Hashing

Anh Huynh

1. Assume you have a simple single-dimensional array

```
array = [200, 400, 100, 50, 350]
```

Linear search will take O(N). Write a code in C++/Python to improve the search operation efficiency from O(N) to O(1). **4 pts**

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


int main(){
    // [200, 400, 100, 50, 350]
    // create a hash table named list
    unordered_map<int, string> list;

    // store key-value pairs in unordered
    list[200] = "one";
    list[400] = "two";
    list[100] = "three";
    list[50] = "four";
    list[350] = "five";

    // search for this key
    int key = 100;

    // if the key exists in the hash table, then print its value
    if (list.find(key) != list.end()){
        cout << "Key found: " << list[key] << endl;
    }
    // key doesn't exist
    else {
        cout << "Key not found" << endl;
    }

    return 0;

}
```

Output:
Key found: three

2. Use C++, generate hash value of your name. **1 pts**

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

int main(){
    string name = "Anh Huynh";
    hash<string> nameHash;

    cout << "Hashed Name: " << nameHash(name) << endl;

    return 0;
}
```

Output:
Hashed Name: 18314300974228188660

3. With the help of a figure, explain the problem that occured due to introducing a **tombstone** to mark the deleted cell. **5 pts**

For linear probing, if a key-value is deleted and the slot is simply marked as empty, searches for other keys may stop at that empty slot and give wrong results. To solve this, tombstone markers are placed instead of clearing the slot. A tombstone means “this spot used to hold a value, but now it’s deleted so keep probing.”

However, tombstones accumulate over time. They remain in the table until replaced by a new insertion. As a result, searches and insertions become slower because the probe sequence has to scan through many tombstones.

```
Initial table
Index: 0   1   2   3   4   5
Table: 5   10  15  20  --  --

Delete 15 and place a tombstone (T):
Index: 0   1   2   3   4   5
Table: 5   10  T   20  --  --

Delete more:
Index: 0   1   2   3   4   5
Table: T   10  T   20  T   --
       ^
           ^
               ^
```

One way to fix this is to rehash the table periodically by reinserting all live records into a new hash table. This removes the tombstones and restores fast performance.