Hash Map: Intuition and Implementation

Course on Basic Data Structures (C++)

Pulkit Chhabra • Lesson 14 • Feb 13, 2021

Hashing

Mapping elements to an integer in a way that if $e_1 = e_2$, then $hash(e_1) = hash(e_2)$

---

strings

```
int hash (string s)
{
    return 5;
}
```

An example of a bad / useless hash function.

We have a string. You need to store the frequency of each character that is there in the string. (s[i] → 'a' to 'z')

```
int cnt[26] = {0};

for (char ch : s)
    cnt[ch - 'a']++;
```

```
int hash(char ch)
{
    return ch - 'a';
}
```

Frequency of numbers in an array.

$a[i] \in [-100, 100]$

```
{ int freq[201] = {};
  for (int i : a)
    freq[i + 100] ++;
}
```

```
int hash(int x)
{
    return 100 + x;
}
```

An array of strings. ($len(s[i]) \leq 3$).

Store the freq. of each string

aba

bac

zxy

ab

int freq[27][27][27]

Look at these 2 scenarios:

1) $a[i] \in [-10^9, 10^9]$

2) $1 \leq \text{len}(s[i]) \leq 10^5$

1.) Have an array of size n;

2.) $hash(i) = i \% n$ $[0, n-1]$

Hash Maps

stores { key, value } pairs.

<int, int>

<string, int>

<string, string>

<double, string>

<int, vector<int>>

<string, vector<string>>

hash_function

$[0, size-1]$

1) Maps the key to some integer (i)

2) The {key, value} pair is stored at i.

Taking Care of Collisions :

1) Linear Probing

1	2	3	4	5	6	7	8	9	10
"kunal"				["Anand", 2]		["Kirti", 10]			

10

["akash", 2]

hash(s) → len(s) % 10

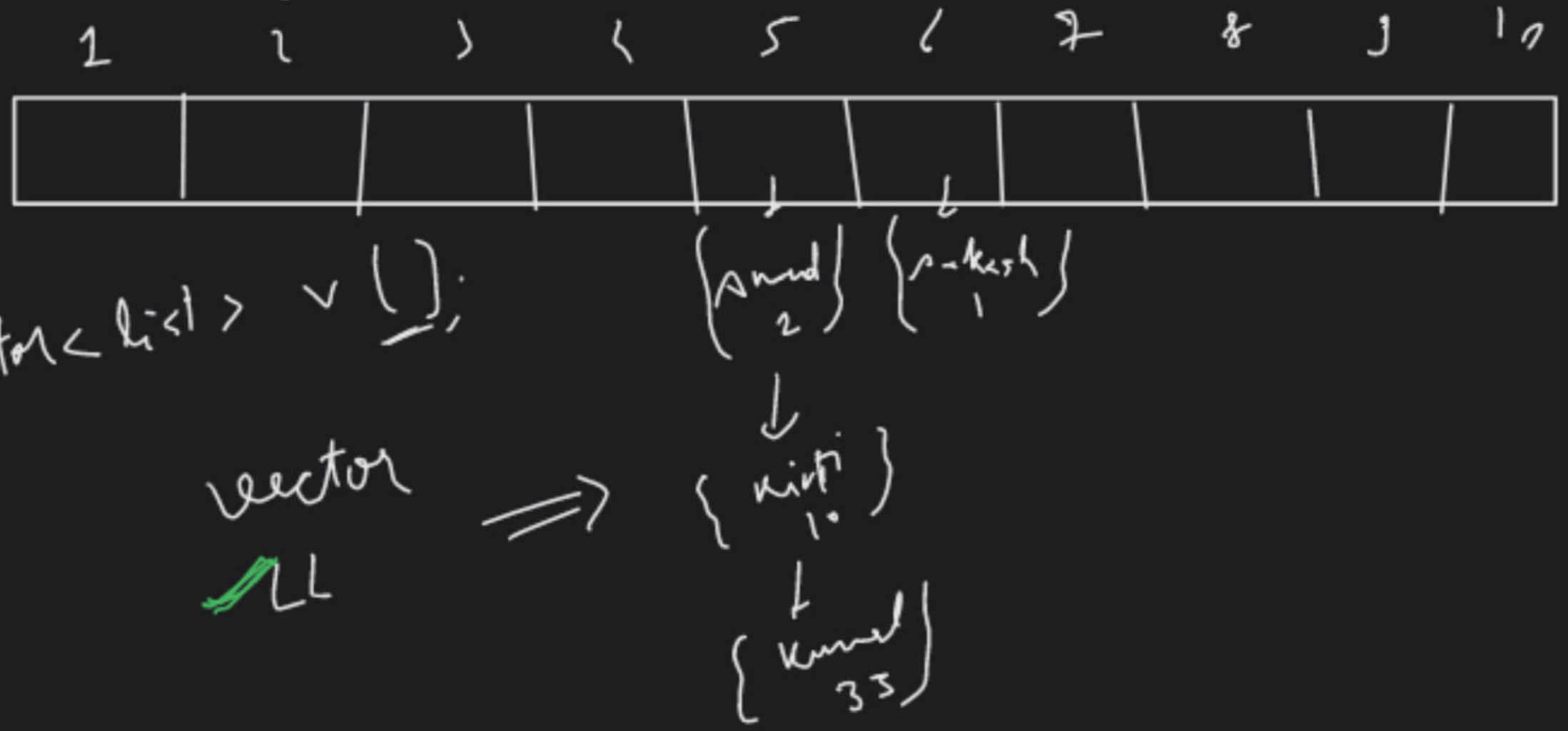
m["kunal"]

m["akash"] = 1

m["Anand"] = 2

m["Kirti"] = 10

2) Chaining / Bucketing



vector<list> v[];

vector

LL

1) Update / Insert $\leftarrow O(\text{len})$

2) Delete $\leftarrow O(\text{len})$

3.) Search $\leftarrow O(\text{len})$

$\Rightarrow$ Avg. Insert/Delete/Search $\Rightarrow O(1)$

1) Insert / Update

if we insert the key to its correct position if list was sorted.

$O(N*$ ___)

2.) Delete

BS is search ⇒ To delete $O(N)$

Terminologies

size → no. of keys in hm.

bucket_count ← no. of buckets in

load_factor ← size / buck_count → (Default 1.0)

max_load_factor ← buck_count is increased to

the smallest prime ≥ 2 × cur buck_cnt if lf > mx_lf

```
hash (string s)
{
    int ans = 0;
    for (char ch : s)
        ans += (int) ch;
    return ans % bucket_count;
}
```

Base ← p (prime > 256)

Polynomial Rolling Hash Function.

$$hash(s) = \left(s[0] * p^0 + s[1] * p^1 + s[2] * p^2 \ldots\ldots s[n-1] * p^{n-1} \right)$$

$\% \text{ bucket\_size}$.

unordered_set

Insert a value

Check if a value is present or not

Delete a value

No. of subarrays with sum = K.

l to $r \Rightarrow$ $pref[r] - pref[l-1] = K$

For r, calculate no. of indices i s.t. $i < r$ and $pref[i] = pref[r] - K$.

$[i+1, r]$

$\hookrightarrow$ Sum of this = Ans

Find no. of subArrays S s.t.

sum(S) = length(S)

[1 2 0 5]

Ans: 3