8/2 = 4/2 = 2 2/2 1

Hashing on Strings

CLASS - 12

29/10/23

1) $h(x) = \left(\sum_{i=0}^{N-1} str(i)\right) \% M$ abc bac

$x = 97+98+99$ $xy = 98+97+99$

$x = y$ → wrong

↓

with this 2 different strings have same hash value

→ collision.

2) $h(x) = \left(\sum_{i=0}^{N-1} str(i) * i\right) \% M$ abc bac

$(97)0 + (98)1 + 99(2)$ → $98(0) + 97(1) + 99(2)$

$98 + 198$ $= 97 + 198$

$\neq$ $a \neq b$.

+

but ab cb

$(97)0 + (98)1 + \dots : (99)0 + 98(1)$

$\Rightarrow b = cb$

128

2×64

~~3×4~~

4×32

8×16

3) $: \left(\sum_{i=0}^{N-1} str(i) * (i+1)\right) \% M$

~~aa~~ aab bba

$(97\times1 + 97\times2 + 98\times3)$ $(98\times1 + 98\times2 + 97\times3)$

585 ~~586~~

+

4) $\left(\sum_{i=0}^{N-1} str(i)\,(i+1)^2\right) \% M$

abcab

$97(1) + 98(4) + 99(9) + 97(16) + 98(25)$

$97 + 392 + 891 +$

bbacq

$98(1) + 98(4) + 97(9) + 99(16) + 98(25)$

$98 + 392 + 824 +$

X failing for few

Best Hash function

$$h(x) = \left(\sum_{i=0}^{N-1} str(i) * p^{(i+1)}\right) \% M$$

p = prime number

Finding frequency

Hashmap → Unordered map $N(I) + Q(I) \ldots N$

insert search space

Ordered (sorted) $N(\log_2 N) + Q(\log_2 N) \ldots N$

I (find position to insert)

S (find position) space.

a_N : 5 12 6 24 39 10 15

k = a + b.

a ≠ b.

Soln

1) N^2, 1 — 2 ptr

2) NlogN + N, 1
   sort — search

3) NlogN + (N * log N/2)
   sort — b.s.

1) Take sort.
2) Take each ele. (-6)
   -6 + k = ~~-6+27~~ 27 - (-6) = 33
   check for 33 in the list — logN
3) Repeat for all ele — N.

Hashmap.

1) Insert all ele in h.m

```
unordered_map mp;
for(int i=0; i<N; i++)
{ mp[arr[i]]++;

//search
for(int i=0; i<N; i++)
    int b= k-arr[i]
{   if map.containg
    if (map.find(b) != mp.end())
        return true;
    if(arr[i]==b && mp[b]>=2)
        return true
```

ele — N(1) + N(1), N : unordered map

Insert — search

⇒ If we add ~~compete~~ data to HM then it an ~~time~~ issue.

→ So we ~~ie~~ add elements only after checking with b. then it easy

Code: 2) HM

```
for(i=0 → N-1)
{
    b = k-arr[i]
    if (b in map)
        return ~~false~~ true.
    mp[arr[i]) += 1
}
return false
```

~~Insert check~~

N(1+1), N

↓ ↓

check insert map

3) Set. keep comparing if not there insert if present → print.

unordered_set<int>s;

{same as above

}

so insert(i)

N(1+1), N

N(logN + logN), N

↓

sorted

a_N: 10, -12, 13, 6, 4, -40, 16

Print max sum subarray can generate.

6+4+5+3+4+5

(0) (1) (2) (3) (2) (1)

$= 6\frac{(6+1)}{2} = N\frac{(N+1)}{2}$

$= \frac{6(7)}{2} = 24$

subseq → 2^N → Follow order

subset → 2^N → any order

```
for(int i=0; i<N; i++)
  { for(int j=i; j<N; j++)
    {
      for(int k=0; k<=j; k++)
        sum += ar[k]
    }
    ans = max(ans, sum)
  }
return ans;
}
```

2^N

0	1	2
21	31	19
0	0	0
0	0	1
0	1	0
0	1	1

T.C

$O(N * N * N), 1$

$N^2 * N, 1$

↓ generating arrays ↓ iterate over arrays.

Approach -2

1) Diff b/w previous and current subarray is 1 element. So just add the new element to the previous sum.

```
ans = -∞, sum = 0
for(int i=0; i<N; i++)
{
    sum = 0
    for(int j=i; j<N; j++)
    {
        sum += ar[j];
        ans = max(ans, sum);
    }
}
```

$N^2, 1$

Approach -3

Dynamic Prog

PROBLEM

	0	1	2	3	4	5	6	7
arr:	12	-5	17	19	-3	5	16	24

Non decreasing subsequences.

Consider small arr

0	1	2	
12	-6	20	
2	1	0	
0	0	1	{12}
0	1	0	{-6}
0	1	1	{12, -6}
1	0	0	{20}
1	0	1	{12, 20}
1	1	0	{-6, 20}
1	1	1	{12, -6, 20}

$2^3 - 1$

```
int c=0
for (int i=1 ; i<(1<<N); i++)
    {  if (subsequence (i, arr, N))
            {
                c++;
            }
    }
return c;
```

```
bool subsequence (int i, int arr, int N)
{
  for(int j=0; j<32; j++)
  {
    if ((i&j) == i)
      return T
  }
}
```

```
bool subsequence (int i, int arr, int N)
  int prev = -∞
  for (int j=0; j< N; j++)
    { if (checkBit (i, j))
        { if (prev <= arr[j])
            prev = arr[j]
          else
            return false.
        }
    }
  return true;
```

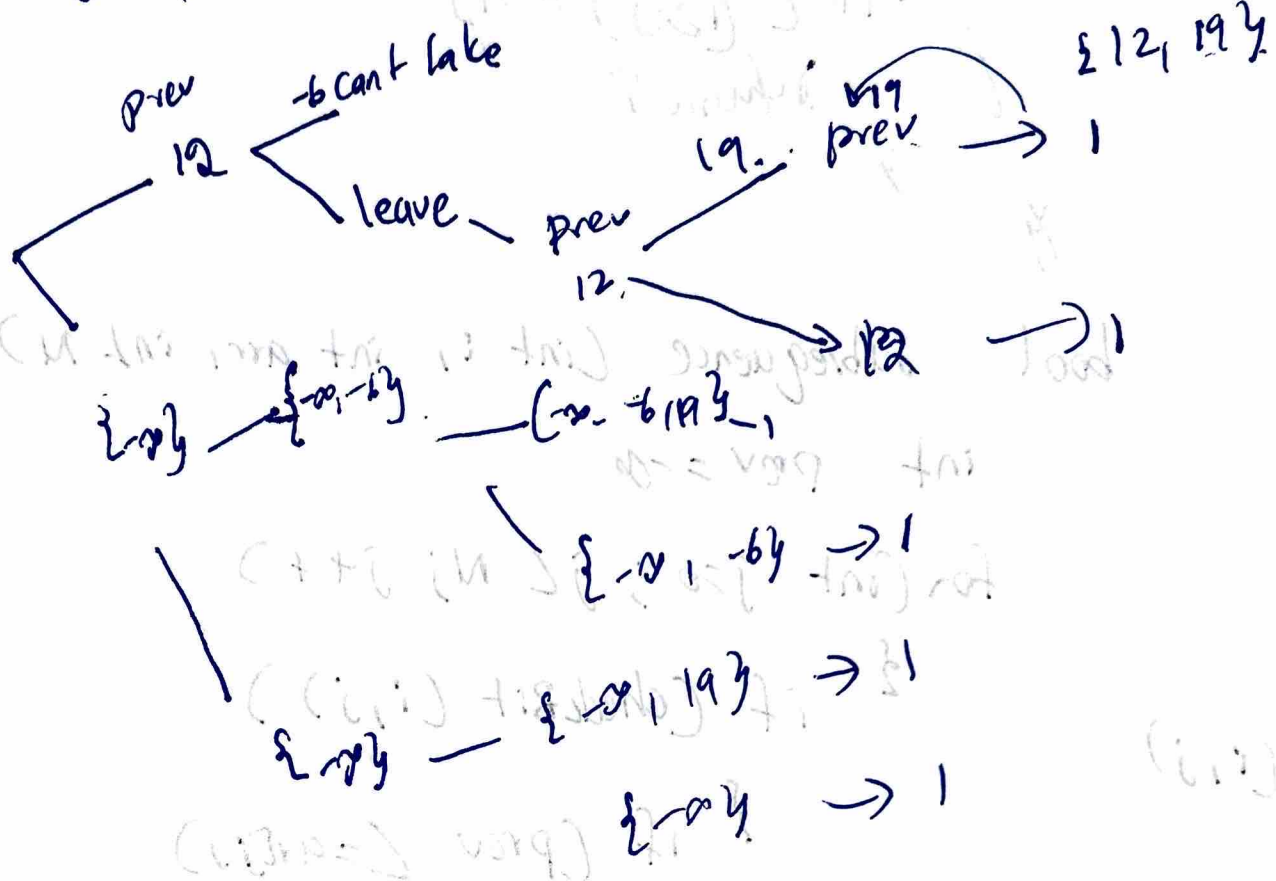
checkbit(i, j)
if(i

Comp :

Approach - 2

12 -6 19.

every element → take (or) leave.



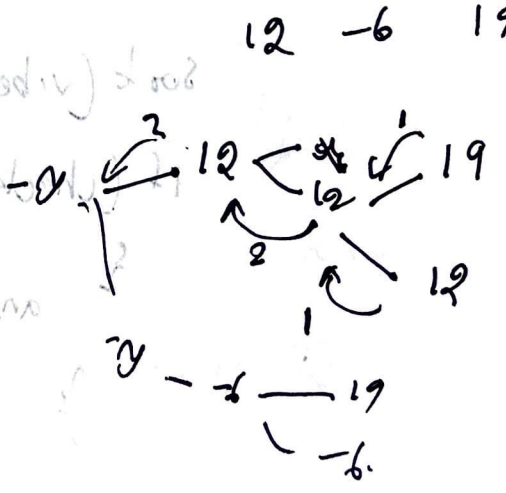
Code

```
int subsequence (int arr[], int N, int idx, int prev)
{
    prev = -∞
    if (idx == N+1)
    { if (prev < arr[idx])
        prev = arr[idx].
    }
    Subsequence (arr, N, idx+1, prev)
    ||
    (arr, N, idx+

    if (idx == N)
    { return 1
    }

    if (Prev ≤ arr[idx])
    { Prev = arr[idx].
      return Subsequence (arr, N, idx+1, arr[idx].
             + subsequence (arr, N, idx+1, prev)
    }
    else
    { return subsequence (arr, N, idx+1, prev)
    }
}
```

$O(2^N, 1)$



PROBLEM

0	1	2	3	4	5	6	7	8	9	10
12	14	15	13	20	9	7	5	6	12	13

ar_N:

Find the maximum length of the subarray which can be rearranged in a contiguous order.

diff: 1
No duplicates

Best longest → 12 14 15 13

Soln // 2 for loops i, j
sort each subarray

```
for (int i=0; i<N; i++)
{
  for (int j=i; j<N; j++)
  {
    vector<int> v;
    for (int k=i; k<=j; k++)
      v.push_back(ar[k]);
    sort(v.begin(), v.end());
    if (check(v))
    {
      ans = max(ans, j-i+1)
    }
```

$O(N^2)$

$N \log N$

```
bool check(v)
{
  for (int i=0; i<len(v)-1; i++)
  {
    if (v[i] != v[i+1]-1)
      return false
  }
  return true
}
```

$O(N)$

Comp: $O(N^2)$

$N^2(N + N\log N + N)$, N

2 loops → storing → sort → check → dynamic array.

optimization : instead do the insertion sort

Approach-2

```
for(int i)                                  (Array - forward)
{ for(int j) vector v;
    v.push insertion(v, arr[j])   // for loop goes.
  y
  if check(v))
  {
     ans = max(ans, j-i+1)
  }
```

$N^2(N + N)$, N dynamic array

insertion → check

Approach-3

```
for(int i)
{ for(int j=i; j<N; j++)
    { min=+∞ , max=-∞
      for(int k=i; k<=j; k++)
        { m = min(min, arr[k]), Max = max(max, arr[k])
          if(M-m+1 == j-i+1)
          [return M-m) ans = max(ans, M-m+1)
```

subarray → $N^2 * (N)$, 1 ← iterating

2 loops → k loop

Approach-4 Min max with Carry Forward

```
for (i -> N)
{
    m = +∞ , M = -∞
    for (j = i -> N)
    {
        m = min(m, arr[j]) , M = max(M, arr[j])
    }
    if (M - m + 1 == j - i + 1)
        ans = max(ans, j - i + 1)
}
```