ЛЕКЦІЯ 13

ПОШУК У ТАБЛИЦЯХ

Пошук в масиві іноді називають пошуком в таблиці, особливо якщо ключ сам є складовим об‘єктом, таким як масив чисел або символів. Для того щоб встановити факт співпадіння, необхідно переконатись, що всі символи стрічок які порівнюються співпадають. Порівняння зводиться до пошуку їх неспівпадіння., тобто пошуку на нерівність. Якщо нерівних частин не існує, то можна говорити про рівність. Якщо розмір шуканого взірця невеликий, то доцільно використати алгоритм лінійного пошуку. В противному випадку необхідно впорядкувати таблицю та скористатисьалгоритмом пошуку діленням пополам. Від вибраного алгоритму пошуку буде залежити алгоритмічна складність.

13.1. Пошук у таблицях з обчислюваними адресами

Розглянемо метод, у якому тривалість пошуку буде незалежною від числа записів у таблиці. Для цього таблиці треба організовувати так, щоб місцезнаходження її елементів визначалося за допомогою обчислюваної адреси. Це так звані таблиці з обчислюваними адресами. Загальна ідея організації таких таблиць полягає у тому, щоб використовувати ключ безпосередньо в ролі адреси або визначати адресу за допомогою функції від ключа і запам‘ятовувати запис за цією адресою.

Функції, що відображають ім‘я або ключ у деяке ціле число, називають функціями**хешування** $h(k)$ для ключа k. Їх також називають функціями перемішування або рандомізації.

Ситуацію, при якій $h(k_i)= h(k_j)$ при $k_i != k_j$, називають **колізією**.

Таблиці з обчислюваними адресами поділяються на таблиці з прямим або безпосереднім доступом і перемішані, або хеш-таблиці. Задача колізії має місце тільки в хеш-таблицях.

13.2. Пошук у таблицях з прямим доступом

Таблицю називають таблицею з **прямим**або безпосереднім **доступом**, якщо для визначення місцезнаходження кожного запису використовується його ключ.Функція хешування визначається як відображення $K->A$, де K- множинаключів, які можуть ідентифікувати записи в таблиці з прямим доступом; A -множина адрес. Доступ до запису за ключем k здійснюється в такому випадку за значенням функції $h(k)$.

Мірою використання пам‘яті в таблицях з прямим доступом є коефіцієнт заповненняq , що визначається як відношення числа записів n до числа місць m в таблиці:$q=n/m$.

Вибір функції адресації, що забезпечує взаємну однозначність перетворення ключа в адресу її зберігання, взагалі достатньо складна задача. На практиці її можна розв‘язати тільки для постійних таблиць із заздалегідь відомим набором значень ключа. Такі таблиці застосовуються в трансляторах (наприклад, таблиця символів вхідної мови є таблицею з прямим доступом).

Наведемо приклад таблиці з прямим доступом. Якщо маємо n ключів, то кожному заданому ключу k можна поставити у відповідність адресу запису в діапазоні 0.. $n-1$..

Нехай ключами записів є такі шифри автомобільних номерів (табл.13.1).

Таблиця 13.1

Побудова таблиці з прямим доступом

Ключ	Коди букв	Адреса
AN	0, 13	13
AX	0, 23	23
BC	1, 2	28
BI	1, 8	34
BJ	1, 19	35
CM	2, 12	64
JR	9, 17	251
MO	12, 14	326

Тут використано одну з можливих хеш-функцій, що опирається на алгебраїчне кодування. Функція адресації обчислюється в залежності від коду ключа. Наприклад, тіло ключа MO запам‘ятовується за адресою 12 * 26 + 14 = 326; де 12 – код літери M, 14 – код літери O. Тіло ключа CM за адресою 2 * 26 + 12 =64. За такою функцією адресації всього буде 26^2 можливих адрес (26 - кількість букв латинського алфавіту). З цього прикладу видно, що пошук у такихтаблицях найшвидший, оскільки безпосередньо потрапляємо в потрібне місце без порівняння з іншими ключами.

13.3. Пошук у Хеш-таблицях

Природним розвитком застосування таблиць з безпосереднім доступом є обчислення адреси не в повному діапазоні*0.. n-1,* а в деякому обмеженому. Цей метод відомий як метод перемішаних адрес або метод змістовної адресації.

У простій таблиці з безпосереднім доступом з *n* можливих ключів можна згенерувати унікальну адресу в діапазоні*0.. n-1*. Але якщо з‘являється тільки *l*<<*n*ключів, то втрачається великий обсяг пам‘яті. Отже, доцільно обчислювати адресу в діапазоні *l* < *m*<<*n*, однак це збільшить ймовірність появи колізії.

Розглянемо приклад з табл. 9.1. Для восьми елементів відведемо тільки 10 одиниць пам‘яті, тобто *l=8, m =10*.

Наприклад, для побудови хеш-таблиці будемо обчислювати хеш-адресу від ключа в діапазоні 0..9 таким чином: складемо всі десяткові коди обох символів, що утворюють ключ, а потім візьмемо залишок від ділення цієї суми на 10 (ділення за модулем 10). Значення такої хеш-функції наведені в табл. 13.2, а в табл. 13.3 - побудова хеш-таблиці за цією функцією. Для ліквідації ситуації колізії застосовувалося повторне хешування.

Таблиця 13.2.

Обчислення хеш-адреси

№	Ключ	Коди букв	Адреса
1	AN	0 13	3
2	AX	0 23	3
3	BC	1 2	3
4	BI	1 8	9
5	BJ	1 19	0
6	CM	2 12	4
7	JR	9 17	6
8	MO	12 14	6

Таблиця 13.3.

Побудова хеш-таблиці

Адреса	Ключ
0	BJ
1	
2	
3	AX
4	CM
5	AN
6	MO
7	JR
8	BC
9	BI

Для фіксованих таблиць можна знайти достатньо ефективну функцію адресації. Якщо ж таблиці не фіксовані і всі можливі ключі вибираються з деякої достатньо великої множини потенційно можливих ключів, то ідеальної функції адресації не існує.

Довжина пошуку в хеш-таблицях залежить від коефіцієнта заповнення адресного простору, тобто відношення $q=n/m$, де n - кількість записів; m -розмірність таблиці, а також методу розв‘язку задачі колізії.