

K Sai Dithvika CSE

1. Kth Smallest Element

Python:

```
class Solution:
    def kthSmallest(self, arr, k):
        return sorted(arr)[k-1]
```

- **Time Complexity:** ($O(n \log n)$)
- **Space Complexity:** ($O(n)$)

2. Minimize the Heights II

Python:

```
class Solution:
    def getMinDiff(self, arr, k):
        arr.sort()
        n = len(arr)
        ans = arr[n-1] - arr[0]
        smal = arr[0] + k
        larg = arr[n-1] - k

        for i in range(n-1):
            min_val = min(smal, arr[i+1] - k)
            max_val = max(larg, arr[i] + k)
            if min_val < 0:
                continue
            ans = min(ans, max_val - min_val)
        return ans
```

- **Time Complexity:** ($O(n \log n)$)
- **Space Complexity:** ($O(1)$)

3. Binary Search

Python:

```
class Solution:
    def binarysearch(self, arr, k):
        a = len(arr)
        low = 0
        high = a - 1
        while low <= high:
            mid = low + (high - low) // 2
            if arr[mid] == k:
                return mid
            elif arr[mid] < k:
                low = mid + 1
            else:
                high = mid - 1
        return -1
```

- **Time Complexity:** ($O(\log n)$)
- **Space Complexity:** ($O(1)$)

4. Union of Two Arrays

Python:

```
class Solution:
    def findUnion(self, a, b):
        return len(set(a) | set(b))
```

- **Time Complexity:** ($O(m + n)$)
- **Space Complexity:** ($O(m + n)$)

5. Next greater elements in the array

```
class Solution:
    # Function to find the next greater element for each element of the
    array.
    def nextLargerElement(self, arr):
```

```
# code here
n =len(arr)
stack = []
res = [-1] *n
for i in range(n-1,-1,-1):
    while stack and stack[-1] <= arr[i]:
        stack.pop()
    if stack:
        res[i]= stack[-1]
    stack.append(arr[i])
return res
```