Хеш-таблицы

Лекция 8

Хеш-функция

Определение. **Хеш-функция, это функция, которая преобразует произвольное число в число из определенного диапазона 0,…,M-1.**

Примеры.

h(x)=x mod M

h(x)=A*x+B mod M

h(x)=x, если |x|<M и h(x)=0 иначе.

Хеш-таблица

Хеш-таблица – это такая таблица, в ячейках которой хранятся элементы в соответствии со значениями их хеш-функций.

Метод открытой адресации

Хеш функция h(x)=x mod 10

Строим хеш-таблицу из чисел

7, 54, 20, 1, 45, 32.

0	1	2	3	4	5	6	7	8	9
20	1	32		54	45		7		

Добавим число 50. Помещаем в первую свободную ячейку после 0. Добавим 33. Помещаем в первую свободную ячейку после 3.

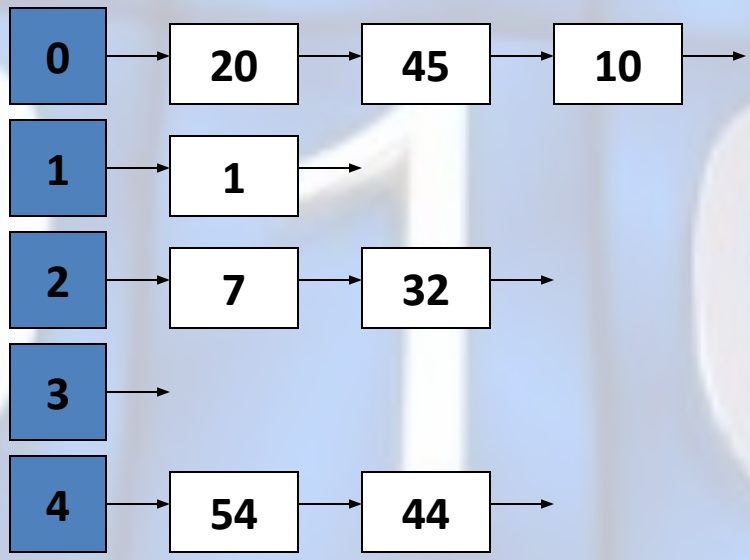
20	1	32	50	54	45	33	7		

Метод цепочек

Хеш функция h(x)=x mod 5

Строим хеш-таблицу из чисел

7, 54, 20, 1, 45, 32, 10, 44



Операции с хеш-таблицей

1. **Вставка элемента**

1. **Удаление элемента**

1. **Поиск элемента**

Реализация метода открытой адресации - 1

```
#include <iostream>
using namespace std;

const int size = 10;
struct Element {
        int item;
        bool empty;
};
Element hashtable[size];
int hfunc(int x) {
        return x%size;
}
```

Реализация метода открытой адресации - 2

```
void init() {
        for (int i=0; i<size; i++) {
                hashtable[i].empty = true;
        }
}
```

Реализация метода открытой адресации - 3

```
int getEmptyPosition(int x) {
        int h = hfunc(x);
        int i=h;
        int trials=0;
        while (trials<size) {
                if (hashtable[i].empty) {
                        return i;
                }
                i=(i+1)%size;
                trials++;
        }
        return -1;
}
```

Реализация метода открытой адресации - 4

```
void add(int x) {
        int pos = getEmptyPosition(x);
        if (pos==-1) {
                cout << "No place\n";
        } else {
                hashtable[pos].item=x;
                hashtable[pos].empty=false;
        }
}
```

Реализация метода открытой адресации - 5

```
void displayTable() {
	for (int i=0; i<size; i++) {
		if (!hashtable[i].empty) {
			cout << hashtable[i].item << " ";
		}
	}
	cout << endl;
}
```

Реализация метода открытой адресации - 6

```
void traceTable() {
      for (int i=0; i<size; i++) {
            if (!hashtable[i].empty) {
                  cout << hashtable[i].item << " ";
            } else {
              cout << -1 << " ";
            }
      }
      cout << endl;
}
```

Реализация метода открытой адресации - 7

```
int main()
{
	init();
	add(rand()%100);
	add(rand()%100);
	add(rand()%100);
	add(rand()%100);
	add(rand()%100);

	displayTable();
	traceTable();
	system("pause");
	return 0;
}
```

```
0 41 34 24 67 78 69
0 41 -1 -1 34 24 -1 67 78 69
```

Реализация метода цепочек - 1

```
struct Element {
        int item;
        Element* next;
};

int size=10;
Element* hashtable[10];

void init() {
        for (int i=0; i<size; i++) {
                hashtable[i] = NULL;
        }
}
```

Реализация метода цепочек - 2

```
Element* createElement(int x) {
        Element* newElement = new Element;
        newElement->item = x;
        newElement->next = NULL;
        return newElement;
}

int hfunc(int x) {
        return x%size;
}
```

Реализация метода цепочек - 1

```
void add(int x) {
        int h = hfunc(x);
        if (hashtable[h]==NULL) {
                hashtable[h] = createElement(x);
        } else {
                Element* temp=hashtable[h];
                while (temp->next != NULL) {
                        temp=temp->next;
                }
                temp->next=createElement(x);
        }
}
```

Реализация метода цепочек - 1

```
void displayChain(Element* chain) {
        Element* temp=chain;
        while (temp != NULL) {
                cout << temp->item << " ";
                temp=temp->next;
        }
}
```

Реализация метода цепочек - 1

```
void displayTable() {
      for (int i=0; i<size; i++) {
            displayChain(hashtable[i]);
      }
      cout << endl;
}

void traceTable() {
      for (int i=0; i<size; i++) {
            cout << i << " -> ";
            displayChain(hashtable[i]);
            cout << endl;
      }
}
```

Реализация метода цепочек - 1

```
int main() {
        init();
        for (int i=0; i<20; i++) {
                add(rand()%100);
        }
        displayTable();
        traceTable();
        system("pause");
        return 0;
}
```

Реализация метода цепочек - 1

```
0 41 81 61 62 34 24 64 5 45 67 27 78 58 69
0 -> 0
1 -> 41 81 61
2 -> 62
3 ->
4 -> 34 24 64
5 -> 5 45
6 ->
7 -> 67 27
8 -> 78 58
9 -> 69
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое хеш-таблица? Привести примеры.
2. Какие методы построения хеш-таблиц существуют?
3. Что такое коллизия?
4. Описать алгоритм и примести примеры построения хеш-таблицы методом открытой адресации.
5. Описать алгоритм и привести примеры построения хеш-таблицы методом открытий адресации.
6. Привести примеры вырожденных случаев, когда сложность поиска в хеш-таблице становится того же порядка, что и при последовательном поиске.

ПРАКТИЧЕСКИЕ ЗАДАНИЯ

1. Реализуйте операции вставки, проверки на пустоту и вывода хеш-таблицы методом открытой адресации.
2. Реализуйте операции вставки, проверки на пустоту и вывода хеш-таблицы методом цепочек.
3. Реализуйте подсчет количества чисел, больших введенного пользователем числа в хеш-таблице.
4. Реализуйте вычисление суммы элементов хеш-таблицы.
5. Реализуйте нахождение максимального и минимального элемента в хеш-таблице.
6. Реализуйте вывод нечетных элементов хеш-таблицы.
7. Определите, есть ли введенный пользователем элемент в хеш-таблице.
8. Вычислите среднее арифметическое элементов хеш-таблицы.
9. Определите количество двузначных элементов в хеш-таблицы.