DSA Practice-8 (20-11-2024)

Name: Atchayaa T

Department: B.Tech CSBS

Register No.: 22CB004

1. 3 Sum Closest

```
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        Arrays.sort(nums);
        int ans = nums[0] + nums[1] + nums[2];

        for (int i = 0; i < nums.length - 2; i++) {
            int l = i + 1, r = nums.length - 1;
            while (l < r) {
                int c = nums[i] + nums[l] + nums[r];

                if (c == target) {
                    return target;
                }

                if (Math.abs(c - target) < Math.abs(ans - target)) {
                    ans = c;
                }

                if (c < target) {
                    l++;
                } else {
                    r--;
                }
            }
        }

        return ans;
    }
}
```

Time Complexity: O(n^2)

2. Jump Game II

```
class Solution {
public:

    int jump(vector<int>& nums) {

      for(int i = 1; i < nums.size(); i++)
      {
        nums[i] = max(nums[i] + i, nums[i-1]);
      }
      int ind = 0;
      int ans = 0;

      while(ind < nums.size() - 1)
      {
        ans++;
        ind = nums[ind];
      }

      return ans;
    }
};
```

Time Complexity: O(N)

3. Group Anagrams

```
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String, List<String>> d = new HashMap<>();
        for (int i=0;i<strs.length;i++){
            int[] freq = new int[26];
            for (char c : strs[i].toCharArray()) {
                freq[c - 'a']++;
            }
            StringBuilder s = new StringBuilder();
            for (int j=0;j<freq.length;j++){
                s.append(freq[j]).append('#');
            }
            String key = s.toString();
            if (!d.containsKey(key)){
                d.put(key,new ArrayList<>());
            }
            d.get(key).add(strs[i]);
        }

        return new ArrayList<>(d.values());
    }
}
```

Time Complexity: O(n*k)

4. Decode Ways

```
class Solution {
  public int numDecodings(String s) {
      int strLen = s.length();

      int[] dp = new int[strLen + 1];

      dp[0] = 1;

      if (s.charAt(0) != '0') {
          dp[1] = 1;
      } else {
          return 0;
```

```
        }


        for (int i = 2; i <= strLen; ++i) {

            if (s.charAt(i - 1) != '0') {
                dp[i] += dp[i - 1];
            }

            if (s.charAt(i - 2) == '1' ||
                    (s.charAt(i - 2) == '2' && s.charAt(i - 1) <= '6')) {
                dp[i] += dp[i - 2];
            }
        }

        return dp[strLen];
    }
}
```

Time Complexity: O(N)

5. Buy and Sell Stock II

```
import java.util.List;

class Solution {
    public int maxProfit(List<Integer> prices) {
        int c = 0;
        int s = prices.get(0);
        for (int i = 0; i < prices.size(); i++) {
            if (s < prices.get(i)) {
                int prof = prices.get(i) - s;
                c += prof;
            }
            s = prices.get(i);
        }
        return c;
    }
}
```

Time Complexity: O(n)

6. Number of Islands

```
class Solution {
public:
    bool isvalid(int i,int j, int row, int col, vector<vector<char>>& grid){
        if (i>=0 && i<row && j>=0 && j<col && grid[i][j]=='1'){
            return true;
        }
        return false;
    }
    void makezero(int i, int j, int row, int col,vector<vector<char>>& grid){
        grid[i][j]='0';
        if (isvalid(i+1,j,row,col,grid)){
            makezero(i+1,j,row,col,grid);
        }
        if (isvalid(i-1,j,row,col,grid)){
            makezero(i-1,j,row,col,grid);
        }
        if (isvalid(i,j+1,row,col,grid)){
            makezero(i,j+1,row,col,grid);
        }
        if (isvalid(i,j-1,row,col,grid)){
            makezero(i,j-1,row,col,grid);
        }
    }
    int numIslands(vector<vector<char>>& grid) {
        int row = grid.size() , col = grid[0].size();
        int islands = 0;
        for (int i=0 ; i<row ; i++){
            for (int j=0 ; j<col; j++){
                if (grid[i][j]=='1'){
                    islands++;
                    makezero(i,j,row,col,grid);
                }
            }
        }
        return islands;
    }
};
```

Time Complexity: O(m*n)

7. Quick Sort

```
import java.util.Arrays;

public class QuickSort {
    public static int partition(int[] arr, int high, int low){
        int ind = arr[high];
        int i = low-1;

        for (int j=low;j<=high-1;j++){
            if (arr[j] < ind){
                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }

        int temp = arr[i+1];
        arr[i+1] = arr[high];
        arr[high] = temp;
        return i+1;

    }
    public static void solution(int[] arr,int low,int high){
        if (low<high){
            int ind = partition(arr, high, low);

            solution(arr,low,ind-1);
            solution(arr,ind+1,high);
        }

    }
    public static void main(String[] args) {
        int[] arr = {10, 7, 8, 9, 1, 5};
        solution(arr,0,arr.length-1);
        System.out.println(Arrays.toString(arr));
    }

}
```

Time Complexity: O(n logn)

8. Merge Sort

```
import java.io.*;

class MergeSort {
    static void merge(int arr[], int l, int m, int r)
    {
        int n1 = m - l + 1;
        int n2 = r - m;
        int L[] = new int[n1];
        int R[] = new int[n2];
        for (int i = 0; i < n1; ++i)
            L[i] = arr[l + i];
        for (int j = 0; j < n2; ++j)
            R[j] = arr[m + 1 + j];
        int i = 0, j = 0;
        int k = l;
        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                arr[k] = L[i];
                i++;
            }
            else {
                arr[k] = R[j];
                j++;
            }
            k++;
        }
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }
        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }

    static void sort(int arr[], int l, int r)
    {
        if (l < r) {

            int m = l + (r - l) / 2;
            sort(arr, l, m);
            sort(arr, m + 1, r);
            merge(arr, l, m, r);
        }
    }
    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }
    public static void main(String args[])
    {
        int arr[] = { 12, 11, 13, 5, 6, 7 };

        System.out.println("Input array is");
        printArray(arr);

        sort(arr, 0, arr.length - 1);

        System.out.println("\nSorted array is");
        printArray(arr);
    }
}
```

Time Complexity: O (n log n)

9. Ternary Search

```
class TernarySearch{

  static int ternarySearch(int l, int r, int key, int ar[])
  {
      if (r >= l) {

          int mid1 = l + (r - l) / 3;
```

```java
          int mid2 = r - (r - l) / 3;

          if (ar[mid1] == key) {
              return mid1;
          }
          if (ar[mid2] == key) {
              return mid2;
          }


          if (key < ar[mid1]) {

              return ternarySearch(l, mid1 - 1, key, ar);
          }
          else if (key > ar[mid2]) {

              return ternarySearch(mid2 + 1, r, key, ar);
          }
          else {

              return ternarySearch(mid1 + 1, mid2 - 1, key, ar);
          }
      }

      return -1;
  }

  public static void main(String args[])
  {
      int l, r, p, key;


      int ar[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

      l = 0;

      r = 9;

      key = 5;

      p = ternarySearch(l, r, key, ar);

      System.out.println("Index of " + key + " is " + p);

      key = 50;

      p = ternarySearch(l, r, key, ar);

      System.out.println("Index of " + key + " is " + p);
  }
}
```
Time Complexity: O (2 log n)

10. Interpolation Search

```java
import java.util.*;

class interpolationSearch {

  public static int solution(int arr[], int lo,int hi, int x)
  {
    int pos;

    if (lo <= hi && x >= arr[lo] && x <= arr[hi]) {

      pos = lo+ (((hi - lo) / (arr[hi] - arr[lo]))* (x - arr[lo]));

      if (arr[pos] == x)
        return pos;

      if (arr[pos] < x)
        return solution(arr, pos + 1, hi,x);

      if (arr[pos] > x)
        return solution(arr, lo, pos - 1,x);
    }
    return -1;
  }

  public static void main(String[] args)
  {


    int arr[] = { 10, 12, 13, 16, 18, 19, 20, 21,22, 23, 24, 33, 35, 42, 47 };

    int n = arr.length;

    int x = 18;
    int index = solution(arr, 0, n - 1, x);

    if (index != -1)
      System.out.println("Found at:  "+ index);
    else
      System.out.println("Element not found.");
  }
}
```

Time Complexity: O(log2(log2 n))