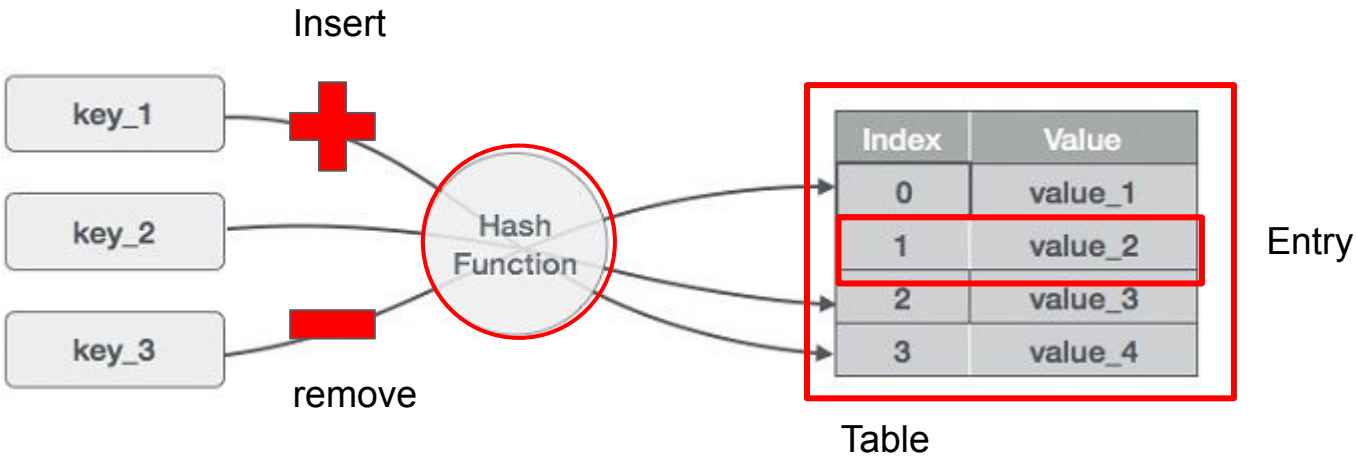
Session 9: Hash Tables

Data Structures and Algorithm 1 - Lab

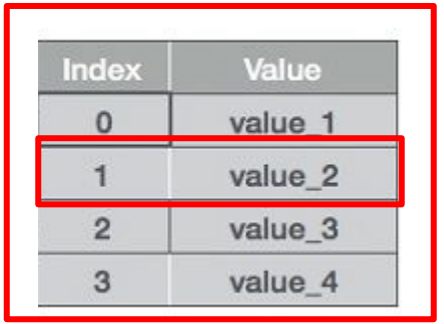
Yahya Tawil
10 Dec 2021

Hash Table Implementation



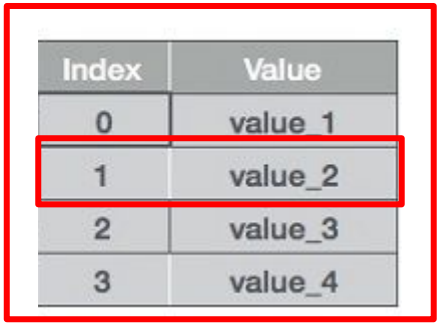
Hash Table Implementation: Data Structure

```
class HashTableEntry {
   public:
      int k; // key
      int v; // value
      HashTableEntry(int k, int v) {
         this->k= k;
         this->v = v;
      }
};
```



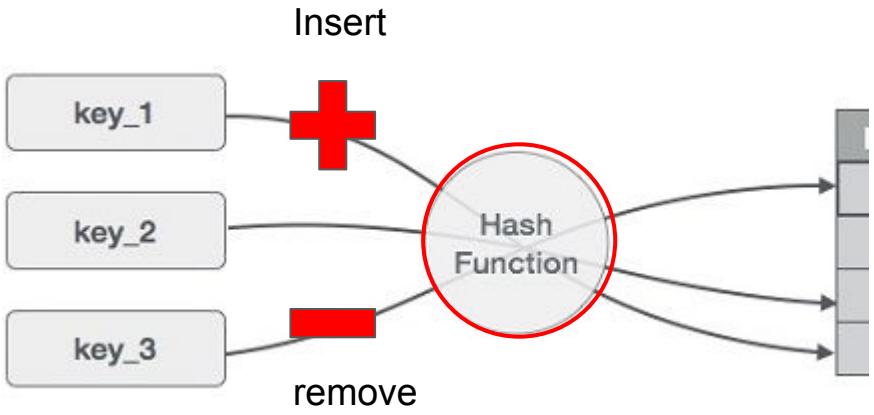
Hash Table Implementation: Data Structure

```
const int T_S = 200; // Table size

class HashMapTable {
   private:
      HashTableEntry **t;
   public:
      HashMapTable() {
         t = new HashTableEntry * [T_S];
         for (int i = 0; i< T_S; i++) {
               t[i] = NULL;
         }
      }
};
```



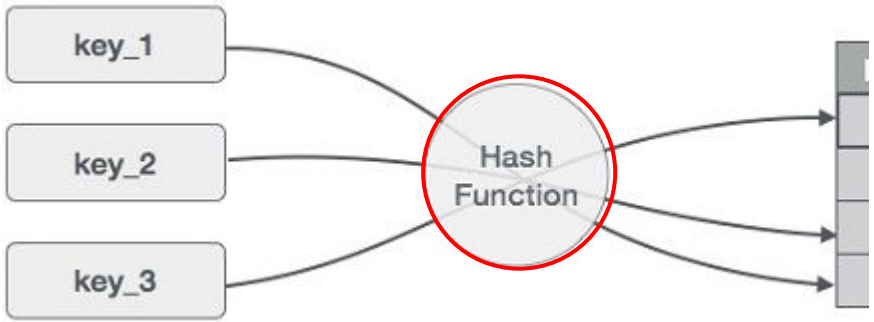
Hash Table Implementation: Data Structure

```
class HashMapTable {
  private:
    HashTableEntry **t;
  public:
    HashMapTable() {
      t = new HashTableEntry * [T_S];
      }
    }
     int HashFunc(int k);
    void Insert(int k, int v);
    int SearchKey(int k);
    void Remove(int k);
};
```



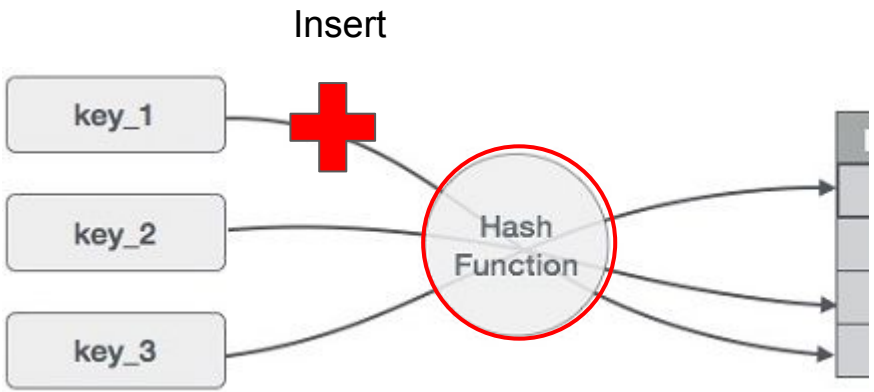
Hash Table Implementation: Hash Function

```
int HashFunc(int k) {
        return k % T_S; // mod table size
    }
```



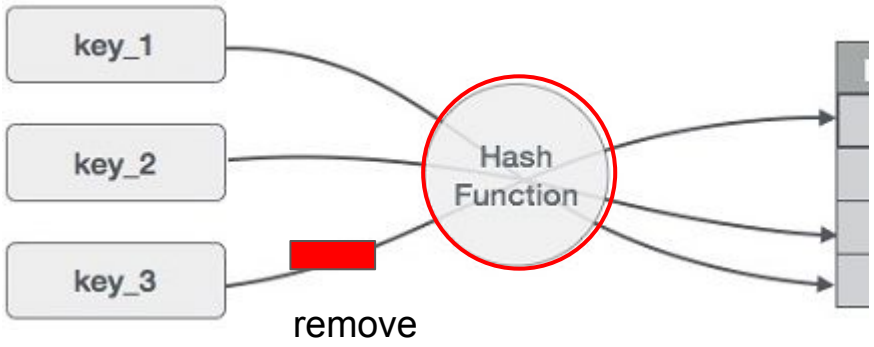
Hash Table Implementation: Insert

```
void Insert(int k, int v) {
    int h = HashFunc(k);

    while (t[h] != NULL && t[h]->k != k) {
        h = HashFunc(h + 1);
    }
    if (t[h] != NULL)
        delete t[h];
    t[h] = new HashTableEntry(k, v);
}
```



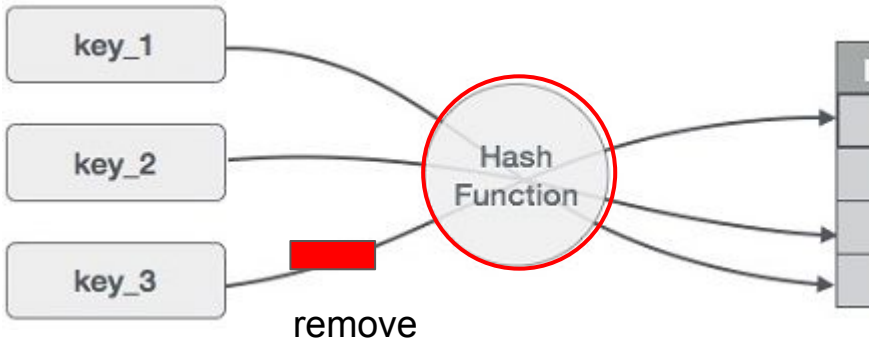
Hash Table Implementation: Remove

```
void Remove(int k) {
    int h = HashFunc(k);
    while (t[h] != NULL) {
        if (t[h]->k == k)
        break;

        h = HashFunc(h + 1);
    }
    if (t[h] == NULL) {
        return;
    } else {
        delete t[h];
    } }
```



Hash Table Implementation: Search

```
int SearchKey(int k) {
    int h = HashFunc(k);

    while (t[h] != NULL && t[h]->k != k) {
        h = HashFunc(h + 1);
    }
    if (t[h] == NULL)
        return -1;
    else
        return t[h]->v;
}
```



Hash Table Implementation: Discussions

- How to change the previous implementation to support integer keys and string values.

- How to change the previous implementation to support using user-defined hash function.

Hash Table Implementation: Discussions

- How to change the previous implementation to support integer keys and string values.

Possible solution:

```
class HashTableEntry {
    public:
        int k; // key
        int v; // value
        string str_v;
        HashTableEntry( int k, int v) {
            this->k= k;
            this->v = v;
        }
        HashTableEntry( int k, string v) {
            this->k= k;
            this->str_v = v;
        }};
```

Hash Table Implementation: Discussions

- How to change the previous implementation to support using user-defined hash function.

Possible solution: use Hash function as pointer to function

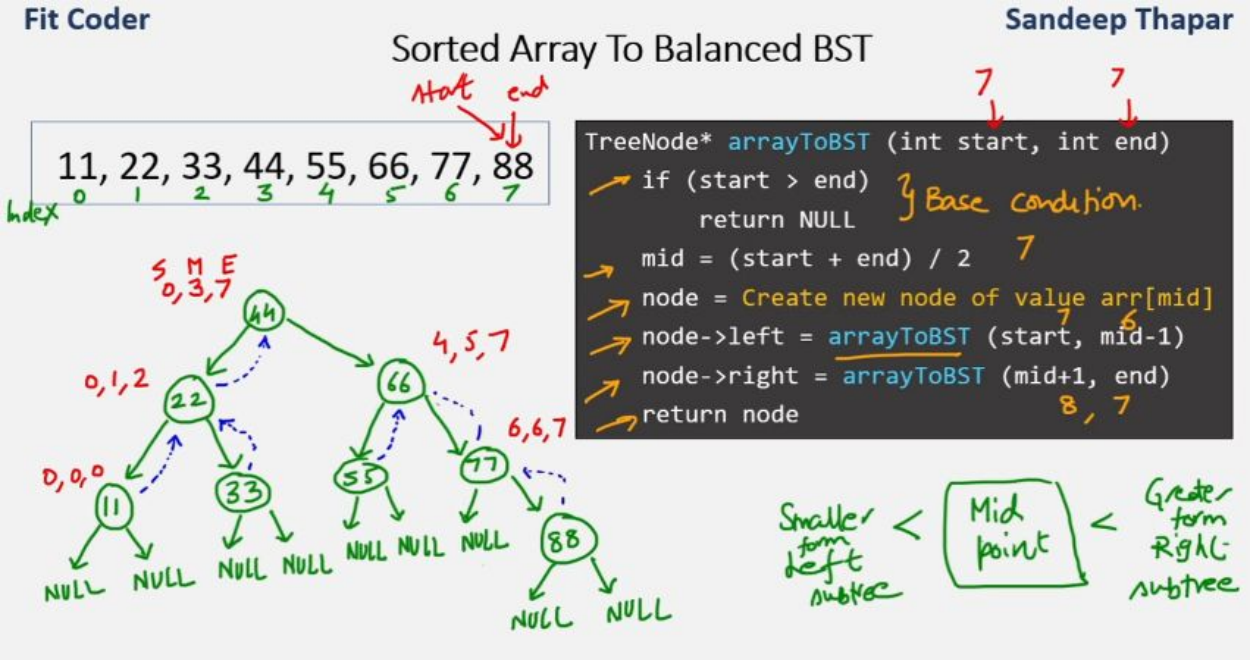
```
class HashMapTable {
   private:
      HashTableEntry **t;
      int (*hf)(int) = nullptr;
}

   void setHashFunc(int (*_hf)(int))
   {
       hf = _hf;
   }
```

```
int HashFunc(int k) {
   if(hf == nullptr) return k % T_S ;
   return hf(k); // mod table size
}

int myHashFunction(int key){
    cout<<"My hash Function"<<endl;
    return key % 10 ;
}
```

Hash Table Implementation: Discussions

- For the string hash functions, the size of the hash table limits the length of the string that can be hashed. (True or False ?)

Lab Project: Third week resources



https://www.youtube.com/watch?v=-FfprfvY4IQ