→ Contest problems discussions

→ Mock interview

① Check String Acronym

You are given an array of N strings A and another string B. The acronym of the array A is formed by concatenating the first characters from each of the strings in A in the same order. Check whether the string B is equal to the acronym of the array of strings A.

A[] → ["hello", "iam", "hi", "owl", "what"]

0 1 2 3 4

↑

i

⇓

hihow

B → "hihow"

01234

#code →

```
if ( B.length () != N) { return false }

for( i = 0; i< N; i++){
    if( A[i]. charAt(0) != B.charAt(i)){
        return false;
    }
}

return true;
```

T.C → O(N)

S.C → O(1)

② Intersection of two arrays.

You are given two integer arrays A and B of size N and M respectively. Return an array of their intersection. Each element in the result must be unique and you may return the result in any order.

A[] → [2, 3, 1, 3, 5]

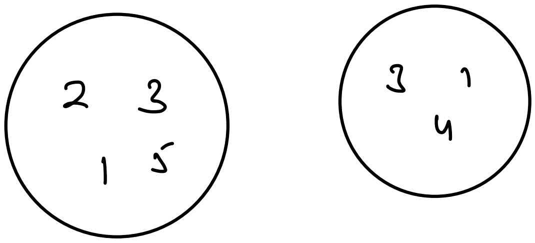
B[] → [3, 1, 4]

A[] ∩ B[] → [3, 1]

A[] → [1, 2, 2, 1]

B[] → [2, 2]

A[] ∩ B[] → [2]



⇒ [3, 1]

idea. → Create (h1, h2) hashsets for both the arrays. For every element in h1, check if that element is present in h2 or not.

code. →

```
Hashset<int> h1 , h2 ;

for ( i = 0; i < N ; i++) {
            h1.add ( a[i]);
}

for ( i = 0; i < m; i++) {
            h2.add ( b[i]);
}

   list<int> ans ;

   for( int val : h1) {
         if ( val is present in h2) {
                  ans.insert( val);
         }
   }

   return ans;
```

T.C → O(N+m)
S.C → O(N+m)

③ Find minimum element in sorted rotated array →

log N

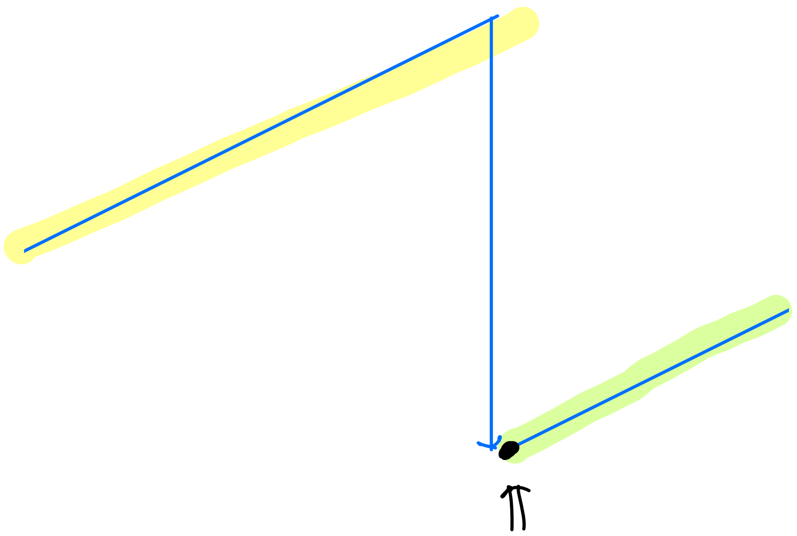
→ B.F → linear search   T.C → O(N)

→ idea.2. → Binary Search

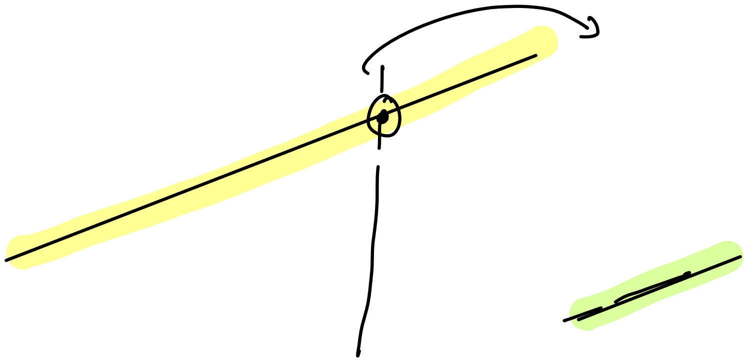
search space → [0, N-1]

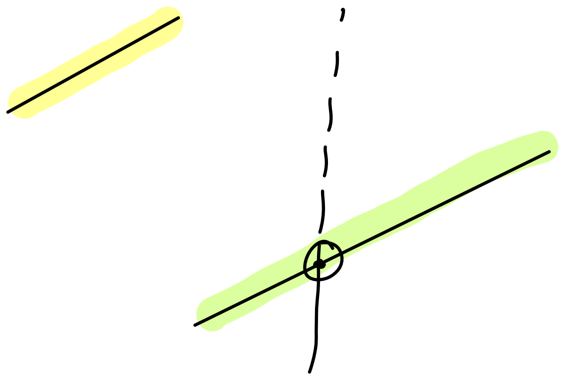
target → min. element

condn →



first element in 2nd part of the array.





arr[] =	[70	80	100	2	3	4	20	50	60]
	0	1	2	3	4	5	6	7	8

↑ l ↑ r

l	r	mid	middle element in which part?	

0	8	$\frac{0+8}{2} = 4$	2^{nd}	ans = 3
				r = mid-1

0	3	$\frac{0+3}{2} = 1$	1^{st}	l = mid+1
2	3	$\frac{2+3}{2} = 2$	1^{st}	l = mid+1

3	3	$\frac{3+3}{2} = 3$	2^{nd}	ans = 2
				r = mid-1

3	2			

code :-

```
if (N==1) {return arr[0]}
else if ( arr[0] < arr[N-1]) { return arr[0]} ;

l = 0, r = N-1 , ans = -1

while( l <= r){
        int mid = (l+r)/2;
        if ( arr[mid] >= arr[0] ){   //middle element in 1st half
              l = mid+1 ;
        }
        else{
              ans = arr[mid]
              r = mid-1 ;
        }
}
return ans ;
```

T.C → $O(\log_2 N)$

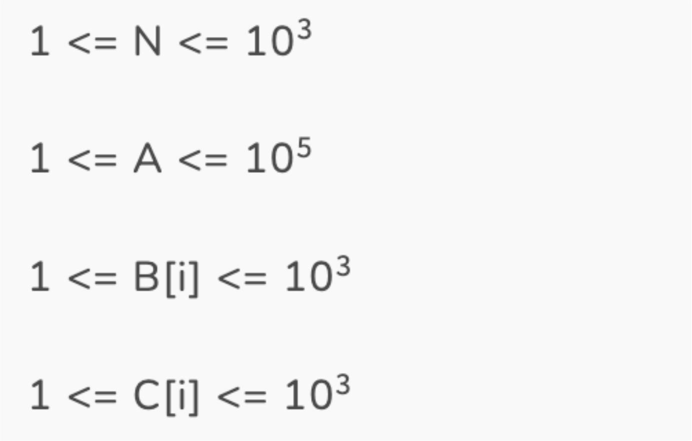
S.C → O(1)

④ Bob and chocolates →

You are in a chocolate shop that sells N number of different chocolates. You are given that the price of each chocolate is B[i] and the sweetness of each chocolate is C[i].

You have decided that the total price of your purchases will be atmost A. You can buy each chocolate at most once. What is the maximum sweetness we can get using atmost A rupees?

Please read the examples given below carefully to better understand the



$1 <= N <= 10^3$

$1 <= A <= 10^5$

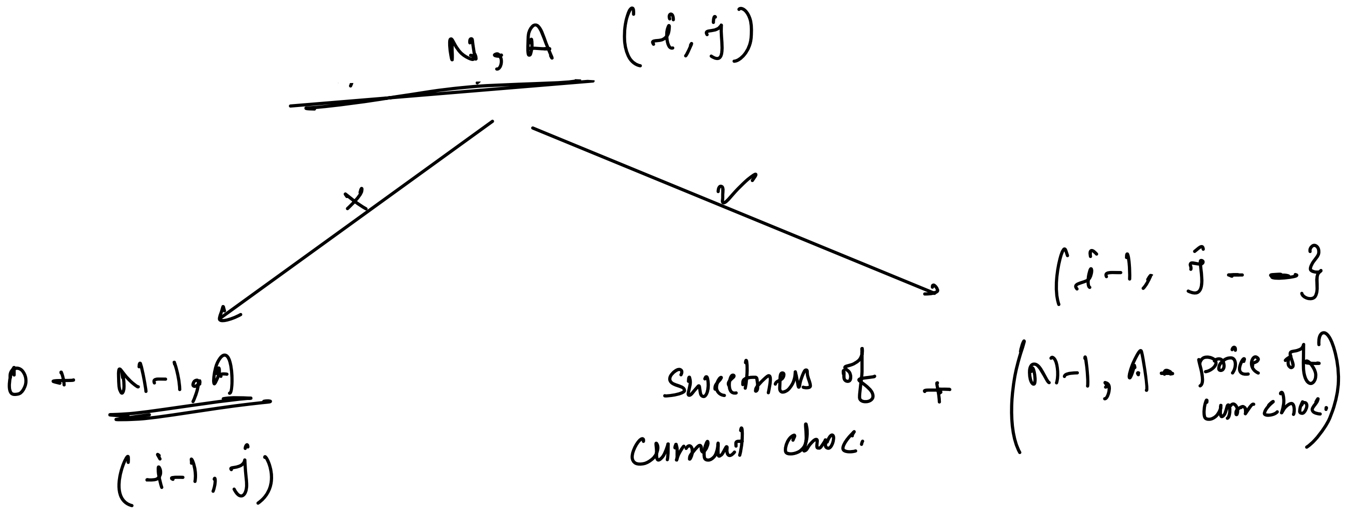
$1 <= B[i] <= 10^3$

$1 <= C[i] <= 10^3$

A → 10

B[] → [4, 8, 5, 3] (price)

C[] → [5, 12, 8, 1] (sweetness)



$dp[N+1][A+1] \longrightarrow 10^3 \times 10^5 \rightarrow 10^8.$

A → 10

B[] → [4, 8, 5, 3] ✓

C[] → [5, 12, 8, 1]

	i \ j	0	1	2	3	4	5	6	7	8	9	10
	0	0	0	0	0	0	0	0	0	0	0	0
5, 4	1	0	0	0	0	5	5	5	5	5	5	5
12, 8	2	0										
8, 5	3	0										
1, 3	4	0										

	0	1	2	3	4	5	6	7	8	9	10
dp[] →	0	0	0	0	~~0~~5	~~0~~5	~~0~~5	~~0~~5	~~0~~5	~~0~~5	~~0~~5

code ·→

```
dp[A+1] ;     ∀i, dp[i] = 0

for ( i = 1 ; i ≤ N ; i++) {
    for( j = A ; j ≥ 1 ; j--){
        if ( B[i-1] ≤ j ){
            dp[j] = Max( dp[j] , C[i-1] + dp[j-B[i-1]]);
        }
    }
}
return dp[A].
```

T.C → O(N*A)
S.C → O(A)

⑤ Rotten Oranges

Given a matrix of integers A of size N x M consisting of 0, 1 or 2.

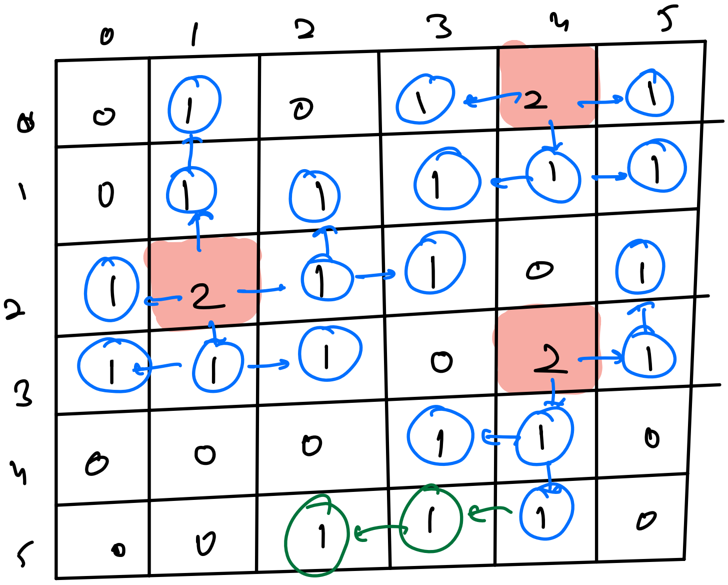
Each cell can have three values:

The value 0 representing an empty cell.

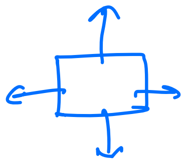
The value 1 representing a fresh orange.

The value 2 representing a rotten orange.

Every minute, any fresh orange that is adjacent (Left, Right, Top, or Bottom) to a rotten orange becomes rotten. Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return -1 instead.



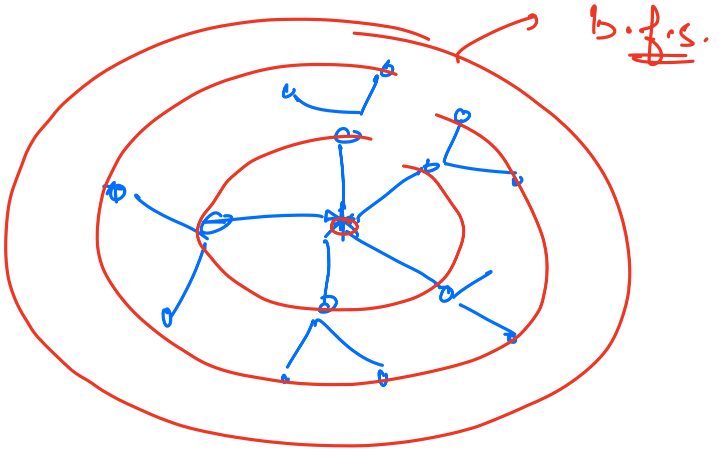
	0	1	2	3	4	5
0	0	1	0	1	2	1
1	0	1	1	1	1	1
2	1	2	1	1	0	1
3	1	1	1	0	2	1
4	0	0	0	1	1	0
5	0	0	1	1	1	0



t = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

0	1	1	0	0
1	1	1	0	0
1	2	1	0	1
1	1	0	0	0

ans = -1



	0	1	2	3	4	5
0	~~1~~ 2	~~1~~ 2	~~1~~ 2	0	2	2
1	0	~~1~~ 2	~~1~~ 2	2 ~~1~~	2 ~~1~~	~~1~~ 2
2	0	~~1~~ 2	~~1~~ 2	0	0	0
3	2 ~~1~~	2	~~1~~ 2	0	~~1~~ 2	~~1~~ 2
4	2 ~~1~~	0	0	0	2	~~1~~ 2
5	0	~~1~~ 2	~~1~~ 2	~~1~~ 2	~~1~~ 2	~~1~~ 2

8

~~(0,4)~~	~~(3,1)~~	~~(4,4)~~	~~(0,5)~~	~~(1,4)~~	~~(2,1)~~	~~(3,0)~~	~~(3,2)~~	~~(3,4)~~	~~(5,4)~~	(4,5)

$\underbrace{\qquad}_{t=0}$ $\underbrace{\qquad}_{t=1}$

~~(1,5)~~	~~(1,3)~~	~~(1,1)~~	~~(2,2)~~	~~(4,0)~~	~~(3,5)~~	~~(5,3)~~	~~(5,5)~~	~~(1,2)~~	~~(0,1)~~	~~(5,2)~~	~~(0,2)~~	~~(0,0)~~	~~(5,1)~~

$\underbrace{\qquad}_{t=2}$ $\underbrace{\qquad}_{t=3}$ $\underbrace{\qquad}_{t=4}$ t=5