4. Перемішані таблиці

Природним розвитком таблиць з прямим доступом є таблиці, обчислення адреси для яких відбувається не в повному проміжку від 0 до N–1, а в деякому обмеженому діапазоні. Цей метод відомий як **метод перемішаних адрес**, або перемішаної пам'яті.

У простій таблиці з прямим доступом є N можливих ключів і для кожного з них можна згенерувати унікальну адресу з діапазону від 0 до N–1. Якщо насправді є лише K ключів і K<N, то не використовується великий обсяг пам'яті. У такому випадку треба обчислювати адресу в меншому діапазоні, наприклад, від 0 до p–1, де K<p<N. Це означає, що різні ключі можуть привести до однієї і тієї ж адреси. Такий випадок називають **колізією**. Для розв'язання колізії є два способи: **рехешування** та **метод ланцюжків** (або використання області переповнення).

Суть рехешування. Нехай обчислено значення хеш-функції для ключа S і це значення — h. При спробі занести елемент з цим ключем у таблицю виявляється, що інший елемент зайняв місце за адресою h. Тоді порівнюємо S з елементом поля $(h + p_1) \bmod N$ (де N — довжина таблиці) для деякого значення p_1. Якщо знову виникає колізія, то порівнюємо S з елементом $(h + p_2) \bmod N$. Цей процес продовжують доти, доки не буде знайдено деякий елемент $(h + p_j) \bmod N$, який або порожній, або містить S , або знову є елементом h. В останньому випадку робота алгоритму припиняється, оскільки таблиця повна. Отже, якщо виникло j колізій, буде виконано j+1 порівняння з елементами $h_j = (h + p_j) \bmod N$. Величину p_j треба вибирати так, щоб очікуване число порівнянь E було невеликим і щоб можна було розглянути якнайбільше елементів. В ідеальному випадку p_j повинні охоплювати цілі числа 0, 1, …, N–1. Тип рехешування визначається тим, як вибирають значення p_j . Найпоширенішими є:

- лінійне рехешування — коли $p_j = j$;
- квадратне — $p_j = a\, j^2 + bj + c$, де a, b, c — константи, які вибирають так, щоб забезпечити перегляд більшої кількості позицій;
- рехешування додаванням — $p_j = j * h$;
- випадкове рехешування;

Випадкове рехешування доцільно використовувати тоді, коли довжина таблиці дорівнює степеневі числа 2, тобто $N = 2^q$. Алгоритм обчислення p_j у цьому випадку такий:

1. Викликаючи програму, прийняти, що цілочислова змінна R дорівнює 1;
2. Обчислювати кожне p_j так:
    - визначити $R=R*5$;
    - виділити молодші q+2 розряди R і помістити результат у R;
    - взяти величину з R, зсунути її праворуч на 2 розряди і результат назвати p_j;

Важлива властивість цього методу, який запобігає нагромадженню елементів в одній частині таблиці, полягає в тому, що всі числа від p_j до p_{j+q} різні.

Очікувану кількість порівнянь дає формула: $E = 1 / lf * \ln(1 - lf)$.

Рехешування іноді називають **відкритою адресацією**.

Розглянемо приклад побудови таблиці, в яку заноситимемо прізвища студентів групи з 25 осіб. За хеш-функцію візьмемо порядковий номер в алфавіті першої букви прізвища. Це дасть змогу побудувати таблицю з 32-ох елементів (33 букви абетки мінус буква «ь»). Для розв'язання колізії скористаємось випадковим рехешуванням, оскільки величина таблиці є степенем числа 2 (довжина таблиці N=32; це 2 в п’ятому степені, тому q=5).

Таблиця складатиметься з двох частин: таблиці означень (в яку заноситимемо посилання на відповідні імена) та таблиці імен. У таблиці імен будуть знаходитись ключі (імена) і значення, які міститимуть деяку додаткову інформацію, пов'язану з відповідним іменем. Для посилання на таблицю імен будуть використані значення індексів з цієї таблиці.

Нехай треба помістити в таблицю такі прізвища:

1. Будна
2. Тройська
3. Годій
4. Гойцак
5. Гупаловська
6. Дзіковська

Ці прізвища разом з інформацією, яку ми не описуватимемо, будуть у таблиці імен, а в таблицю означень буде занесено вказівки на них. Для кожного імені, яке заносимо в таблицю, обчислюємо хеш-функцію та, у разі потреби усунення колізії, значення p_j. Обчислимо хеш-функцію для першого прізвища:

h(Будна) = 2 (бо буква «Б» в абетці має номер 2). Оскільки таблиця порожня, то з занесенням у неї цього елемента немає жодних проблем.

Так само без проблем буде занесено у таблицю і друге прізвище, бо h(Гронська) = 4.

0		12		24	
1		13		25	
2	100	14		26	
3	300	15		27	
4	150	16		28	
5	200	17		29	
6	350	18		30	
7		19		31	
8		20		32	
9		21		33	
10	250	22		34	
11		23		35	

Табл. 1. Таблиця означень.

<адреса>	<ключ>
100	Будна
150	Гронська
200	Годій
250	Гойцак
300	Гупаловська
350	Дзіковська

Табл. 2. Таблиця імен.

Спроба занести у таблицю прізвище Годій спричинить колізію, бо h(Годій) = 4. Обчислимо p_1, використавши алгоритм випадкового рехешування: беремо R=1*5=5; беремо 7 молодших бітів цифри «5», отримуємо 0000101; зсуваємо праворуч на два розряди, отримуємо 00001; тобто $p_1 = 1$. $(h + p_1) \bmod 32 = 5$, тому новий елемент буде занесено за адресою 5.

І так далі.

h(Гойцак) = 4 — колізія. Обчислюємо p_1 (це описано вище); $(h + p_1) \bmod 32 = 5$ — знову колізія і тому обчислюємо p_2: R=5*5=25; у бітовому вигляді це 0011001; зсуваємо на дві біти праворуч, отримуємо 00110, тобто $p_2 = 6$; $(h + p_2) \bmod 32 = 10$ тому прізвище Гойцак буде занесено за адресою 10.

h(Гупалонська) = 4 — колізія. Обчислюємо p_1; $(h + p_1) \bmod 32 = 5$ — колізія, обчислюємо p_2; $(h + p_2) \bmod 32 = 10$ — знову колізія і тому обчислюємо p_3: R=25*5=125; 01111101 ; $p_3 = 31$; $(h + p_3) \bmod 32 = 3$ тому прізвище Гупаловська буде занесено за адресою 3.

h(Дзіковська) = 5 — колізія. Обчислюємо $p_1 = 1$; $(h + p_1) \bmod 32 = 6$ і тому прізвище Дзіковська буде занесено за адресою 6.

Інший спосіб розв'язання колізії передбачає наявність, крім основної, додаткової таблиці, куди поміщають записи, які вступили у колізію. Зберігання записів у додатковій таблиці організовують по-різному: наприклад, в ній розташовують усі записи послідовно, а це означає, що для пошуку у додатковій таблиці застосовують послідовний перегляд. Пошук у таблиці може бути пришвидшений, якщо у кожній позиції основної і додаткової таблиць створити додаткове поле для зберігання посилання на записи, які вступили у колізію. Такі таблиці називають **таблицями з ланцюжками для розв'язання колізії**.

Метод ланцюжків використовує хеш-таблицю, елементами якої є вказівники з порожнім початковим значенням, та таблицю елементів. Таблиця спочатку порожня і вказівник **p**, який показує на поточне положення останнього елемента в таблиці, встановлено на елемент перед таблицею. Елементи таблиці мають додаткове поле, яке може містити порожній вказівник або адресу іншого елемента таблиці. Хеш-функція, застосована до ключа, дає місце в хеш-таблиці, де розташовано вказівник,

який або порожній, або вказує на перший елемент таблиці елементів з даним значенням хеш-функції. Поле і адреси кожного елемента використовують для того, щоб зв'язати у ланцюжок елементи, для яких хешування ключа дає одне і те ж значення.

Після того, як обчислено значення хеш-функції, виконують таке:

1. змінюють значення вказівника **p** у таблиці елементів;
2. значення елемента заносять у позицію, визначену цим вказівником;
3. значення **p** заносять у хеш-таблицю на місце, визначене хеш-фукцією;

За цим алгоритмом заносять ті елементи, які мають різні значення хеш-функції. Якщо ж виникає колізія, то змінюють поле адреси у таблиці елементів.

Розглянемо як будуть виглядати обидві таблиці після занесення в них імен B1, A, A2, C, B2 (рис. 3). За хеш-функцію візьмемо номер в алфавіті першої літери імені.

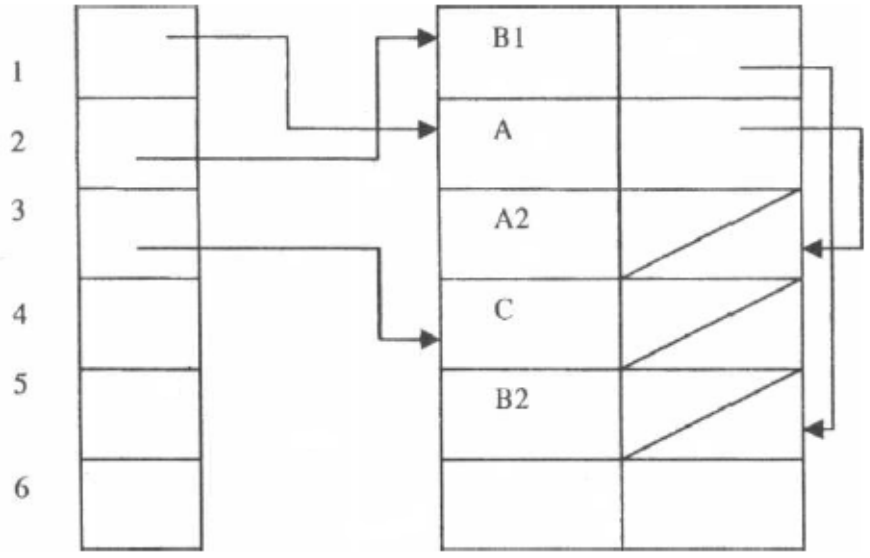

Рис. 3. Хеш-таблиця та таблиця елементів.