ЛЕКЦІЯ 6
МЕТОДИ СОРТУВАННЯ НА ДЕРЕВАХ

6.1. Сортування на деревах

Розглянемо два методи сортування, які використовують деревоподібне зображення початкової таблиці: простий і складний. Простий метод подібний до класу алгоритмів вибірки, складний - спирається на пірамідальне зображення дерева, його називають також по імені автора - алгоритмом Флойда .

6.1.1. Метод вибірки з дерева /алгоритмН /.

Послідовність чисел розбивається на пари, які об'єднуються за принципом «син-батько». Батьком з двох синів стає найбільше число. Процес повторюється, доки не буде виділене одне число, найбільше, яке стане корнем утвореного дерева. У разі відсутності одного числа, батьком стає єдиний нащадок (син). Число, що попало в корінь замінюється на безмежність. Процес повторюється для знаходження наступного найбільшого числа і т.д. З рис. 6.1 видно, що задана послідовність буде впорядкована у низхідному порядку за 10-1=9 кроків.

Сумарний час виконання такого сортування приблизно пропорційний величині $n \log_2 n$. Існує декілька модифікацій цього алгоритму, які скорочують цей час .

Рис.6.1. Схема сортування методом вибірки з дерева:

а - послідовність з десяти чисел для відбору першого найбільшого елемента;

б - вибір другого найбільшого елемента.

6.1.2. Пірамідальне сортування.

Цей метод опирається на пірамідальну структуру дерева. Він складається з двох частин - побудови піраміди (алгоритм Р) і безпосередньо сортування (алгоритм F).

Послідовність ключів K_1 , $K_2 ... K_n$ називають "пірамідою", якщо $K_j < K_i$ при $2<j<n$. $i = [j/2]$ - ціла частина від $j/2$.

Бінарне дерево розміщується послідовно таким чином, що індекси лівого і правого "синів" запису будуть мати відповідно значення $2i$ і $2i+1$. Навпаки, індекс "батька" запису буде мати значення $[j/2]$.Якщо знайдено пірамідальне зображення таблиці, то запис з найбільшим ключем знаходиться в корені дерева.

На вхід алгоритму Р подається невідсортована послідовно розміщенатаблиця, а на виході - утворюється піраміда. Початковим моментом у роботіалгоритму *P* є побудова піраміди, яка на початку складається з одного запису. Потім в неї послідовно вставляються чергові записи, поки не вичерпаються всі записи вхідної таблиці і в результаті чого сформується піраміда.

В алгоритмі Р використані такі позначення: вхідна таблиця *R* міститьnзаписів $R_1, ... , R_n$; індексна змінна *q*служить для керування кількістю виконаних вставок; змінна *i* є індексом

"батька" запису R_j ; *NEW*- робоча область для запису; *KEY* містить ключ запису, який у даний момент повинен бути вставлений у наявну піраміду.

Алгоритм P побудови піраміди.

P1. Побудова піраміди. Повторювати кроки *P2 –P6* при $q=2,3,4,...,n$.

P2. Ініціалізація: встановити $j=q$, $NEW=R_q$.; $KEY= K_q$.

P3. Вставка нового запису в наявну піраміду. Повторювати кроки P4, P5 доти, доки $i>1$.

P4. Знаходження "батька" нового запису: встановити $i=[j/2]$.

P5. Перестановка записів: якщо $KEY > K_i$,то встановити $R_i = R_j$, $i=j$, інакше перейти до кроку P6.

P6. Копіювання нового запису у відповідне місце: встановити $R_j =NEW$.

P7. Кінець. Вихід.

Перший крок алгоритму - оператор, що повторюється. Він керує побудовою потрібної піраміди, послідовно вставляючи нові записи. У кроці P2 вибирається запис, який повинен бути встановлений у наявну піраміду, і виконується копіюванням цього запису в область NEW. У кроках P4 і P5 новий запис приєднується /у вигляді листка/ до наявної піраміди /тобто до бінарного дерева/ і просувається на дереві по шляху між новим листком і вершиною піраміди. Цей процес повторюється доти, доки новий запис не досягне в дереві позиції, що задовольняє визначенню піраміди. Копіювання нового запису у відповідне йому місце в дереві виконується у кроці P6.

Безпосередньо сортування наявної піраміди зводиться до багаторазового виключення вершини піраміди і подальшої її реконструкції - запису цієї вершини на своє остаточне місце. Алгоритмсортування використовує алгоритм P -алгоритм побудови піраміди.

Алгоритм F сортування з використаннямалгоритму *P- **алгоритм Флойда***.

Нехай задана таблиця *R* , що містить *n*записів $R_1,... , R_n$, і алгоритм побудови піраміди P.
Цей алгоритм здійснює сортування таблиці у зростаючому порядку. Змінна *q* є індексом обходу дерева. Індексні змінні *i ,j,* використовуються у тому випадку, коли *j* є індексом лівого "сина" запису з ключем K_i;*SAVE* - робоча область зберігання запису;*KEY* - змінна, що містить ключ запису при кожному проходженні; *i* - індекс вершини - "батька".

Алгоритм F по кроках.

Fl. Побудова піраміди: виклик програми P.

F2. Виконання сортування: повторювати кроки F2 - F8 при $q=n,n-1,...,2$.

F3. Локалізація запису з індексом *q* : встановити $R_1= R_q$.

F4. Ініціалізація індексів: встановити $i=1,,SAVE= R1,KEY=k1, j=2$.

F5. Реконструкція піраміди: повторювати кроки F6, F 7 доки $j<q-1$.

F6. Одержання індекса "сина" з найбільшим значенням ключа:

якщо $j+1<q, K_{j+1} > K_j$, встановити$j=j+1$.

F7. Чи потрібна перестановка записів? Якщо $K_j > KEY,$ то встановити $R_i = R_j, i=j, j=2*I;$ Інакше перейти на крок F8.

F8. Копіювання запису на відповідне місце: встановити $R_i=SAVE$.

F9. Кінець: вихід.

Робота даного алгоритму починається з побудови піраміди для вхідної таблиці. Крок F2 здійснює керування *n-1*проходженням всієї таблиці, необхідним для її сортування, інші кроки аналогічні крокам алгоритму Р для побудови нової піраміди після включення нового запису. Ця аналогія полягає у тому, що вершинаR_1 піраміди поміняється місцями з вершиною R_n, а потім нова піраміда, що містить *n-1* записів, реконструюється за допомогою програми Р. Результатом такої реконструкції є розміщення запису з другим найбільшим значенням ключа у вершині R_1. Цей запис поміняється місцями із записом R_{n-1}, і будується нова піраміда, що містить *n-2* записи, і т.д.

Аналіз найгіршого випадку для цьогоалгоритму показав, що число порівнянь, які потрібно виконати, дорівнює величині порядку $O(n \log_2 n)$. Крім того, для даногоалгоритму непотрібно додаткових робочих областей пам‘яті, за виключенням поля для одного запису.

Отже, розглянуто велику кількість різних методів сортування. Кращі методи потребують порядку $n \log_2 n$ порівнянь, найпростіші - порядку n^2 . Методи сортування з обчисленням адреси і числове або цифрове сортування використовують додаткову інформацію про дані і вимагають *n* порівнянь. Кращі методи сортування не вимагають додаткової пам‘яті, крім тієї, яка потрібна для зберігання даних, програми і фіксованого набору керуючих величин. Багато методів сортування використовують додаткову пам‘ять обсягом порядку $\log_2 n$ або *n* квантів.

При виборі методу сортування необхідно враховувати структуру даних, вимоги до часу і обсягу пам‘яті, а також складність програмування алгоритму.