Gmb. for Cally

Lecture - 60 (Hash map 1)

FREEMIND

Date

Page

→ Why hashing ?

→ Used to reduce the time complexity of searching

→ Finding a unique index for all elements to store them ↳ hash value

→ What is hashing ?

① Hash Values = 682, 761, 494, 567, 869 → 2, 1, 4, 7, 9

② $h(k) = k \bmod 10$

Hash Function

Search(761) ← O(1)

③ Hash table

0	
1	761
2	682
3	
4	494
5	
6	
7	567
8	
9	869

→ What are hash functions ?

1) Division method

$h(k) = k \bmod m$ → buckets

$m = 11$

$1276 \bmod 11 = 0$

2) Mid square Method

$h(k) = k^2$ & extract middle digits

$k = 60$ → $k^2 = 3600$

$k = 13$ → $k^2 = 169$

3) Digit folding Method

Fold key into equal size parts.

$K_1 K_2 K_3 K_4 K_5$

$K = 12345$

$12 + 34 + 5 = 51$ → hash value

FREEMIND

Date

Page

4. Multiplication Method → size of hash table

$$h(k) = \text{floor}(M(kA \bmod 1))$$

eg: $k = 12345$, $M = 10$

$A = 0.01$

$k * A = 123.45$

$kA \bmod 1 = (123.45 \bmod 1) = 0.45$

↳ $XM = 0.45 \times 10 = 4.5$

$\text{floor}(M(kA \bmod 1)) = \text{floor}(4.5) = 4$

⇒ Collisions :-

when 2 elements have same hash value.

--> Open hashing

Separate Chaining

Jha bhi 2 ya 2 se jyada elements ke hash Value same hogi to unhe uss hash value par LL bnaker add karenge

→ Close hashing (linear Probing)

c	
1	761
2	682
3	
4	494
5	634
6	794
7	567
8	
9	869

682, 761, 494, 567, 869, 634 (2), 794 (3)

$h(k) = k \bmod 10$

at $(h(k)+i) \bmod 10$ $\quad 0 \le i \le 9$

→ $h(794) = (4+0) \bmod 10 = 4$

$= (4+1) \bmod 10 = 5$

$= (4+2) \bmod 10 = 6$

FREEMIND

Date

Page

→ Close hashing (Open Addressing)

- **Quadratic Probing**

$$h(k) = k \bmod M$$

$$\text{at } (h(k) + i^2) \bmod M \qquad 0 \le i \le M$$

- **Double hashing**

$$h_2(k) = k \bmod 5$$
$$h_1(k) = k \bmod 10$$

store at $(h_1(k) + i\, h_2(k)) \bmod M$

$$\Rightarrow [h(634) = h(634) + 1 * h_2(634)] \bmod 10$$
$$\Rightarrow [4 + 1 \times 4] \bmod 10 = 8$$

⇒ Load Factor

n → no. of elements

m → no. of buckets

$$\text{Load factor} = \frac{n}{m}$$ → Avg entries in one bucket

⇒ Rehashing

↖ Load factor limit = 0.75

L.F > limit

→ Increasing size of hash table & redistributing elements in it.

FREEMIND

Date

Page

Q-> Implement hash table with closed addressing.
Separate chaining.

Ans->
```
#include <vector>
#include <list>
using namespace std;

class Hashing {
        Vector<list<int>> hashtable;
        int buckets;
public:
        Hashing (int size) {
            buckets = size;
            hashtable.resize (size);

        }
        int hashvalue (int key) {
            return key % buckets; //division method
        }
//add key   Void add (int key)
            int idx = hashvalue (key);
            hashtable[idx].push_back (key);
        }
//search key  list<int>::iterator search (int key) {
            int idx = hashvalue (key);
            return find (hashtable[idx].begin(), hashtable[idx].end()
                        , key);
        }
//delete key  Void delete (int key) {
            int idx = hashvalue (key);
            if (search (key) != hashtable[idx].end()) {
                hashtable[idx].erase (search (key));
                Cout << key << " is deleted" << endl;
            }
        }
```

FREEMIND
Date
Page

```
    else {
        cout << "key is not present in the hashtable";
    }
  }
};
int main() {
    Hashing h(10);
    h.addkey(5);
    h.addkey(9);
    h.addkey(3);

    h.deletekey(3);
    h.deletekey(3);

    return 0;
}
```

Lecture-61

Hash Map

Date
Page

⇒ Map in C++ STL
- → STL container which stores key-value pairs.
- → The elements are stored in ascending / descending order.
- → Maps cannot have duplicate keys.
- → implemented through BST

(A) Initialization

→ Header file required

```
#include <map>
```

→ Declaration

* By default order is ~~asc~~ ascending.

```
map<key_datatype, value_datatype> map_name
```

Descending order :- `map<datatype1, datatype2, greater<datatype1>> map_name`

→ Initialize :-

```
map<datatype1, datatype2> map_name = { {key1, value1},
                                       {key2, value2},
                                       {key3, value3} };
```

directory — Name, No.

→ Insertion :-

O(1) `directory.insert(make_pair("ABC", 379));`

→ Printing the elements :-

- for each loop

key value pair

```
for (auto element : map1) {
    key = element.first;
    value = element.second;
}
```

FREEMIND
Date
Page

⇒ erase() O(log n)

m.erase(itr)
m.erase(key)
m.erase(start_itr, end_itr) — O(n)

⇒ swap()

m1.swap(m2)
swap(m1, m2)

⇒ clear()

m.clear() → clear all elements

⇒ empty()

m.empty() → return 1 if empty

⇒ m.size()

⇒ m.max_size()

⇒ m.find(key) → return itr to element if present, else it returns map.end() itr
O(log n)

⇒ count() → no. of occurances of key

m.count(key) = 1

⇒ upper_bound() → returns an itr to next greater element.

⇒ lower_bound() → return itr to element if present, else itr to next greater element.

FREEMIND

Date

Page

⇒ begin() and end() / or rbegin() and rend()

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

int main() {
    map<string, int> directory;
    directory["naman"] = 38760;
    directory["animesh"] = 74562;
    directory["ritu"] = 12369; // updation if value already present

    for (auto element : directory) {
        cout << "Name - " << element.first << endl;
        cout << "Phone No.- " << element.second << endl;
    } cout << endl;

    directory.insert(make_pair("ritu", 4629)); // No change

    for (auto element : directory) {
        cout << "Name - " << element.first << endl;
        cout << "Phone No.- " << element.second << endl;
    }

// using iterator  map<string, int>::reverse_iterator itr;
    for (itr = directory.rbegin(); itr != directory.rend(); itr++) {
        cout << itr->first << " - " << itr->second << endl;
    }
    return 0;
}
```

Q) Sum of repetitive elements

You are given an integer n, representing the no. of elements. Then you will be given n elements. You have to return the sum of repetitive elements i.e. the elements that appear more than one time

FREEMIND

Date

Page

Input:
n = 7
Elements = {1,1,2,1,3,3,3}

Output: 4
The repetitive elements are 3, 1 and their sum is 1+3 = 4

Sol:

```
#include <map>
#include <vector>
using namespace std;

int main()
    int n;
    cin>>n;
```

Time O(n)

```
    Vector <int> v;
    for (int i=0; i<n; i++){
        cin>>v[i];
    }
```

Space O(m)

```
    map<int,int> m;          //sorting freq. of every element in input array.
    for (int i=0; i<n; i++){
        m[v[i]]++;
    }
```

{1,1} {1,2} {1,3} {2,1} {3,1} {3,2} {3,3}

```
    int sum=0
    for (auto pair : m){
        if (pair.second > 1){          //finding sum of repetitive elements.
            sum += pair.first;
        }
    }
    return 0;
}
```

FREEMIND

Date

Page

⭐ Unordered Map

→ STL containers, store key value pairs
→ elements are not ordered
→ Keys will be Unique

Insertion
Deletion
Retrieval/Search
} O(1) implemented using hashing

header file ⇒ #include <Unordered_map>

→ Member functions

m.insert()
m.erase()
m.find()
} O(1) avg
O(n) worst

Same functions as map

⭐ Multimap

→ duplicate keys are allowed.

Insert
Delete
Search
} O(log N) BST

same functions as map
same header file → #include <map>

multimap <int, int> m;

⭐ Unordered Multimap

→ elements are not ordered
→ duplicate keys are allowed
→ Hashing

Insertion
Deletion
Search
} O(1) Avg case
O(n) Worst case

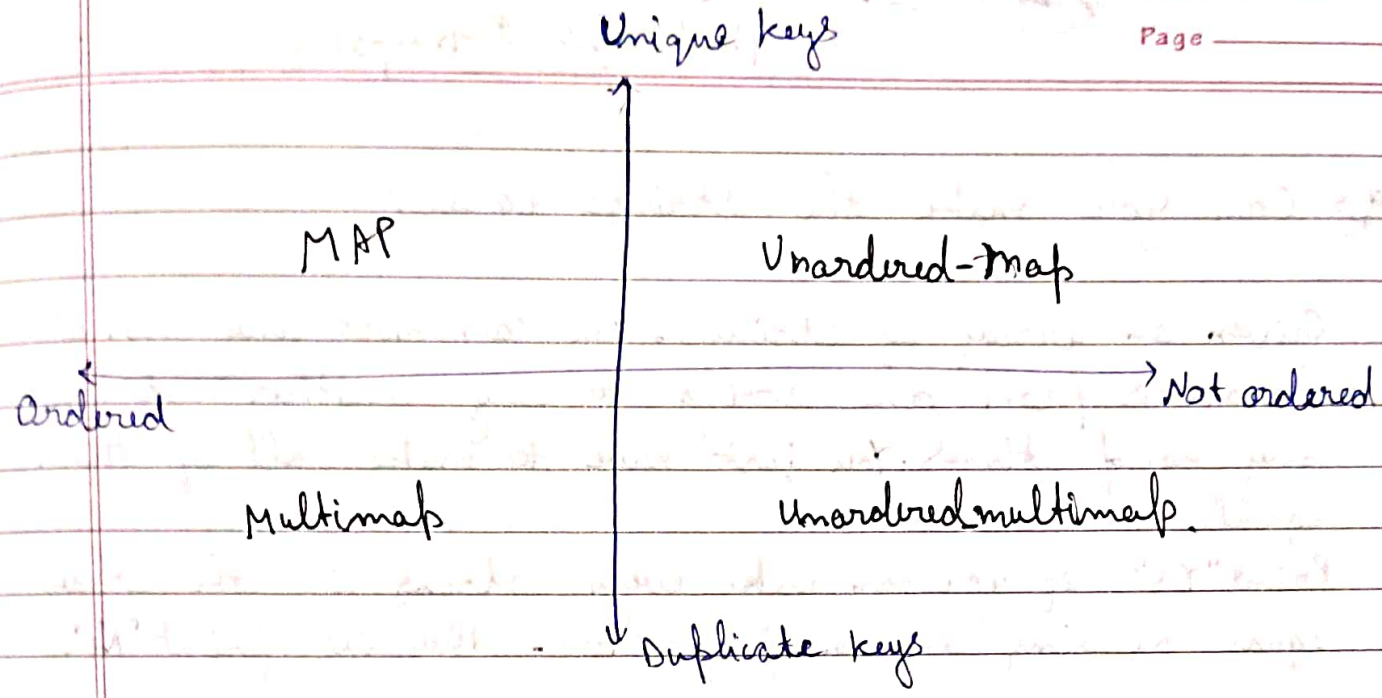
header file ⇒ #include <unordered_map>

unordered_multimap <string, int> Umap;

FREEMIND

Date

Page



Q4 Give Output:

```
Multimap<int,int> m;
m.insert({1,1});
m.insert({2,3});
m.insert({3,8});
m.insert({4,9});
m.insert({4,20})
m[5] = 25;      //Line 1
m[4] = 20;      // Line 2
```

(22) start. itr ⟶ end. itr ⟶ map

Ans

(C) compilation error; there is no [] operator in multimap.

Q23

```
auto a = equal_range(4); //returns bounds of range of elements with key passed.
for (auto it = a.first; it != a.second; it++) {
    cout << it->first << "-" << it->second << endl;
}
```

Ans23

(B) 4 - 16
4 - 20

Lecture-62

Hashmaps 3 (Problems)

Q1 => Can you make the strings equal?

Given an array of strings. You can move any no. of characters from one string to any another string any no. of times. You just have to make all of them equal.
Print "Yes" if you can make every string in the array equal by using any no. of operations. otherwise print "No".

Input: ["collegee", "coll", "collegge"]
Output: Yes
String at 1 index can take 2 'e' from 0 index string and one 'g' from 2 index string.

Concept => (1) We don't need to transfer strings by moving characters
(2) Each ~~char~~ character should be present in multiple of length of array.

Ans =>
```
#include <vector>
#include <unordered_map>
using namespace std;

bool canMakeEqual(vector<string>& v){

    unordered_map<char, int> m;
    for(auto str:v){
        for(char c:str){
            m[c]++;
        }
    }
}
```

FREEMIND
Date
Page

```
    int n = V.size();
    for(auto ele: m) {
        if(ele.second%n!=0) {
            return false;
        }
    }
    return true;
}
int main() {
    int n;
    cin>>n;

    Vector<string> V[n];
    for(int i=0; i<n; i++) {
        cin>> V[i];
    }

    cout<<(canMakeEqual(V)? "Yes" : "No")<<endl;

    return 0;
}
```

Q3 Check whether two strings are anagram of each other. Return true if they are else return false. An anagram of a string is another string that contains the same characters, only the order of characters can be different.
For example, "abcd" and "dabc" are an anagram of each other.
input1: triangle
integral
output:- true

FREEMIND

Date

Page

Methods to Solve

1. 2 arrays
2. 2 maps
3. 1 map

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

bool checkAnagrams (string s1, string s2) {
    if (s1.length() != s2.length()) { return false; }
    unordered_map<int> m;

    for (auto c1 : s1) {
        m [c1]++
    }
    for (auto c2 : s2) {
        if (m.find(c2) == m.end()) {
            return false;
        } else {
            m [c2]--;
        }
    }
    for (auto ele : m) {
        if (ele.second() != 0) {
            return false;
        }
    }
    return true;
}
int main
    s1, s2 → enter
    cout << (checkAnagrams (s1, s2) ? "Anagram" : "Not Anagram")
    return 0;
}
```

FREEMIND

Date

Page

Q) Check whether if two strings are isomorphic of each other.

Two strings are isomorphic of each other if there is a one to one mapping possible for every character of the first string to every character of the second string and all occurances of every character in first string maps to the same character in the second string.

input 1: a a a b

x x y

Output 1: True

Soln

```
#include <unordered>
#include <vector>
using namespace std;

bool checkIsomorphic (String S1, String S2) {

    if (S1.length() != S2.length()) {
        return false;
    }

    unordered_map <char, char> m;

    for (int i=0; i<S1.length(); i++) {
        if (m.find([S1[i]]) != m.end()) {
            if (m([S1[i]]) != S2[i])
                return false;
        } else {
            m([S1[i]]) = S2[i];
        }
    }
```

```
bool checkonetomanymapping
    (String S1, String S2) {

//checking one to
many mapping
from S1 → S2

bool S1S2 = checkonetomany mapping (S1, S2);

bool S2S1 = ——— (S2, S1);
```

FREEMIND

Date

Page

// { Now same for loop for S2 [change S2,S1] in code }

return S1.S2 && S2.S1

}

int main(){

string S1, S2;

cin >> S1 >> S2;

cout << (checkIsomorphic(S1,S2) ? "Yes" : "No");

return 0;

}

Q→ Given an array of length n and a target, return a pair whose sum is equal to the target. If there is no pair present, return -1.

Input 1: n = 7

Elements = [1, 4, 5, 11, 13, 10, 2]

Target = 13

output 1: [3, 6]

Method ① → Nested loops

i - 0 to n

j = i → n

a[i] + a[j]

② 2 Pointer approach

array should be sorted

1 2 4 5 10 11 13

↑ ↑ ↑ ↑ O(logn)

③ O(n) using map & storing previous elements.

FREEMIND

Date

Page

Sol →
```
#include <vector>
#include <unordered_map>
using namespace std;

vector<int> targetsumpair(vector<int> &V, int targetsum) {

    unordered_map<int, int> m;
    vector<int> ans{2,-1};

    for(int i=0; i<V.size(); i++)
        if(m[targetsum - V[i]] != m.end())
            ans[0] = m[targetsum - V[i]];
            ans[1] = i;
        } else {
            m[V[i]] = i;
        }
    }
    return ans;
}
int main() {
        n → Enter
        V → Enter
        targetsum → Enter
        vector<int> ans = targetsumpair(V, targetsum);
        cout << ans[0] << " " << ans[1];
        return 0;
}
```

Q → Given an array arr[] of length N, find the length of the longest subarray with a sum equal to 0.

input: n = 8
arr[] = {15, -2, 2, -8, 1, 7, 10, 23}

output: 5

FREEMIND

Date

Page

Method 1 → BF → Nested Loops

(2) $a[l \dots r] = 0$

$prefix_r - prefix_{l-1} = 0$

$prefix_r = prefix_{l-1}$

Time, Space
O(n²)

```
#include <vector>
#include <unordered_map>
using namespace std;

int maxlengthzerosumSubarray(vector<int> &V) {

    unordered_map<int, int> m;
    int prefix = 0;
    int maxlen = INT_MIN;
    for (int i=0; i < v.size(); i++) {
        prefix += V[i]
        if (prefix == 0) {
            maxlen++;
        }
        if (m.find(prefix) != m.end()) {
            maxlen = max(maxlen, i - m[prefix]);
        } else {
            m[prefix] = i;
        }
    }
    return maxlen;
}
int main() { no vector
    cout << maxlen(V) << endl;
}
```