INDEX

Name KUMUD RAJ GHIMIRE Class 4C

Roll No CS160 Subject ADA Lab School BMSCE

Sl .No.	Date	Title	Page No.	Teacher Sign/ Remarks
1	31/03/2025	LeetCode Programs • Minimum deletion to make string bal. • Design Circ. Queue • Removing stars • Rev. Polish Not. • Winner of Circ. Game • Stack using Queue.	1-9	10/10 4/3/25
2	11/03/2025	LeetCode Programs • Insertion sort • Sort list • Validate BST • Flatten BST to LL • Next right pointer • K^{th} smallest in BST		10/10 11/3/25
3.	18/03/2025	Merge Sort along with time complexity.	10	
4.	25/03/2025	Quick Sort along with time complexity	10	
5.	01/04/2025	Prim's Algorithm Krushkal's Algorithm	10	1/4/25
6.	08/04/2025	Floyd's Algorithm Warshall's Algorithm	10	8/4/25
7.	15/04/2025	0/1 Knapsack problem Dijkstra's Algorithm	10	15/4/25
8.	22/04/2025	• Topological Sort (DFS & source rem) • Johnson Traters Algorithm	10	22/4/25

S.N	Date	Title		Signature
9.	29/04/2025	• Fractional Knapsack • Additional Programs	10	
10.	06/04/2025	• Heap sort • N - queens	10	6/5/25

```
        else {
            stk.push(Integer.parseInt(token));
        }
    }
    return stk.pop();
}
```

1823. Find the winner of the Circular Game

```
class Solution {
    public int findTheWinner(int n, int k) {
        int winner = 0;
        for (int i = 1; i <= n; i++) {
            winner = (winner + k) % i;
        }
        return winner + 1;
    }
}
```

225. Stack using Queue

```
class MyStack {
    private Queue<Integer> queue1;
    private Queue<Integer> queue2;

    public MyStack() {
        queue1 = new LinkedList<>();
        queue2 = new LinkedList<>();
    }
```

```
public boolean empty(){
    return queues.isEmpty();
}
```

3

3

10

10

4/3/25

230. Kth Smallest Element in BST

```
class Solution {
    public int kthSmallest(TreeNode root, int k) {
        ArrayList<Integer> inorderList = new ArrayList<>();
        inorderTraversal(root, inorderList);
        return inorderList.get(k-1);
    }

    void inordesTraversal(TreeNode node, ArrayList<Integer> list) {
        if (node == null) {
            return;
        }

        inorderTraversal(node.left, list);
        list.add(node.val);
        inorderTraversal(node.right, list);
    }
}
```

Output:

root = [3,1,4,null,2]

k = 1

output

1

```
    for (i=0; i<n; i++){
        printf("%d", a[i]);
    }
    printf("\n Time taken to sort = %f \n", time_taken);
}

void merge (int a[], int low, int high)
{
    int mid = low + (high-low)/2;
    if (low < high){
        merge (a, low, mid);
        merge (a, mid+1, high);
        sort (a, low, mid, high);
    }
}

void sort (int a[], int low, int mid, int high)
{
    int i= low, j= mid+1, k= low;
    int temp[n];

    while (i<=mid && j<=high){
        if (a[i] < a[j])
            temp[k++] = a[i];
            i++;
        else {
            temp[k++] = a[j];
            j++; }
    }
```

```
while (i<=m){
        lst.add(a[i++]);
}

while(j<=n){
        lst.add(a[j++]);
}

for(i=0; i<lst.size(); i++){
    a[i+1] = lst.get(i);
}
}
}
```

Output:

nums = [2,0,2,1,1,0]

output : [0,0,1,1,2,2]

```
        else
        {
            k= a[j];
            a[j]= a[low];
            a[low]=k;
        }
    }
    return j;
}
```

Output:

```
Enter the no. of elements to sort: 10
The random generated array is:
41  67  134  100  169  124  78  158  162  64
The sorted array elements are:
41  64  67  78  100  124  134  158  162  169

Time taken = 0.000000
Do you want to continue?(0/1):0
```

```
        k++;
        count++;
    }
  }
}

int find (int parent[10], int i){
    while (parent[i]!=i){
        i = parent[i];
    }
    return i;
}
```

Output:

Enter the number of vertices: 5
Enter the cost adjacency matrix (999 for no edge):
0 1 2 999 999
1 0 999 1 100
2 999 0 3 999
999 1 3 0 4
999 100 999 4 0
Edges of the minimal spanning tree:
(0,1) (1,3) (0,2) (3,4)
sum of minimal spanning tree: 8

```
        k++;
        count++;
    }
  }
}

int find (int parent[10], int i ){
    while ( parent[i] != i ){
        i = parent[i];
    }
    return i;
}
```

Output:

```
Enter the number of vertices: 5
Enter the cost adjacency matrix (999 for no edge):
0 1 2 999 999
1 0 999 1 100
2 999 0 3 999
999 1 3 0 4
999 100 999 4 0
Edges of the minimal spanning tree:
(0,1) (1,3) (0,2) (3,4)
sum of minimal spanning tree: 8
```

```
        }
    }

    for(k=0; k<n; k++){
        for(i=0; i<n; i++){
            for(j=0; j<n; j++){
                D[i][j]= min(D[i][j], D[i][k]+D[k][j]);
            }
        }
    }

int min(int a, int b){
    if(a<b)
        return a;
    else
        return b;
}
```

TIME COMPLEXITY:

$$TC = \sum_{k=0}^{n-1} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} 1$$

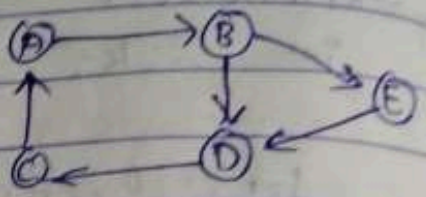
$$= \sum_{k=0}^{n-1} \sum_{i=0}^{n-1} n$$

$$= \sum_{k=0}^{n-1} n^2$$

$$= n^3$$

proved

Output:



Enter the numbes of nodes : 5
Enter the adjacency matrix:

0	1	0	0	0
0	0	0	1	1
1	0	0	0	0
0	0	1	0	0
0	0	0	1	0

	A	B	C	D	E
A	0	1	0	0	0
B	0	0	0	1	1
C	1	0	0	0	0
D	0	0	1	0	0
E	0	0	0	1	0

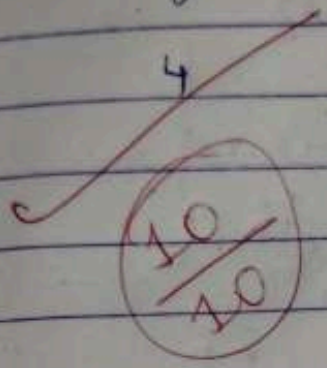
The path matrix is shown below:

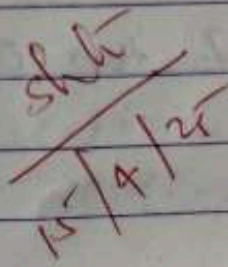
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

10/10

8/4/25

vertex	Distance
1	0
2	14
3	4
4	5





```
    scanf("%d", &arr...
    generatePermutations(arr, 0, n-1);
    free(arr);
    return 0;
}
```

Output:

```
Enter the numbes of elements: 3
Enter the elements: 1 2 3
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

Time Complexity:

- We generate all permutations of 'n' elements
- Total numbes of permutations = $n!$

∴ Time Complexity: $O(n \times n!)$

```
for (int i = start; i < n; i++) {
    int temp = path[start];
    path[start] = path[i];
    path[i] = temp;
    tsp(dist, path, start+1, n, minCost);
    temp = path[start];
    path[start] = path[i];
    path[i] = temp;
  }
}

int main() {
    int path[V] = {0, 1, 2, 3}, minCost = INT_MAX;
    tsp(dist, path, 0, V, &minCost);
    printf("Min Travel Distance: %d \n", minCost);
    return 0;
}
```

Output:

Min Travel Distance: 80

29/4/25

10/10

```
int place(int k, int j) {
    int i;
    for(i=1; i<k; i++)
        if(x[i]==j) || (abs(x[i]-j)) == abs(i-k))
            return 0;
    return 1;
}
```

Output:

Enter the number of queens: 4

Solution 1:

Row 1 ---> Column 2
Row 2 ---> Column 4
Row 3 ---> Column 1
Row 4 ---> Column 3

Solution 2:

Row 1 ---> Column 3
Row 2 ---> Column 1
Row 3 ---> Column 4
Row 4 ---> Column 2

Time Complexity: O(n!)

9/5/24
6/5/24