Design and Analysis of Algorithms Lab Assessment-3

Name: Armaan Indani

Reg. No.: 22BCE3347

Slot : L41+42

Faculty: Dr. Joshva Devdas T – 16700

Question 1:

SCOPE / BCSE204P_VL2023240504902 / Assessment LAB 3

Assessment LAB 3

Back

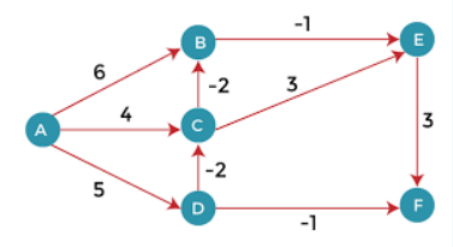
Question 1

Correct

Marked out of 20.00

Flag question

Write a program to compute the shingle source shortest path algorithm using Bellman Ford Algorithm



For example:

Test	Input	Result
1	A-B 6 A-C 4 A-D 5 C-B -2 D-C -2 B-E -1 C-E 3 D-F -1 E-F 3	A-0 B-1 C-3 D-5 E-0 F-3

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <limits.h>

int min(int a, int b)
{
    return a > b ? b : a;
}

int main()
{
    int x = INT_MAX;
    int n = 6;
    int c[6][6] = {
        {x, x, x, x, x, x},
        {x, x, x, x, x, x},
        {x, x, x, x, x, x},
        {x, x, x, x, x, x},
        {x, x, x, x, x, x},
        {x, x, x, x, x, x},
    };

    int d[6] = {0, x, x, x, x, x};

    for (int i = 0; i < 9; i++)
    {
        char s, d, dash;
        int wt;
        scanf("%c%c%c", &s, &dash, &d);
        scanf("%d", &wt);
        getchar();

        int src = (int)s - 65;
        int dest = (int)d - 65;

        if (src > 5 || dest > 5)
        {
            printf("Error src = %d, dest = %d invalid", src, dest);
        }
        else
        {
            c[src][dest] = wt;
        }
    }

    for (int k = 0; k < n; k++)
    {
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                if (i != j)
                {
                    if (c[i][j] != INT_MAX && d[i] != INT_MAX)
                    {
                        d[j] = min(d[j], d[i] + c[i][j]);
                    }
                }
            }
        }
    }

    printf("A-%d\n", d[0]);
    printf("B-%d\n", d[1]);
    printf("C-%d\n", d[2]);
    printf("D-%d\n", d[3]);
    printf("E-%d\n", d[4]);
    printf("F-%d\n", d[5]);

    return 0;
}
```

Check

Output:

	Test	Input	Expected	Got	
✓	1	A-B 6 A-C 4 A-D 5 C-B -2 D-C -2 B-E -1 C-E 3 D-F -1 E-F 3	A-0 B-1 C-3 D-5 E-0 F-3	A-0 B-1 C-3 D-5 E-0 F-3	✓

Passed all tests! ✓

Question 2:

SCOPE / BCSE204P_VL2023240504902 / Assessment LAB 3

Assessment LAB 3

Back

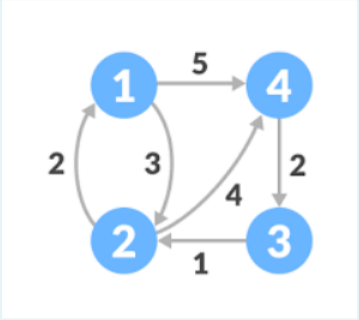
Question 2

Not complete

Marked out of 20.00

Flag question

Write a program to determine all pair shortest path algorithm using the following graph



Note : Assume infinity as 100

For example:

Test	Result
1	A0 = 0 3 100 5 2 0 100 4 100 1 0 100 100 100 2 0 A1= 0 3 100 5 2 0 100 4 100 1 0 8 100 100 2 0 A2= 0 3 100 5 2 0 100 4 3 1 0 5 100 100 2 0 A3= 0 3 100 5 2 0 100 4 3 1 0 5 5 3 2 0 A4= 0 3 7 5 2 0 6 4 3 1 0 5 5 3 2 0

Answer: (penalty regime: 0 %)

```
int min(int a, int b)
{
    return a > b ? b : a;
}

int main()
{
    int x = 100;
    int n = 4;
    int a[4][4] = {
        {0, 3, x, 5},
        {2, 0, x, 4},
        {x, 1, 0, x},
        {x, x, 2, 0},
    };

    printf("A0 =\n");
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
    printf("\n");

    for (int k = 0; k < n; k++)
    {
        printf("A%d=\n", k + 1);
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                if (a[i][k] != INT MAX && a[k][j] != INT MAX)
                {
                    a[i][j] = min(a[i][j], a[i][k] + a[k][j]);
                }
            }
        }
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                printf("%d ", a[i][j]);
            }
            printf("\n");
        }
    }

    return 0;
}
```

Check

Output:

	Test	Expected	Got	
✓	1	A0 = 0 3 100 5 2 0 100 4 100 1 0 100 100 100 2 0 A1= 0 3 100 5 2 0 100 4 100 1 0 100 100 100 2 0 A2= 0 3 100 5 2 0 100 4 3 1 0 5 100 100 2 0 A3= 0 3 100 5 2 0 100 4 3 1 0 5 5 3 2 0 A4= 0 3 7 5 2 0 6 4 3 1 0 5 5 3 2 0	A0 = 0 3 100 5 2 0 100 4 100 1 0 100 100 100 2 0 A1= 0 3 100 5 2 0 100 4 100 1 0 100 100 100 2 0 A2= 0 3 100 5 2 0 100 4 3 1 0 5 100 100 2 0 A3= 0 3 100 5 2 0 100 4 3 1 0 5 5 3 2 0 A4= 0 3 7 5 2 0 6 4 3 1 0 5 5 3 2 0	✓

Passed all tests! ✓

Question 3:

SCOPE / BCSE204P_VL2023240504902 / Assessment LAB 3

Assessment LAB 3

Back

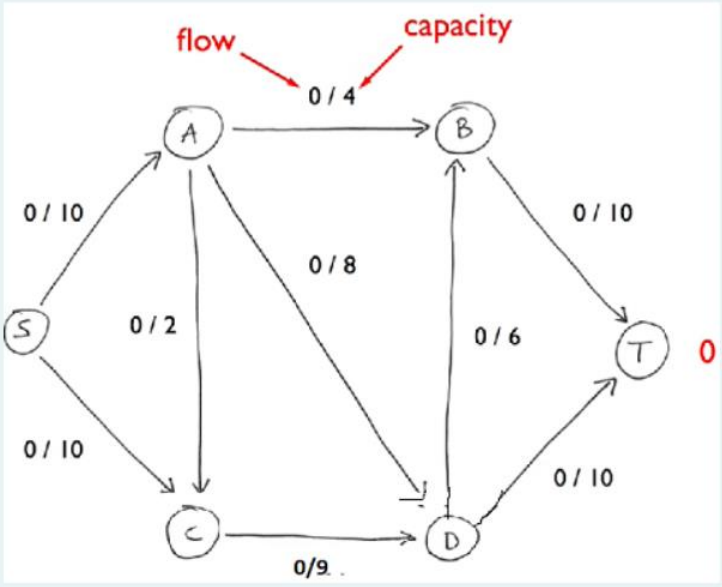
Question 3

Correct

Marked out of 20.00

Flag question

Write a Program to implement the max flow algorithm Ford Fulkerson by using the following graph



note : given the input based on the directed graph

For example:

Test	Result
1	The maximum flow of the algorithm is 19

Answer: (penalty regime: 0 %)

```c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <string.h>

int min(int a, int b)
{
    return a > b ? b : a;
}

struct edge
{
    int capacity;
    int flow;
    int from;
    int to;
};

struct queue
{
    int v[100];
    int front;
    int back;
};

void enqueue(struct queue *Q, int v_no)
{
    Q->front += 1;
    Q->v[Q->front] = v_no;
}

void dequeue(struct queue *Q)
{
    Q->back += 1;
}
```

```
36
37  struct edge *newE(int cap, int flow, int v1, int v2)
38  {
39      struct edge *e = (struct edge *)malloc(sizeof(struct edge));
40      e->capacity = cap;
41      e->flow = flow;
42      e->from = v1;
43      e->to = v2;
44      return e;
45  }
46
47  int main()
48  {
49      int n = 6;    // No of vertices
50      int n_e = 18; // No of edges
51
52      struct edge *E[18]; // For Residual edges as well
53      int ans = 0;
54
55      E[0] = newE(10, 0, 0, 1);  // Normal Edge
56      E[1] = newE(0, 0, 1, 0);   // Residual Edge
57      E[2] = newE(10, 0, 0, 3);  // Normal Edge
58      E[3] = newE(0, 0, 3, 0);   // Residual Edge
59      E[4] = newE(4, 0, 1, 2);   // Normal Edge
60      E[5] = newE(0, 0, 2, 1);   // Residual Edge
61      E[6] = newE(2, 0, 1, 3);   // Normal Edge
62      E[7] = newE(0, 0, 3, 1);   // Residual Edge
63      E[8] = newE(8, 0, 1, 4);   // Normal Edge
64      E[9] = newE(0, 0, 4, 1);   // Residual Edge
65      E[10] = newE(10, 0, 2, 5); // Normal Edge
66      E[11] = newE(0, 0, 5, 2);  // Residual Edge
67      E[12] = newE(9, 0, 3, 4);  // Normal Edge
68      E[13] = newE(0, 0, 4, 3);  // Residual Edge
69      E[14] = newE(6, 0, 4, 2);  // Normal Edge
70      E[15] = newE(0, 0, 2, 4);  // Residual Edge
71      E[16] = newE(10, 0, 4, 5); // Normal Edge
72      E[17] = newE(0, 0, 5, 4);  // Residual Edge
```

```
73
74      int pathAvailable = 1;
75      while (pathAvailable == 1)
76      {
77          int visited[6] = {0};
78          int parent[n];
79          for (int i = 0; i < n; i++)
80          {
81              parent[i] = -1;
82          }
83          struct queue *Q = (struct queue *)malloc(sizeof(struct queue));
84          Q->front = -1;
85          Q->back = 0;
86
87          enqueue(Q, 0);
88          visited[0] = 1;
89
90          while (visited[n - 1] != 1)
91          {
92              if (Q->front < Q->back)
93              {
94                  pathAvailable = 0;
95                  break;
96              }
97              for (int i = 0; i < n_e; i++)
98              {
99                  int currentV = Q->v[Q->back];
100                 if (E[i]->from == currentV)
101                 {
102                     if (visited[E[i]->to] == 1)
103                     {
104                         continue;
105                     }
106                     if (E[i]->capacity - E[i]->flow == 0)
107                     {
108                         continue;
109                     }
110
```

```
111                 // printf("Going from vertex %d to %d\n", E[i]->fr
112                 visited[E[i]->to] = 1;
113                 enqueue(Q, E[i]->to);
114                 parent[E[i]->to] = E[i]->from;
115             }
116         }
117         dequeue(Q);
118     }
119
120     // printf("[");
121     // for (int i = 0; i < n; i++)
122     // {
123     //     printf("%d ", visited[i]);
124     // }
125     // printf("]\n[");
126     // for (int i = 0; i < n; i++)
127     // {
128     //     printf("%d ", parent[i]);
129     // }
130     // printf("]\n");
131
132     int vert = n - 1;
133     int min_cap = INT_MAX;
134     // printf("  Path (reversed):\n");
135     while (vert != 0)
136     {
137         if (parent[vert] == -1)
138         {
139             pathAvailable = 0;
140             break;
141         }
142         int vert2 = parent[vert];
143         // printf("    %d-%d ", vert2, vert);
144         for (int i = 0; i < 2 * n_e; i++)
145         {
146             if ((E[i]->from == vert2) && (E[i]->to == vert))
147             {
```

```
                    min_cap = min(min_cap, E[i]->capacity - E[i]->flow
                    // printf("\n    {%d %d %d %d}\n", min_cap, E[i]->
                    break;
                }
            }
            vert = vert2;
        }

        if (!pathAvailable)
        {
            break;
        }

        // printf("\n  min_cap = %d\n\n", min_cap);
        ans += min_cap;

        vert = n - 1;
        while (vert != 0)
        {
            int vert2 = parent[vert];
            // printf("\n(vert2 = %d)\n", vert2);
            for (int i = 0; 2 * i < n_e - 1; i++)
            {
                // printf("[[2*i = %d]]", 2 * i);
                if ((E[2 * i]->from == vert2) && (E[2 * i]->to == vert
                {
                    // printf("{%d} ", 2 * i);
                    E[2 * i]->flow += min_cap;     // Forward Edge
                    E[2 * i + 1]->flow -= min_cap; // Residual Edge
                    // printf("\nChanging flow for edge %d,%d\n", E[2
                }
            }
            vert = vert2;
        }
    }

    printf("The maximum flow of the algorithm is %d", ans);

    return 0;
}
```

Check

	Test	Expected	Got
✓	1	The maximum flow of the algorithm is 19	The maximum flow of the algorit

Passed all tests! ✓

Question 4:

SCOPE / BCSE204P_VL2023240504902 / Assessment LAB 3

Assessment LAB 3

Back

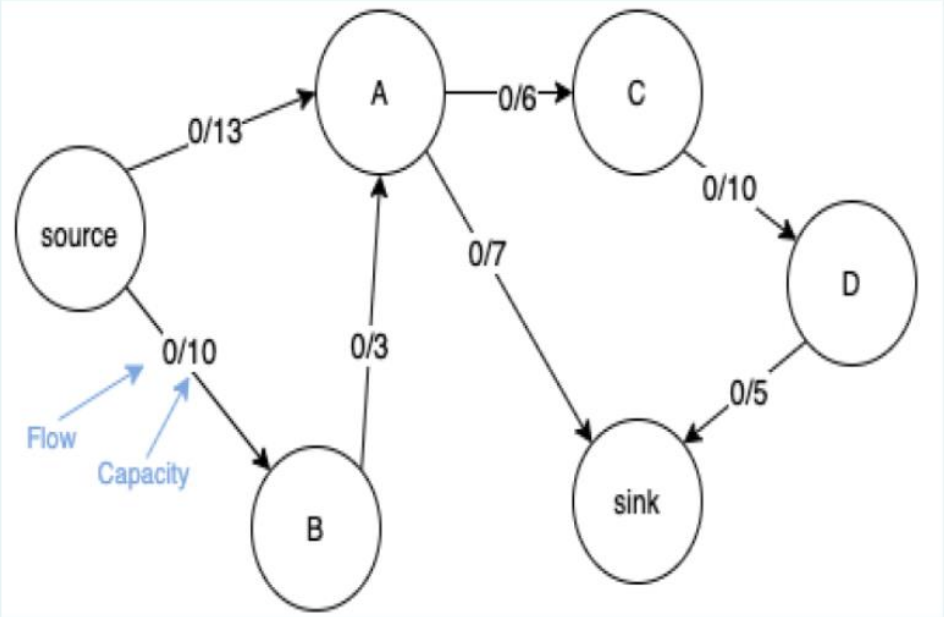
Question 4

Not complete

Marked out of 20.00

Flag question

Write a program to implement Push Relabel Algorithm to compute the max flow in the given network at the sink node T



Note : use the directed edges as input to determine the maxflow network

For example:

Test	Result
1	The max flow at the Sink node is 12

Answer: (penalty regime: 0 %)

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

int min(int a, int b)
{
    return a > b ? b : a;
}

struct edge
{
    int src;
    int dest;
    int flow;
    int cap;
};

struct edge *newEdge(int s, int d, int f, int c)
{
    struct edge *e = (struct edge *)malloc(sizeof(struct edge));
    e->src = s;
    e->dest = d;
    e->flow = f;
    e->cap = c;

    return e;
}

int main()
{
    int n_v = 6;
    int n_e = 7;

    struct edge *E[7];

    E[0] = newEdge(0, 1, 0, 13);
```

```
    E[1] = newEdge(0, 2, 0, 10);
    E[2] = newEdge(1, 3, 0, 6);
    E[3] = newEdge(1, 5, 0, 7);
    E[4] = newEdge(2, 1, 0, 3);
    E[5] = newEdge(3, 4, 0, 10);
    E[6] = newEdge(4, 5, 0, 5);

    int ht[7] = {0};
    int ov[7] = {0};

    ht[0] = 6;

    for (int i = 0; i < n_e; i++)
    {
        if (E[i]->src == 0)
        {
            E[i]->flow = E[i]->cap;
            ov[E[i]->dest] = E[i]->cap;
        }
    }
    int sum_of_overflows = 1;
    while (sum_of_overflows > 0)
    {
        for (int v = 1; v < n_v - 1; v++)
        {
            if (ov[v] > 0)
            {
                // printf("{%d, %d, %d}\n", v, ov[v], ht[v]);
                // Only forward edges
                for (int e = 0; e < n_e; e++)
                {
                    if (E[e]->src == v && E[e]->cap > E[e]->flow && ht[
                    {
                        int f1 = min(ov[v], E[e]->cap - E[e]->flow);
                        E[e]->flow += f1;
```

```
                    ov[v] -= f1;
                    ov[E[e]->dest] += f1;
                    // printf("Forward flow of %d to %d increased
                }
            }

            // Only backward edges
            for (int e = 0; e < n_e; e++)
            {
                if (E[e]->dest == v && E[e]->flow > 0 && ht[E[e]->
                {
                    int f1 = min(ov[v], E[e]->flow);
                    E[e]->flow -= f1;
                    ov[v] -= f1;
                    ov[E[e]->src] += f1;
                    // printf("Forward flow of %d to %d decreased
                }
            }
            if (ov[v] == 0)
            {
                // printf("    Vertex %d has overflow 0\n", v);
                continue;
            }
            else if (ov[v] > 0)
            {
                ht[v]++;
                v = 0;
                continue;
            }
        }
    }
    sum_of_overflows = 0;
    for (int i = 1; i < n_v - 1; i++)
    {
        sum_of_overflows += ov[i];
    }
}

printf("The max flow at the Sink node is %d", ov[n_v-1]);
return 0;
}
```

Check

	Test	Expected	Got
✓	1	The max flow at the Sink node is 12	The max flow at the Sink node is 12

Passed all tests! ✓