Aayush Shah
19BCE245
18 September 2021

Design and Analysis of Algorithms

Practical 5

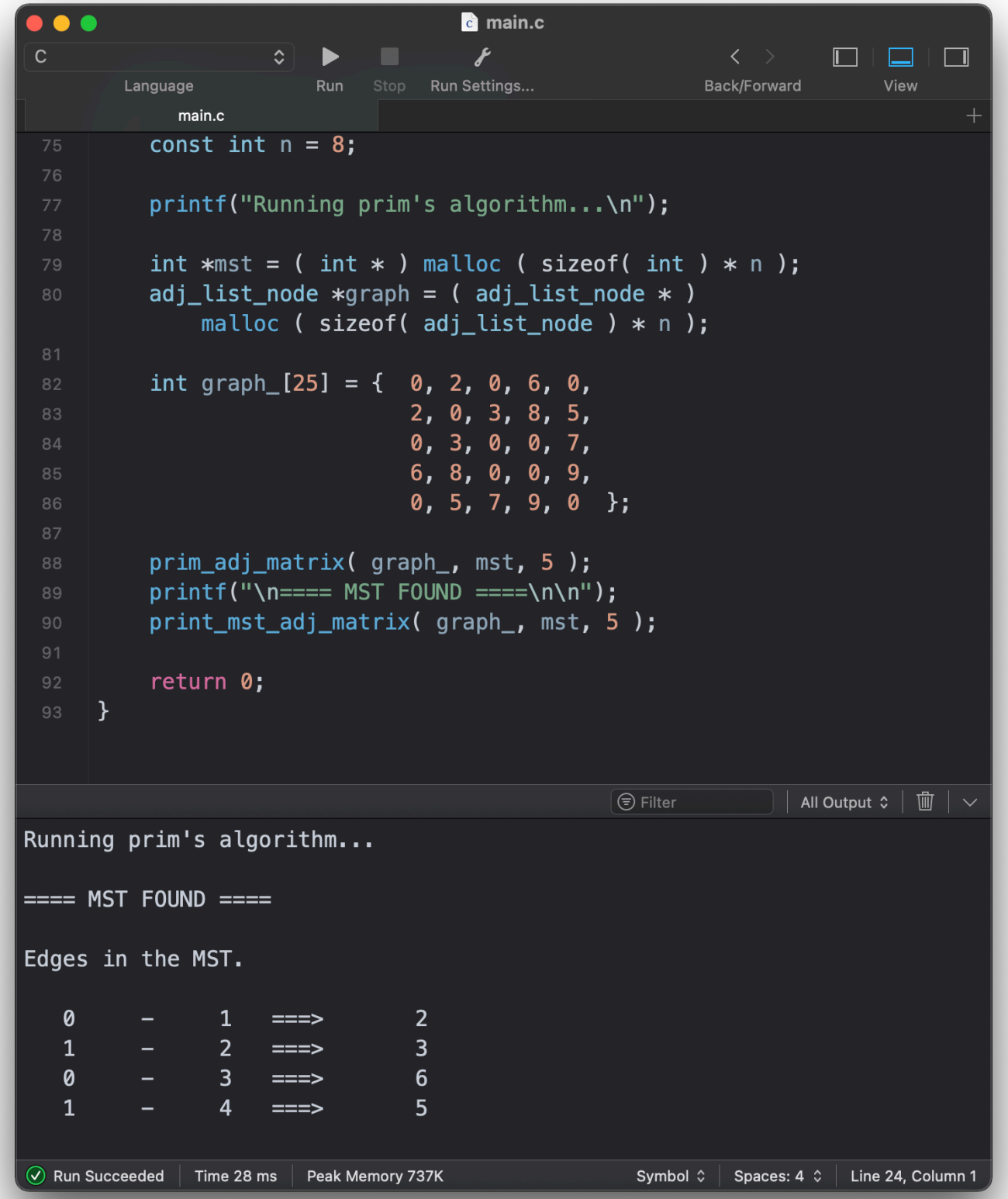
• **Code :**

```
/*
19BCE245 Aayush Shah
DAA practical 5
Prims Algorithm
*/

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <limits.h>


#define inf INT_MAX
#define MAX_SIZE 100
#define graph( i,j ) graph[ (i)*n + (j) ]


/* adjecency list representation for Graphs */
typedef struct {
    int weight[MAX_SIZE];
    int adj[MAX_SIZE];
    int adj_size;
} adj_list_node;

void print_mst_adj_matrix( int *graph, int *mst, const int n )
{
    printf("Edges in the MST.\n\n");
    for ( int i=1 ; i<n ; i++ ) {
```

```
        printf("   %-5d - %5d   ===>   %5d   \n", mst[i], i,
graph( i, mst[i] ) );
    }
}

void prim_adj_matrix( int *graph, int *mst, const int n )
{
    bool *visited = ( bool * ) calloc ( n , sizeof( bool ) );
    int  *key     = ( int  * ) malloc ( n * sizeof( int  ) );

    /* 'key' values of all the vertices will initially be
'inf' */
    for ( int i=0 ; i < n ; i++ ) key[i] = inf;

    key[0] = 0;  /* Start with vertex 0. */
    mst[0] = -1; /* No parent of the first vertex. */

    for ( int count=0 ; count < n-1 ; count++ ) {

        /* Find minimum key and include it in visited array.
*/
        int min_key = inf;
        int u = -1; /* Vertex corresponding to the minimum
key. */
        for ( int i=0 ; i < n ; i++ ) {
            if ( key[i] < min_key && visited[i] == false ) {
                min_key = key[i];
                u = i;
            }
        }

        visited[u] = true; /* Mark 'u' as visited. */

        /* Iterate throught all the edges of the node 'u'. */
        for ( int v=0 ; v < n ; v++ ) {

            if ( graph( u, v ) && visited[v] == false &&
graph( u, v ) < key[v] ) {
                mst[v] = u;
                key[v] = graph( u, v );
            }
        }
    }
```

```
    free( visited );
    free( key );
}


int main( int argc, char *argv[] )
{
    const int n = 8;

    printf("Running prim's algorithm...\n");

    int *mst = ( int * ) malloc ( sizeof( int ) * n );
    adj_list_node *graph = ( adj_list_node * ) malloc
( sizeof( adj_list_node ) * n );

    int graph_[25] = {  0, 2, 0, 6, 0,
                        2, 0, 3, 8, 5,
                        0, 3, 0, 0, 7,
                        6, 8, 0, 0, 9,
                        0, 5, 7, 9, 0  };

    prim_adj_matrix( graph_, mst, 5 );
    printf("\n==== MST FOUND ====\n\n");
    print_mst_adj_matrix( graph_, mst, 5 );

    return 0;
}
```

• **Output :**


main.c
C
Language
Run
Stop
Run Settings...
Back/Forward
View
main.c
75 const int n = 8;
76
77 printf("Running prim's algorithm...\n");
78
79 int *mst = (int *) malloc (sizeof(int) * n);
80 adj_list_node *graph = (adj_list_node *)
malloc (sizeof(adj_list_node) * n);
81
82 int graph_[25] = { 0, 2, 0, 6, 0,
83 2, 0, 3, 8, 5,
84 0, 3, 0, 0, 7,
85 6, 8, 0, 0, 9,
86 0, 5, 7, 9, 0 };
87
88 prim_adj_matrix(graph_, mst, 5);
89 printf("\n==== MST FOUND ====\n\n");
90 print_mst_adj_matrix(graph_, mst, 5);
91
92 return 0;
93 }
Filter
All Output
Running prim's algorithm...
==== MST FOUND ====
Edges in the MST.
0 - 1 ===> 2
1 - 2 ===> 3
0 - 3 ===> 6
1 - 4 ===> 5
Run Succeeded
Time 28 ms
Peak Memory 737K
Symbol
Spaces: 4
Line 24, Column 1