3/4/2025 – lab2

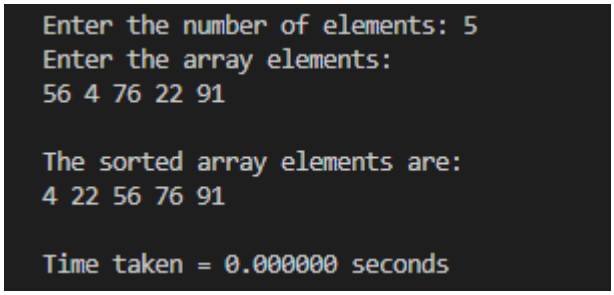
1. Sort a given set of N integer elements using quick sort technique and compute its time taken

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX 5000

void quicksort(int[], int, int);
int partition(int[], int, int);

int main() {
   int i, n, a[MAX], ch = 1;
   clock_t start, end;

   while (ch) {
      printf("\nEnter the number of elements: ");
      scanf("%d", &n);

      if (n > MAX) {
         printf("Maximum limit exceeded!\n");
         continue;
      }
      printf("Enter the array elements:\n");
      for (i = 0; i < n; i++)
         scanf("%d", &a[i]);

      start = clock();
      quicksort(a, 0, n - 1);
      end = clock();
```

```
        printf("\nThe sorted array elements are:\n");
        for (i = 0; i < n; i++)
            printf("%d ", a[i]);

        double time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;
        printf("\n\nTime taken = %f seconds\n", time_taken);

        printf("\nDo you wish to continue? (0/1): ");
        scanf("%d", &ch);
    }
    return 0;
}

void quicksort(int a[], int low, int high) {
    if (low < high) {
        int mid = partition(a, low, high);
        quicksort(a, low, mid - 1);
        quicksort(a, mid + 1, high);
    }
}

int partition(int a[], int low, int high) {
    int key = a[low], i = low + 1, j = high, temp;

    while (i <= j) {
        while (i <= high && a[i] <= key)
            i++;
        while (a[j] > key)
            j--;
        if (i < j) {
            temp = a[i];
```

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

OUTPUT:

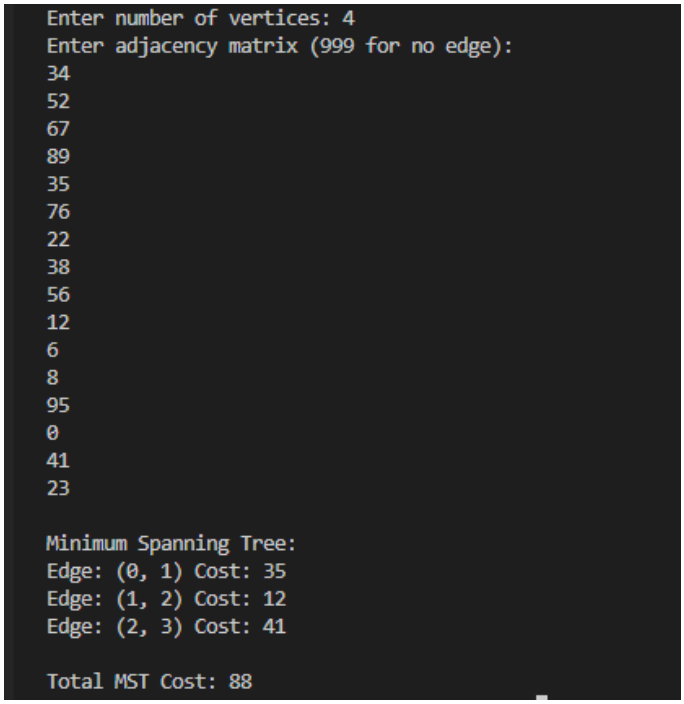


2. Find minimal cost spanning tree of a given undirected graph using Prim's Algorithm

```
#include <stdio.h>
#define INF 999

void prims(int cost[10][10], int n) {
  int i, j, u, min, sum = 0, visited[10] = {0}, d[10], p[10];

  for (i = 0; i < n; i++) {
     d[i] = cost[0][i];
     p[i] = 0;
  }
  visited[0] = 1;

  for (i = 1; i < n; i++) {
     min = INF;
     u = -1;

     for (j = 0; j < n; j++)
        if (!visited[j] && d[j] < min)
           min = d[j], u = j;

     if (u == -1) {
        printf("\nGraph is disconnected. MST not possible.\n");
        return;
     }

     visited[u] = 1;
     sum += cost[u][p[u]];
     printf("Edge: (%d, %d) Cost: %d\n", p[u], u, cost[u][p[u]]);

     for (j = 0; j < n; j++)
        if (!visited[j] && cost[u][j] < d[j])
           d[j] = cost[u][j], p[j] = u;
  }
  printf("\nTotal MST Cost: %d\n", sum);
}

int main() {
  int n, cost[10][10];
  printf("Enter number of vertices: ");
  scanf("%d", &n);
  printf("Enter adjacency matrix (%d for no edge):\n", INF);
  for (int i = 0; i < n; i++)
     for (int j = 0; j < n; j++) {
        scanf("%d", &cost[i][j]);
        if (i == j) cost[i][j] = INF;
```

```
    }
  printf("\nMinimum Spanning Tree:\n");
  prims(cost, n);
  return 0;
}
```

OUTPUT:



3. Find minimal cost spanning tree of a given undirected graph using Kruskal's Algorithm

```
#include <stdio.h>

#define INF 999
int parent[10];

int find(int i) {
  return (parent[i] == i) ? i : (parent[i] = find(parent[i]));
}

void kruskal(int cost[10][10], int n) {
  int u, v, min, sum = 0, edges = 0;

  for (int i = 0; i < n; i++) parent[i] = i;

  printf("\nMinimum Spanning Tree Edges:\n");

  while (edges < n - 1) {
```

```
    min = INF, u = -1, v = -1;

    for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
        if (find(i) != find(j) && cost[i][j] < min)
          min = cost[i][j], u = i, v = j;

    if (u == -1) break;

    parent[find(u)] = find(v);
    printf("(%d, %d) Cost: %d\n", u, v, min);
    sum += min;
    edges++;
  }

  printf("\nTotal MST Cost: %d\n", sum);
}

int main() {
  int n, cost[10][10];

  printf("Enter number of vertices: ");
  scanf("%d", &n);

  printf("Enter adjacency matrix (%d for no edge):\n", INF);
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++) {
      scanf("%d", &cost[i][j]);
      if (i == j) cost[i][j] = INF;
    }

  kruskal(cost, n);
  return 0;
}
```

OUTPUT:

```
Enter number of vertices: 4
Enter adjacency matrix (999 for no edge):
23 45 67 8
12 24 46 68
1 4 8 9
29 92 30 56

Minimum Spanning Tree Edges:
(2, 0) Cost: 1
(2, 1) Cost: 4
(0, 3) Cost: 8

Total MST Cost: 13
```