з елементом який має індекс P. Якщо ж ключ елемента – 1 то ігноруємо його.

**Алгоритм 3 (критерії 2 і 3 )**

• проходимо вхідний масив : якщо наступний елемент строго менший за елемент ,який в даний момент читається, то міняємо їх місцями якщо в іншому випадку йдем далі по масиву

• якщо в кроці 1 ні разу елементи не мінялися місцями то значить масив відсортовано і алгоритм завершив свою роботу .якщо ж все таки елементи мінялися місцями то повторюємо крок 1

**Складність**

**Алгоритм 1( критерії 1 і 2)**

Лінійна складність. У алгоритмі виділяється додаткова пам'ять яка є прямопропорційною кількості елементів у вхідному масиві тобто просторова складність



**Алгоритм 2( критерії 1 і 3)**

Лінійна складність. У алгоритмі виділяється додаткова пам'ять яка рівна одній змінній тобто просторова складність



**Алгоритм 3( критерії 2 і 3)**

Найгірший випадок – спадаючий масив, Тоді складність алгоритму– квадратична тобто складність у найкращому випадку – коли масив уже відсортований буде достатньо пройти його лише раз тобто складність просторова складність : виділяється всього лише одна змінна для обміну тобто просторова складність рівна



**Мова програмування**

С++

**Модулі програми**

void first\_sort(vector<int> &a,int n)  
//виконує сортування з критеріями 1 і 2  
void second\_sort(vector<pair<int,char>> &a)  
//поля ключ і якесь значення ця фунція виконує сортування з критеріями 2 і 3  
void third\_sort(vector<pair<int,char>> &a,int n)  
//сортування з критеріями 1 і 3  
void copy(vector<pair<int,char>> &a, vector <pair<int, char>> b)  
//допоміжна функція яка копіює елементи масиву b у масив a  
void print (vector<pair<int , char>> a)  
//допоміжна функція яка виводить на екран масив

**Інтерфейс користувача**

Вхідні дані випадковим чином генеруються у програмі а результати алгоритмів виводяться на консоль

**Тестові приклади**

|  |  |
| --- | --- |
| input | output |
| 0 0 1 1 0 1 0 1 1 1  C U С Е < 0 Б E Е А | Sort wich characteristics  ( O(n) and stable)  0 0 0 0 1 1 1 1 1 1  C U < Б С Е 0 E Е А  Sort wich characteristics  ( O(n) and additional memory)  0 0 0 0 1 1 1 1 1 1  C U < Б С Е 0 E Е А  Sort wich characteristics  (stable and O(1) additional memory)  0 0 0 0 1 1 1 1 1 1  C U < Б С 0 Е E Е А |

Нехай маємо масив ключів {0 0 1 1 0 1 0 1 1 1}

first\_sort:

виділимо пустий масив {} за перше проходження у нього ми запишемо

{0 0 0 0} за друге проходження масиву у допоміжний масив запишеться

{1 1 1 1 1 1 } і отже допоміжний масив матиме вигляд {0 0 0 0 1 1 1 1 1 1} що і буде результатом роботи функції

second\_sort:

за перше проходження масиву він набере вигляду {0 0 1 1 0 0 1 1 1 1 } оскільки здійснювалися зміни позицій елементів то повторюємо крок 1

за друге проходження масив буде {0 0 1 0 0 1 1 1 1 1} знову повторюємо крок 1 бо були зміни позицій далі масив набуде вигляду {0 0 0 0 1 1 1 1 1 1}

змін позицій не відбувалося то функція завершує свою роботу

third\_sort:

Нехай змінна р = 0 тоді йдемо по масиву 1 елемент - 1 ігноруємо його другий елемент -0 отже міняємо його з елементом з р –індексом збільшуємо р на 1

Масив має вигляд {0 1 0 0 0 1 1 1 1 1 } третій елемент -1 отже ігноруємо його, 4 елемент – 0 отже міняємо його з елементом з р –індексом збільшуємо

остаточний масив – {0 0 0 0 1 1 1 1 1 1}

**Висновки**

Так як у масиві всього два ключа то алгоритм з критеріями 2 і 3 не буде досягати складності складність буде десь посередині між залежить від співідношення розміщення ключів на початку масиву .Решта алгоритмів мають лінійну складність



**Література**

* Лекція №6
* <https://uk.wikipedia.org/wiki/%D0%A1%D0%BE%D1%80%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D1%83%D0%BB%D1%8C%D0%B1%D0%B0%D1%88%D0%BA%D0%BE%D1%8E>
* <https://studfile.net/preview/3741387/>
* <http://delphi.dp.ua/sortuvannya-shlyahom-obminu/>