Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Завдання №4

Виконав студент 2-го курсу

Групи К-28

Самойлич Євгеній Костянтинович

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**

Алгоритм, що відповідає критеріям 1 і 2 дуже схожий на сортування підрахунком, а так як у нас будуть ключі лише 2 значень, то ми можемо спочатку записати усі елементи зі значенням ключа 0, у додатковий масив, а потім туди ж дописати усі елементи зі значенням ключа 1.

Для реалізація алгоритму який задовольняв би критеріям 1 і 3 візьмемо за основу алгоритм вставками і модифікуємо його під наші потреби. Побудуєм його наступним чином : оскільки значень ключів усього два, то можна просто запам’ятовувати останній елемент з ключем 1 а потім якщо після нього буде елемент з ключем 0 міняти їх місцями. Додаткова пам'ять не виділяється тобто сортування проводиться на місці

Для реалізація алгоритму який задовольняв би критеріям 2 і 3 можна взяти алгоритм сортування бульбашкою, в якому виділяється всього одна змінна (сортування проводиться на місці) та цей алгоритм є стійким (однакові елементи не міняються місцями )

**Алгоритм 1**( критерії 1 і 2)

• пройти весь масив якщо елемент який в даний момент читається містить ключ 0 то записуємо його у допоміжний масив якщо 1 то ігноруємо його

• пройти знову весь масив якщо елемент який в даний момент читається містить ключ 1 то записуємо його у допоміжний масив якщо 1–ий то ігноруємо його

Допоміжний масив і є результатом роботи алгоритму

**Алгоритм 2** ( критерії 1 і 3)

• пройти весь масив: якщо елемент який в даний момент читається містить ключ - 0 то міняємо його місцями з елементом який має індекс p тобто або з самим собою або елементом з ключем - 1 який стоїть перед ним та збільшуємо змінну p щоб пам’ятати де стоїть останній елемент з яким буде мінятися наступний елемент з ключем 0. Якщо ж ключ елемента рівний 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 жодного разу елементи не мінялися місцями, це 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)

Часова складність лінійна() у алгоритмі також використовується додаткової пам’яті.



Алгоритм 2 (критерії 1 і 3)

Часова складність лінійна() у алгоритмі також використовується О(1) додаткової пам’яті.



Алгоритм 3 (критерії 2 і 3)

Найгірший випадок – коли масив розташований у порядку спадання, Тоді складність алгоритму (*n-1)* + (*n*-2) + (*n* – 3) + … + 2 = *n*(*n*-2)/2 – квадратична тобто складність . У найкращому випадку – масив уже відсортований буде достатньо пройти його лише раз тобто складність .



**Мова програмування**

С++

**Модулі програми**

* void sort1(vector <pair <int, char>>& a)

Функція сортування за критеріями 1 і 2

* void sort2(vector <pair <int, char>>& a)

Функція сортування за критеріями 2 і 3

* void sort3(vector <pair <int, char>>& a)

Функція сортування за критеріями 1 і 3

**Інтерфейс користувача**

Вхідний масив даних та ключей формується випадковим чином у програмі, у консоль виводиться відсортований масив.

**Тестові приклади**

Input:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| \_ | 1 | > | ~ | O | P | O | a | P | C | c | Z | 8 | X | 6 | М | e | s | Q | i |

first sort output:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| \_ | O | P | O | a | P | C | c | 6 | М | e | Q | i | 1 | > | ~ | Z | 8 | X | s |

second sort output:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| \_ | O | P | O | a | P | C | c | 6 | М | e | Q | i | X | > | ~ | 1 | s | Z | 8 |

third sort output:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| \_ | O | P | O | a | P | C | c | 6 | М | e | Q | i | 1 | > | ~ | Z | 8 | X | s |

**Висновки**

Перший та другий алгоритми мають лінійну складність, а третій в найгіршому випадку.



**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/