ЛЕКЦІЯ 4

МЕТОДИ СОРТУВАННЯ

(продовження)

4.5. Сортування включенням

При сортуванні включенням до відсортованої множини *R* кожний раз приєднується один елемент, а саме: із невідсортованої вхідної множини *M*вибирається довільний елемент і розміщується у вихідну множину R .

Алгоритм *V.*

Нехай задано множину елементів M, $|M|=n$.

V1. k =1*;* повторювати кроки *V2, V3*, доки k<n.

V2. Вибір довільного елемента *x* вхідної множини $M : x=M(k)$.

V3. Розміщення *x* у вихідній множині R так, щоб вона залишалась відсортованою.

V4. Кінець. Вихід.

Вихідну множину R при кожному включенні можна відсортовувати відомим методом сортування, наприклад, методом простої вибірки. Майже всі методи сортування включенням у найгіршому випадку вимагають порядку n^2 порівнянь, тому їх застосування пов'язане з деяким ризиком. Є багато варіантів цього методу сортування.

Розглянемо **метод Шелла** . Його суть полягає в тому, що на кожному кроці групуються та сортуються елементи, що стоять один від одного на певній відстані*h*. Потім ця відстань зменшується на крок рівний степені двійки. На останньому кроці іде звичайне одинарне сортування. Перша відстань вибирається відносно кількості елементів в масиві і поділена на 2.

Приклад:

44 55 12 42 94 18 06 67 n=8, h=n/2=4

44 18 06 42 94 55 12 67 h=h/2=4/2=2

06 18 12 42 44 55 94 67 h=h/2=2/2=11

06 12 18 42 44 55 67 94 стоп.

Час роботи залежить від вибору значень *h*. Існує декілька підходів вибору цих значень:

- · При виборі , , , …, $h_m = 1$ час роботи алгоритму, в найгіршому випадку, становить $O(N^2)$.
- · Якщо *h* - впорядкований за спаданням набір чисел виду $(3^j – 1)/2$, j Î N, $h_i < N$, то час роботи є$O(N^{1.5})$.
- · Якщо *h* - впорядкований за спаданням набір чисел виду $2^i 3^j$, i, jÎ N,

$h_k < N$, то час роботи є $O(N \cdot \log^2 N)$.

4.6. Сортування розподілом

Методи сортування розподілом являють собою природне узагальнення ручних методів сортування. Вони передбачають розбиття вхідної множини
елементів M наp підмножини $M_{1,...,} M_{p,}$ для яких виконуються умови $M_1 < M_2 < ... < M_p$

Алгоритм R має рекурсивну структуру (позначено квадратними дужками)

Нехай задано множину M елементів, що розбита на p підмножини M_i

R1. Якщо $|M|=0$ або $|M|=1$, кінець.

R2. Інакше [Розбиття M на M_1... M_pтак, щоб $M_1 < M_p$; $i=1$, виконувати доки $i<p$ [Сортування розподілом підмножиниM_i ; $i=i+1$]] .

R3. Кінець. Вихід.

Одну з версій цього алгоритму називають**цифровим** або порозрядним сортуванням за аналогією з системою числення . На практиці механічне сортування починається з молодшого розряду і просувається в напрямку до старшого, при цьому відсортовані підмножини послідовно об'єднуються і перерозподіляються.

Приклад:

Дано	сортування «одиниць»	сортування «десятків»	сортування «сотень»
329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

Цифрове сортування не використовує порівняння елементів, тому не можна оцінювати його в звичайних термінах. Якщо елементи містять m цифр, то потрібно виконати m проходів і, якщо на кожному проході розподіляється nелементів, то основні кроки виконуються$m*n$ раз. Цифрове сортування зручно використовувати для рядків символів.

4.7. Сортування злиттям або об'єднанням

Методи цього класу працюють за таким принципом: невідсортовану вхідну множину довільно розбивають на підмножини M_1,..., M_p. Потім ці підмножини сортують окремо одним з відомих методів сортування і об'єднують в одну відсортовану множину .

Продемонструємо цю схему на двохмножинах, оскільки багаторазове злиття можна здійснювати багаторазовим виконанням попарного злиття.

Нехай маємо дві відсортовані множини X і Y; потрібно об'єднати їх у множину Z, яка також повинна бути відсортованою. У ролі Z_1 приймаємо $min\ (x_1, y_1)$ якщо $Z_1 = x_1 \in X$, тоді $Z_2 = min(x_2, y_1)$ і т.д.

Для запису алгоритму використовуємо таке позначення: i - індекс для множини X, j - індекс для множини Y, k – індекс для множини Z, $i=1..n; j=1..m; k=1..(n+m)$.

Для зручності розмістимо в кінці кожної множини фіктивні елементи: $x_{n+1} = max, y_{m+1} = max$.

Алгоритм С

Дано $X=\{ x_1 \ ,..., x_n \}$; $Y=\{ y_1 \ ,..., y_m \}$.

С1. Ініціалізація індексів $i=1, j=1, k=1$;

С2. Виконувати С3, С4 доки $k<n+m$.

С3 [Якщо $x_i < y_i$ то $[z_k = x_i; i=i+1]$ інакше $[z_k=y_i, j=j+1]$

С4. $k=k+1$]

С5. Кінець. Вихід.

В квадратних дужках записані дії які повторюються. Кількість порівнянь, яку необхідно виконати в алгоритмі С для злиття двох множин, дорівнює $|x|+|y|=n+m$.

Вся процедура злиття разом вимагає не більше ніж n порівнянь для n елементів і потрібно буде виконати $log_2\ n$ переходів із однієї множини у другу.
Тобто алгоритм С вимагає $n\ log_2\ n$ порівнянь.

Час роботи алгоритму злиття $T(n)$ для nелементів задовольняє рекурентному співвідношенню: $T(n) = 2 \cdot T(\cdot n/2) + O(n)$, де $T(\cdot n/2)$ - час на впорядкування половини масиву, $O(n)$ - час на злиття цих половинок. Враховуючи, що $T(1) = O(1)$, розв'язком співвідношення є: $T(n) = O(n \cdot \log(n))$.

4.8. Сортування підрахунком

Сортування підрахунком (англійською «Counting Sort») — алгоритм впорядкування, що застосовується при малій кількості різних елементів (ключів) у масиві даних. Час його роботи лінійно залежить як від загальної кількості елементів у масиві, так і від кількості різних елементів.

Ідея алгоритму полягає в наступному: спочатку підрахувати скільки разів кожен елемент (ключ) зустрічається в вихідному масиві. Спираючись на ці дані можна одразу вирахувати на якому місці має стояти кожен елемент, а потім за один прохід поставити всі елементи на свої місця.

В алгоритмі присутні тільки прості цикли довжини N (довжина масиву), та один цикл довжини K (величина діапазону). Отже, обчислювальна складність роботи алгоритму становить $O(N + K)$.

В алгоритмі використовується додатковий масив. Тому алгоритм потребує $E(K)$ додаткової пам'яті.

В такій реалізації алгоритм є стабільним. Саме ця його властивість дозволяє використовувати його як частину інших алгоритмів сортування (наприклад, сортування за розрядами). Використання даного алгоритму є доцільним тільки у випадку малих K.