Алгоритми та структури даних
Модуль 1: Структури даних
Лекція 6. Хеш-таблиці.

Кудін Олексій Володимирович
avk256@gmail.com

План

1. Поняття хеш-таблиці.
2. Види хеш-функції.
3. Розв’язання колізій.
4. Динамічні хеш-таблиці.

Питання по попередній лекції

- *Що таке дерево, бінарне дерево?*
- *Які існують способи обходу дерев?*
- *З якою задачею пов’язано різні способи обходу дерев?*
- *Способи представлення дерев в пам’яті?*
- *Який АТД називають Словником?*

АТД Словник

Словник (dictionary) – це спеціальний вид АТД Множина з такими операторами.

INSERT(X,D) (Вставка)

DELETE(X,D) (Видалення)

MEMBER(X,D) (Пошук)

Яка часова складність операторів АТД Словник при реалізації традиційними методами (наприклад, за допомогою масивів)?

АТД Словник

Один з можливих варіантів реалізації АТД Словник – використання хеш-таблиць для зберігання даних та реалізації операторів

Вставка

Видалення

Пошук

як функцій роботи з хеш-таблицями.

Основна операція – **Пошук**.

Хеш-таблиця

Хеш-таблиця (асоціативний масив) – абстрактний тип даних, що дозволяє зберігати дані у вигляді набору пар **ключ-значення**, та виконувати операції додавання, видалення та пошуку елементів за їх значеннями.

Перевага – постійний час виконання операцій вставки, видалення, пошуку.

Хеш-таблиця

Алгоритми пошуку у хеш-таблиці складаються з двох частин:

1. Обчислення хеш-функції.
2. Розв’язання конфліктів.

Вимоги до хеш-функції.

Всі можливі ключі переводяться у натуральні числа в діапазоні [0, *M-1*].

Для різних значень ключа генерують різні значення хеш-функції (в ідеальному випадку)

Обчислення Хеш-функції

Модульні хеш-функції для числових ключів:

$H(k)=k \bmod M$

$H(k)=\lfloor k*a \rfloor \bmod M$

$H(k)=(k+a) \bmod M$

Реалізація хеш-функції

Для строкових ключів можлива наступна реалізація хеш-функції.

```
int hash(char *v, int M)
 { int h = 0, a = 127;
   for (; *v != 0; v++)
    h = (a*h + *v) % M;
   return h;
 }
```

```
int hashU(char *v, int M)
 { int h, a = 31415, b = 27183;
   for (h = 0; *v != 0; v++, a = a*b % (M-1))
    h = (a*h + *v) % M;
   return (h < 0) ? (h + M) : h;
 }
```

Реалізація хеш-функції

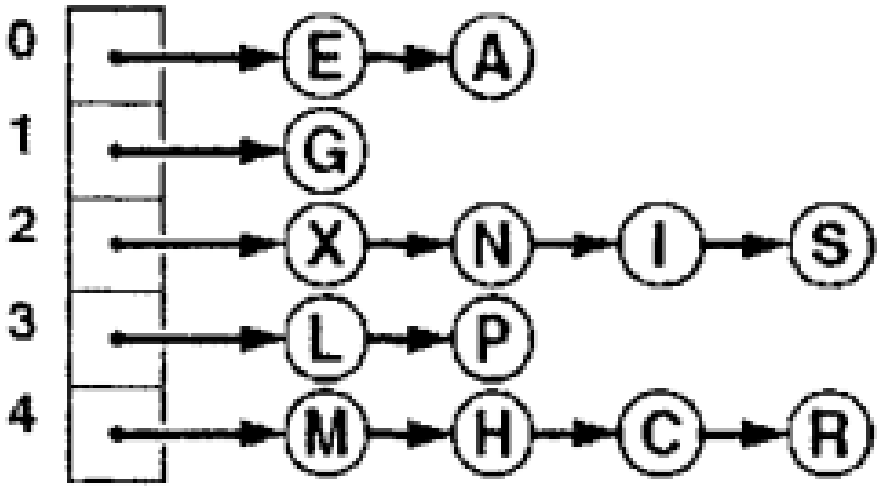
Для ключів типу float, які знаходяться у диапазоні (s,t) можна використати наступну хеш-функцію

```
inline int hash(Key v, int M)
    { return (int) M*(v-s)/(t-s); }
```

Роздільне зв’язення

Один з можливих способів розв’язання колізій – метод роздільного зв’язення.

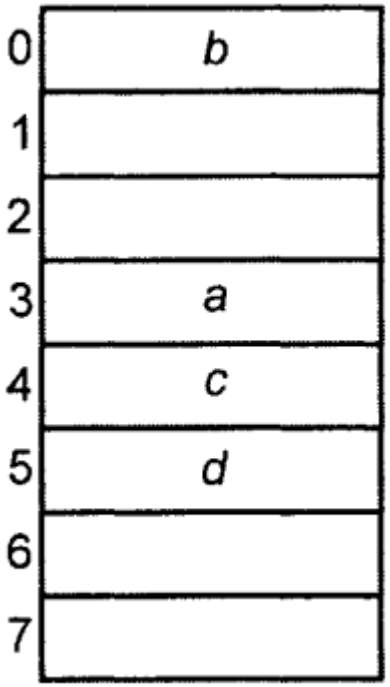
A	S	E	R	C	H	I	N	G	X	M	P	L
0	2	0	4	4	4	2	2	1	2	4	3	3



Лінійне зондування

Другий метод розв’язання колізій – лінійне зондування.

Приклад. Нехай M=8, h(a)=3, h(b)=0, h(c)=4, h(d)=3, hi(x)=(h(x)+i) mod M.



Подвійне хешування

Для запобігання утворення кластерів – послідовності елементів, які мають однаковий хеш – використовують метод подвійного хешування.

```
void insert(Item item)
  { Key v = item.key();
    int i = hash(v, M), k = hashtwo(v, M);
    while (!st[i].null()) i = (i+k) % M;
    st[i] = item; N++;
  }
Item search(Key v)
  { int i = hash(v, M), k = hashtwo(v, M);
    while (!st[i].null())
    if (v == st[i].key()) return st[i];
       else i = (i+k) % M;
    return nullItem;
  }
```

Приклад. В хеш-таблицю розміром M=16 додаються елементи з ключами e, a, s, y, q, u, t, i, o, n

H1(x)=11k mod M

H2(x)=(k mod 3)+1.

Використати метод подвійного хешування.

Динамічні хеш-таблиці

При заповненні хеш-таблиці ефективність операції пошуку значно знижується.

Для запобігання переповнення хеш-таблиці та прискорення операції пошуку використовується операція подвоєння розміру хеш-таблиці.

Резюме

Резюме (що встигли за лекцію)

- Визначили поняття хеш-таблиці.
- Навели основні види хеш-функцій.
- Розглянули 3 метода розв’язання колізій.
- Визначили поняття днамічної хеш-таблиці.

До наступної лекції.

Побудуйте хеш-таблицю для елементів e, a, s, y, q, u, t, i, o, n з початковим розміром M=4. Таблиця збільшується вдвічі після заповнення її на половину. Використовується метод лінійного зондування та хеш-функція вигляду

H(k)=11k mod M.