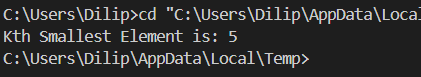
DSA DAY 4

1. K SMALLEST ELEMENT
2. #include <bits/stdc++.h>
3. using namespace std;
4. int kthSmallest(int arr[], int N, int K)
5. {
6. priority\_queue<int> pq;
7. for (int i = 0; i < N; i++) {
8. pq.push(arr[i]);
9. if (pq.size() > K)
10. pq.pop();
11. }
12. return pq.top();
13. }
14. int main()
15. {
16. int N = 10;
17. int arr[N] = { 10, 5, 4, 3, 48, 6, 2, 33, 53, 10 };
18. int K = 4;
19. cout << "Kth Smallest Element is: "
20. << kthSmallest(arr, N, K);
21. }



TC-O(N LOG K)

2)MINIMIZE HEIGHT 2

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int getMinDiff(vector<int>& arr, int k) {

    int n = arr.size();

    if (n == 1) {

        return 0;

    }

    sort(arr.begin(), arr.end());

    int ans = arr[n - 1] - arr[0];

    int smallest = arr[0] + k;

    int largest = arr[n - 1] - k;

    for (int i = 0; i < n - 1; i++) {

        int minHeight = min(smallest, arr[i + 1] - k);

        int maxHeight = max(largest, arr[i] + k);

        if (minHeight < 0) continue;

        ans = min(ans, maxHeight - minHeight);

    }

    return ans;

}

int main() {

    vector<int> arr = {1, 5, 8, 10};

    int k = 2;

    int result = getMinDiff(arr, k);

    cout << "The minimum difference is: " << result << endl;

    return 0;

}



TC-O(N LOG N)

3)PARANTHESES CHECKE  
#include <bits/stdc++.h>

using namespace std;

bool checkMatch(char c1, char c2){

    if (c1 == '(' && c2 == ')') return true;

    if (c1 == '[' && c2 == ']') return true;

    if (c1 == '{' && c2 == '}') return true;

    return false;

}

bool ispar(string s){

    int top = -1;

    for (int i = 0; i < s.length(); ++i){

        if (top < 0 || !checkMatch(s[top], s[i])){

            ++top;

            s[top] = s[i];

        }

        else{

            --top;

        }

    }

    return top == -1;

}

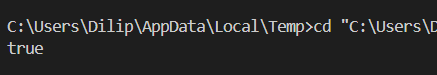
int main(){

    string s = "{()}[]";

    cout << (ispar(s) ? "true" : "false") << endl;

    return 0;

}



TC-O(N)

4)EQUILIBRIUM POINT

#include <iostream>

#include <vector>

using namespace std;

int equilibriumPoint(vector<long long>& arr) {

    if (arr.size() == 0 || arr.size() == 1 || arr.size() == 2) {

        return 1;

    }

    long long sum = 0;

    for (long long num : arr) {

        sum += num;

    }

    long long sum1 = 0;

    sum -= arr[0];

    for (int i = 1; i < arr.size(); i++) {

        sum1 += arr[i - 1];

        sum -= arr[i];

        if (sum1 == sum) {

            return i + 1;

        }

    }

    return -1;

}

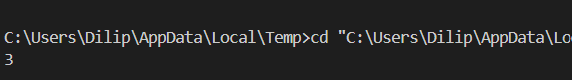
int main() {

    vector<long long> arr = {1, 3, 5, 2, 2};

    cout << equilibriumPoint(arr) << endl;

    return 0;

}



TC-O(N)

5)BINARY SEARCH

#include <bits/stdc++.h>

using namespace std;

int binarySearch(int arr[], int low, int high, int x)

{

    while (low <= high) {

        int mid = low + (high - low) / 2;

        if (arr[mid] == x)

            return mid;

        if (arr[mid] < x)

            low = mid + 1;

        else

            high = mid - 1;

    }

    return -1;

}

int main(void)

{

    int arr[] = { 2, 3, 4, 10, 40 };

    int x = 10;

    int n = sizeof(arr) / sizeof(arr[0]);

    int result = binarySearch(arr, 0, n - 1, x);

    if(result == -1) cout << "Element is not present in array";

    else cout << "Element is present at index " << result;

    return 0;

}



TC-O(LOG N)

6)NEXT GREATER ELEMENT

#include <bits/stdc++.h>

using namespace std;

void printNGE(int arr[], int n)

{

    stack<int> s;

    s.push(arr[0]);

    for (int i = 1; i < n; i++) {

        if (s.empty()) {

            s.push(arr[i]);

            continue;

        }

        while (s.empty() == false && s.top() < arr[i]) {

            cout << s.top() << " --> " << arr[i] << endl;

            s.pop();

        }

        s.push(arr[i]);

    }

    while (s.empty() == false) {

        cout << s.top() << " --> " << -1 << endl;

        s.pop();

    }

}

int main()

{

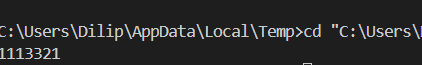
    int arr[] = { 11, 13, 21, 3 };

    int n = sizeof(arr) / sizeof(arr[0]);

    printNGE(arr, n);

    return 0;

}



TC-O(N)

6)UNION OF SETS WITH DUPLICATES

#include <iostream>

#include <vector>

#include <unordered\_set>

using namespace std;

int findUnion(vector<int>& a, vector<int>& b) {

    unordered\_set<int> unionSet;

    for (int num : a) unionSet.insert(num);

    for (int num : b) unionSet.insert(num);

    return unionSet.size();

}

int main() {

    vector<int> a = {1, 2, 3, 4, 5};

    vector<int> b = {2, 3, 5, 7};

    cout << findUnion(a, b) << endl;

    return 0;

}



TC-O(N+M)