

CSE 2003

DATA STRUCTURES AND ALGORITHMS



Lab FAT

L19+L20 | SJT317
FALL SEMESTER 2020-21

by

SHARADINDU ADHIKARI
19BCE2105

Question 1

Problem:

Write a program to implement a queue by using an array and linked list.

Code, SS, & Output in CMD:

Queue using Array:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct queue {
    int head;
    int tail;
    int size;
    int Q[];
}queue;

queue* new_queue(int size) {
    queue *q = malloc(sizeof(queue) + size*sizeof(int));

    q->head = 1;
    q->tail = 1;
    q->size = size;

    return q;
}

int is_empty(queue *q) {
    if(q->tail == q->head)
        return 1;
    return 0;
}

int is_full(queue *q) {
    if(q->head == q->tail+1)
        return 1;
    return 0;
}

void enqueue(queue *q, int x) {
    if(is_full(q)) {
        printf("Queue Overflow\n");
    }
    else {
        q->Q[q->tail] = x;
        if(q->tail == q->size)
            q->tail = 1;
        else
            q->tail = q->tail+1;
    }
}
```

```

int dequeue(queue *q) {
    if(is_empty(q)) {
        printf("Underflow\n");
        return -1000;
    }
    else {
        int x = q->Q[q->head];
        if(q->head == q->size) {
            q->head = 1;
        }
        else {
            q->head = q->head+1;
        }
        return x;
    }
}

void display(queue *q) {
    int i;
    for(i=q->head; i<q->tail; i++) {
        printf("%d\n",q->Q[i]);
        if(i == q->size) {
            i = 0;
        }
    }
}

int main() {
    queue *q = new_queue(10);
    enqueue(q, 10);
    enqueue(q, 20);
    enqueue(q, 30);
    enqueue(q, 40);
    enqueue(q, 50);
    display(q);

    printf("\n");

    dequeue(q);
    dequeue(q);
    display(q);

    printf("\n");

    enqueue(q, 60);
    enqueue(q, 70);
    display(q);
    return 0;
}

```

```
#include <stdio.h>
#include <stdlib.h>

typedef struct queue {
    int head;
    int tail;
    int size;
    int Q[];
} queue;

queue* new_queue(int size) {
    queue *q = malloc(sizeof(queue) + size*sizeof(int));

    q->head = 1;
    q->tail = 1;
    q->size = size;

    return q;
}

int is_empty(queue *q) {
    if(q->tail == q->head)
        return 1;
    return 0;
}

int is_full(queue *q) {
    if(q->head == q->tail+1)
        return 1;
    return 0;
}
```

Process returned 0 (0x0) execution time : 0.050 s
Press any key to continue.

Queue using Linked List:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct node {
    int data;
    struct node *next;
} node;

typedef struct linked_list {
    struct node *head;
    struct node *tail;
} queue;
```

```

//to make new node
node* new_node(int data) {
    node *z;
    z = malloc(sizeof(struct node));
    z->data = data;
    z->next = NULL;

    return z;
}

//to make a new queue
queue* new_queue() {
    queue *q = malloc(sizeof(queue));
    q->head = NULL;
    q->tail = NULL;

    return q;
}

void traversal(queue *q) {
    node *temp = q->head; //temporary pointer to point to head

    while(temp != NULL) { //iterating over queue
        printf("%d\t", temp->data);
        temp = temp->next;
    }

    printf("\n");
}

int is_empty(queue *q) {
    if(q->head == NULL)
        return 1;
    return 0;
}

void enqueue(queue *q, node *n) {
    if(is_empty(q)) {
        q->head = n;
        q->tail = n;
    }
    else {
        q->tail->next = n;
        q->tail = n;
    }
}

int dequeue(queue *q) {
    if(is_empty(q)) {
        printf("Underflow\n");
        return -1000;
    }
    else {
        int x = q->head->data;
        node *temp = q->head;
        q->head = q->head->next;
        free(temp);
        return x;
    }
}

int main() {
    queue *q = new_queue();

```

```

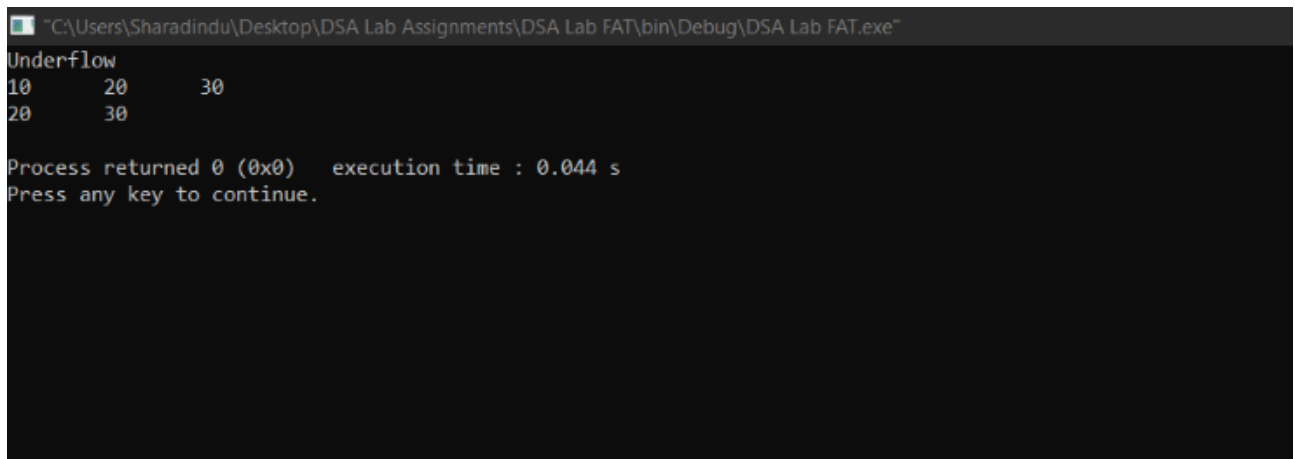
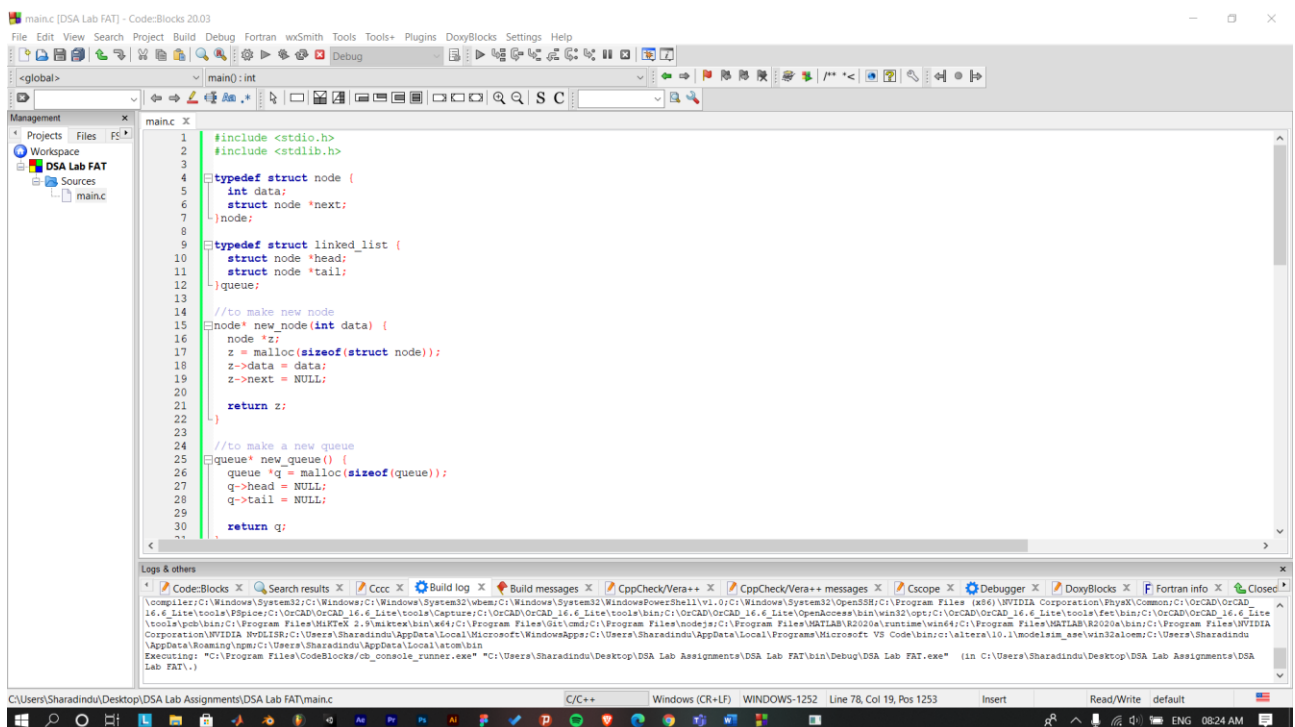
node *a, *b, *c;
a = new_node(10);
b = new_node(20);
c = new_node(30);

deque(q);
enqueue(q, a);
enqueue(q, b);
enqueue(q, c);

traversal(q);
deque(q);
traversal(q);

return 0;
}

```



Question 2

Problem:

Write a Program to implement of Prim's and Kruskal's algorithm.

Code, SS, & Output in CMD:

Prim's algorithm:

```
#include<stdio.h>
int a,b,u,v,n,i,j,pl=1;
int vis[10]={0},min,min_co=0,cost[10][10];
int main()
{
    printf("\nEnter the number of Nodes:");
    scanf("%d",&n);
    printf("\nEnter the Adjacency Matrix:\n");
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    vis[1]=1;
    printf("\n");
    while(pl < n)
    {
        for(i=1,min=999;i<=n;i++)
            for(j=1;j<=n;j++)
                if(cost[i][j]< min)
                    if(vis[i]!=0)
                    {
                        min=cost[i][j];
                        a=u=i;
                        b=v=j;
                    }
        if(vis[u]==0 || vis[v]==0)
        {
            printf("\n Edge %d:(%d %d) cost:%d",pl++,a,b,min);
            min_co+=min;
            vis[b]=1;
        }
        cost[a][b]=cost[b][a]=999;
    }
    printf("\n Minimun cost=%d",min_co);
    return 0;
}
```

```
main.c [DSA Lab FAT] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
<global>
Management
Projects Files FS
Workspace
DSA Lab FAT
Sources
main.c
main.c
1 #include<stdio.h>
2 int a,b,u,v,n,i,j,pl=1;
3 int vis[10]={0},min,min_co=0,cost[10][10];
4 int main()
5 {
6     printf("\nEnter the number of Nodes:");
7     scanf("%d",&n);
8     printf("\nEnter the Adjacency Matrix:\n");
9     for(i=1;i<=n;i++)
10        for(j=1;j<=n;j++)
11        {
12            scanf("%d",&cost[i][j]);
13            if(cost[i][j]==0)
14                cost[i][j]=999;
15        }
16        vis[i]=1;
17        printf("\n");
18        while(pl < n)
19        {
20            for(i=1,min=999;i<=n;i++)
21                for(j=1;j<=n;j++)
22                    if(cost[i][j]< min)
23                        if(vis[i]!=0)
24                        {
25                            min=cost[i][j];
26                            a=u=i;
27                            b=v=j;
28                        }
29            if(vis[u]==0 || vis[v]==0)
30            {
31                printf("Edge %d(%d %d) cost=%d\n",pl,u,b,min);
32            }
33        }
34    }
```

```
"C:\Users\Sharadindu\Desktop\DSA Lab Assignments\DSA Lab FAT\bin\Debug\DSA Lab FAT.exe"

Enter the number of Nodes:3

Enter the Adjacency Matrix:
1
2
3
4
5
6
7
8
9

Edge 1:(1 2) cost:2
Edge 2:(1 3) cost:3
Minimun cost=5
Process returned 0 (0x0)   execution time : 26.427 s
Press any key to continue.
```


Kruskal's algorithm:

```
#include<stdio.h>
#include<stdlib.h>
int I,j,k,x,y,u,v,n,pl=1;
int min,min_c=0,cost[9][9],pa[9];
int find(int i)
{
    while(pa[i])
        i=pa[i];
    return I;
}
int uni(int I,int j)
{
    if(i!=j)
    {
        pa[j]=I;
        return 1;
    }
    return 0;
}
int main()
{
    printf("\nEnter the no. of vertices:");
    scanf("%d",&n);
    printf("\nEnter the cost adjacency matrix:\n");
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            scanf("%d",&cost[i][j]);
            if(cost[i][j]==0)
                cost[i][j]=999;
        }
    }
    printf("The edges of Minimum Cost Spanning Tree are\n");
    while(pl < n)
    {
        for(i=1,min=999;i<=n;i++)
        {
            for(j=1;j <= n;j++)
            {
                if(cost[i][j] < min)
                {
                    min=cost[i][j];
                    x=u=I;
                    y=v=j;
                }
            }
        }
        u=find(u);
        v=find(v);
        if(uni(u,v))
        {
            printf("%d edge (%d,%d) =%d\n",pl++,x,y,min);
            min_c +=min;
        }
        cost[x][y]=cost[y][x]=999;
    }
    printf("\n\tMinimum cost = %d\n",min_c);
    return 0;
}
```

The screenshot shows the Code::Blocks IDE with the 'main.c' file open. The code implements a Minimum Cost Spanning Tree algorithm using Prim's algorithm. It includes headers for `<stdio.h>` and `<stdlib.h>`, and defines a graph with 10 vertices and 9 edges. The `main` function prompts the user to enter the number of vertices and the cost adjacency matrix. The `find` function finds the minimum edge that can be added to the tree without creating a cycle. The `uni` function checks if two vertices are already in the same component. The `main` function prints the edges of the Minimum Cost Spanning Tree and the minimum cost.

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int i,j,k,x,y,u,v,n,pl=1;
4 int min,min_c=0,cost[9][9],pa[9];
5 int find(int i)
6 {
7     while(pa[i])
8         i=pa[i];
9     return i;
10 }
11 int uni(int i,int j)
12 {
13     if(i!=j)
14     {
15         pa[j]=i;
16         return 1;
17     }
18     return 0;
19 }
20 int main()
21 {
22     printf("\nEnter the no. of vertices:");
23     scanf("%d",&n);
24     printf("\nEnter the cost adjacency matrix:\n");
25     for(i=1;i<=n;i++)
26     {
27         for(j=1;j<=n;j++)
28         {
29             scanf("%d",&cost[i][j]);
30             if(cost[i][j]==0)
31                 continue;
32         }
33     }
```

The 'Logs & others' window shows the execution of the program. It displays the input values and the output of the program, including the edges of the Minimum Cost Spanning Tree and the minimum cost.

```
Code::Blocks x Search results x Cccc x Build log x Build messages x CppCheck/Vera++ x CppCheck/Vera++ messages x Cscope x Debugger x DoxyBlocks x Fortran info x Closed
C:\Users\Sharadindu\Desktop\DSA Lab Assignments\DSA Lab FAT\main.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 61, Col 1, Pos 905 Insert Read/Write default ENG 08:27 AM
```

The screenshot shows the command prompt window titled "C:\Users\Sharadindu\Desktop\DSA Lab Assignments\DSA Lab FAT\bin\Debug\DSA Lab FAT.exe". The program prompts the user to enter the number of vertices and the cost adjacency matrix. The user enters 3 for the number of vertices and the following matrix for the cost adjacency matrix:

```
1 2 3
2 1 2 3
3 2 1 3
4 3 3 1
5 4 4 4 1
6 5 5 5 5 1
7 6 6 6 6 6 1
8 7 7 7 7 7 7 1
9 8 8 8 8 8 8 8 1
10 9 9 9 9 9 9 9 9 1
```

The program outputs the edges of the Minimum Cost Spanning Tree and the minimum cost.

```
The edges of Minimum Cost Spanning Tree are
1 edge (1,2) = 2
2 edge (1,3) = 3

Minimum cost = 5

Process returned 0 (0x0) execution time : 18.435 s
Press any key to continue.
```