Практична робота 23-24

Хеш-таблиці й алгоритми їх обробки

Короткі теоретичні відомості

Хеш-функція - функція, перетворювати вхідну послідовність даних довільного розміру в вихідну послідовність фіксованого розміру. Процес перетворення даних називається хешуванням. Результат - хеш-код.

При розв'язанні класу практичних завдань вибирається така хеш-функція, яка є найбільш оптимальною саме для даного класу. У загальному випадку слід використовувати «гарну» функцію. Коли хеш-функцію називають «гарною», то мають на увазі під цим, що вона:

1. обчислюється досить швидко;

2. зводить до мінімуму число колізій.

Запобігти колізії можуть далеко не всі хеш-функції, але «гарні» здатні мінімізувати ймовірність їх появи. При певних обставинах (відома деяка інформація про ключі), можна знайти ідеальну хеш-функцію, яка повністю виключає можливість появи колізій.

Метод ділення.

Нехай **k** - ключ (той, що необхідно хешувати), а **N** - максимально можливе число хеш-кодів. Тоді метод хешування за допомогою ділення полягатиме у взятті залишку від ділення **k** на **N**:

h (k) = k mod N, де **mod** - операція взяття залишку від ділення.

Наприклад, на вхід подаються наступні ключі:

3, 6, 7, 15, 32, 43, 99, 100, 133, 158.

Визначимо **N** рівним 10, з чого слідує, що можливі значення хешів лежить в діапазоні 0 ... 9. Використовуючи цю функцію, отримаємо наступні значення хеш-кодів:

h(3)=3, h(6)=6, h(7)=7, h(15)= 5, h(32)=2, h(42)=2, h(99)=9, h(100)=0, h(133)=3, h(158)=8.

```
int HashFunction(int k)
{
return (k%10);
}
void main()
{
setlocale(LC_ALL, "Rus");
int key;
cout<<"Ключ > ";
cin>>key;
cout<<"HashFunction("<<key<<")="<<HashFunction(key)<<endl;
system("pause>>void");
}
```

Метод множення.

Отримати з вихідної послідовності ключів послідовність хеш-кодів, використовуючи метод множення (мультиплікативний метод), значить скористатися хеш-функцією:

$$h(k)=\lfloor N*(\{k*A\})\rfloor$$

де **A** - раціональне число, по модулю менше одиниці (**0 <A <1**), а **k** і **N** позначають те саме, що і в попередньому методі: ключ і розмір хеш-таблиці. Також права частина функції містить три пари дужок:

- ( ) – дужки пріоритету;
- ⌊ ⌋ – дужки взяття цілої частини;
- { } – дужки взяття дробової частини.

Аргумент хеш-функції **k (k≥0)** в результаті дасть значення хеш-коду **h (k) = x,** що лежать в діапазоні **0 ... N-1**. Для роботи з від'ємними числами можна число x взяти по модулю.

Від вибору **A** і **N** залежить те, наскільки оптимальним виявиться хешування множенням на певній послідовності. Не маючи відомостей про вхідні ключі, у якості N слід вибрати один із ступенів двійки, так як. множення на $\mathbf{2^m}$ рівносильно зсув на **m** розрядів, що комп'ютером проводитися швидше. Непоганим значенням для A (в загальному випадку) буде (√5-1) / 2≈0,6180339887. Воно засноване на властивостях золотого перетину:

Золотий перетин - такий розподіл величини на дві частини, при якому відношення більшої частини до меншої дорівнює відношенню всієї величини до її більшої частини.

Ставлення більшої частини до меншої, виражене квадратичною ірраціональністю:

φ=(√5+1)/2≈1,6180339887

Для мультиплікативної хеш-функції було приведено зворотне відношення:

1/φ=(√5-1)/2≈0,6180339887

При такому **A**, хеш-коди розподіляться досить рівномірно, але багато що залежить від початкових значень ключів.

Для демонстрації роботи мультиплікативного методу, покладемо N = 13, A = 0,618033. Як ключі візьмемо числа: 25, 44 і 97. Підставами їх в функцію:

1. h(k)=⌊13*({25*0,618033})⌋=⌊13*{15,450825}⌋=⌊13*0,450825⌋=⌊5,860725⌋=5
2. h(k)=⌊13*({44*0,618033})⌋=⌊13*{27,193452}⌋=⌊13*0,193452⌋=⌊2,514876⌋=2
3. h(k)=⌊13*({97*0,618033})⌋=⌊13*{59,949201}⌋=⌊13*0,949201⌋=⌊12,339613⌋=12

Приклад 2

```
int HashFunction(int k)
{
int N=13; double A=0.618033;
int h=N*fmod(k*A, 1);
return h;
}
void main()
{
setlocale(LC_ALL, "Rus");
int key;
cout<<"Ключ > "; cin>>key;
cout<<"HashFunction("<<key<<")="<<HashFunction(key)<<endl;
system("pause>>void");
}
```

Постановка задачі

▫У *хеш-таблиці* замість безпосереднього використання ключа як індексу масиву, індекс обчислюється за значенням ключа. Функція, що відображає елемент множини ключів {0, 1, ..., n-1} на множину індексів {0, 1, ..., m-1} ($m < n$), називається *хеш-функцією*. Якщо два ключі хешуються в одну й ту саму комірку, то говорять про виникнення *колізії*. За способом вирішення колізій розрізняють:

- *відкрите хешування* — усі елементи, що хешуються в одну комірку, об'єднуються у зв'язний список
- *закрите хешування* — усі елементи зберігаються безпосередньо у хеш-таблиці, при потраплянні у зайняту комірку обирається послідовність інших хеш-значень.

Необхідно забезпечити для хеш-таблиці реалізацію основних операторів:

- пошук елемента;
- запис елемента;
- читання елемента.

Опис алгоритмів

Опис структур даних

При закритому хешуванні хеш-таблиця є масивом, елементи якого занумеровані від 0 до m-1.

При відкритому хешуванні хеш-таблиця є масивом, кожна комірка якого містить покажчики на заголовок списку всіх елементів, хеш-значення ключа яких дорівнює індексу комірки.

1. Побудова хеш-функції методом ділення

Ідея

Значенням хеш-функції є остача від ділення ключа на розмір хеш-таблиці.

Псевдокод

$h(x)$ = x **mod** m

2.Пошук елемента при відкритому хешуванні

Ідея

Знайти відповідне хеш-значення і здійснити пошук у списку, заголовок на який міститься у відповідній комірці хеш-таблиці.

Псевдокод

заголовок = хеш-таблиця [*h* (*ключ*)]

знайти позицію *ключа* в списку, заданого *заголовком*

повернути *позиція* != NULL

3.Запис елемента при відкритому хешуванні

Ідея

Якщо елемент не знайдено, то додати його на початку списку, що відповідає хеш-значенню.

Псевдокод

якщо Пошук (*ключ*) == FALSE **то**

{

заголовок = хеш-таблиця [*h* (*ключ*)}

вставити *ключ* в позицію 1 списку, заданого *заголовком*

{

4.Вилучення елемента при відкритому хешуванні

Ідея

Вилучити елемент зі списку, заголовок на який міститься у відповідній комірці хеш-таблиці.

Псевдокод

заголовок = хеш-таблиця [*h* (*ключ*)]

p = позиція *ключа* в списку, заданого *заголовком*

видалити елемент в позиції *p* зі списку, заданого *заголовком*

5. Вставка елемента при закритому хешуванні

Ідея

При лінійному хешуванні використовується хеш-функція $h(x, i)=(h1(x)+i)$ **mod** m, де $h1(x)$ — звичайна хеш-функція, i — номер спроби розмістити елемент. Якщо комірка, відповідна хеш-значенню зайнята, то обчислюється значення для наступної спроби.

Псевдокод

i = 0
повторювати
{
k
= *h*(*x*, *i*)
}
поки *хеш-таблиця*[*k*] ==0
хеш-таблиця[*k*] = *x*

Завдання для виконання

1. Проаналізувати та реалізувати приклади 1 та 2.
2. Скласти програму для реалізації операцій з хеш-таблцею при відкритому хешуванні; хеш-фунція будується методом ділення.
3. Скласти програму для реалізації додавання (вилучення) елементів у хеш-таблицю при лінійному хешуванні і заповнення її випадково згенерованими числами.