Algorithms
Final Assignment
Report

알고리즘 03분반

소프트웨어학부

20202475 이동훈

Contents

1. Top-down Cut Rod Algorithm

Program Code

```c
#include <stdio.h>

#define max(a, b) a > b ? a : b

int memoized_cut_rod(int p[11], int n) {
    int r[11] = { 0, };

    for (int i = 0; i <= n; i++) {
        r[i] = -1e9;
    }

    return memoized_cut_rod_aux(p, n, r);
}

int memoized_cut_rod_aux(int p[11], int n, int r[11]) {
    int q;
    int idx = -1;

    if (r[n] >= 0)
        return r[n];
    if (n == 0)
        q = 0;
    else {
        q = -1e9;
        for (int i = 1; i <= n; i++) {
            int tmp = q;
            q = max(q, p[i] + memoized_cut_rod_aux(p, n - i, r));
            if (tmp != q) {
                idx = i;
            }
        }
        if (idx != -1) printf("Length %2d : %d + %d, Value : %2d\n", n, idx, n - idx, q);
    }
    r[n] = q;

    return q;
}

int main() {
    int len = 10;
    int p[11] = { 0, 1, 4, 5, 7, 9, 11, 13, 13, 15, 16 };

    printf("Maximum amount : %d", memoized_cut_rod(p, len));

    return 0;
}
```

Execution Result

```
Length  1 : 1 + 0, Value :  1
Length  2 : 2 + 0, Value :  4
Length  3 : 1 + 2, Value :  5
Length  4 : 2 + 2, Value :  8
Length  5 : 1 + 4, Value :  9
Length  6 : 2 + 4, Value : 12
Length  7 : 1 + 6, Value : 13
Length  8 : 2 + 6, Value : 16
Length  9 : 1 + 8, Value : 17
Length 10 : 2 + 8, Value : 20
Maximum amount : 20
```

2. BFS

Program Code

```c
#include <stdio.h>

typedef enum color {
    WHITE, GRAY, BLACK
}color;

typedef struct vertex {
    color color;
    int d;
    int pre;
}vertex;

int main() {
    int linked_mat[8][8] = {
        {0, 1, 0, 0, 1, 0, 0, 0},
        {1, 0, 0, 0, 0, 1, 0, 0},
        {0, 0, 0, 1, 0, 1, 1, 0},
        {0, 0, 1, 0, 0, 0, 1, 1},
        {1, 0, 0, 0, 0, 0, 0, 0},
        {0, 1, 1, 0, 0, 0, 1, 0},
        {0, 0, 1, 1, 0, 1, 0, 1},
        {0, 0, 0, 1, 0, 0, 1, 0}
    };

    vertex node[8];

    for (int i = 0; i < 8; i++) {
        node[i].color = WHITE;
        node[i].d = 1e9;
        node[i].pre = -1;
    }
```

```
    int s = 1;

    int queue[20];
    int f, r;
    f = r = 0;

    node[s].color = GRAY;
    node[s].d = 0;

    queue[r++] = s;

    while (f < r) {
        int front = queue[f++];

        for (int i = 0; i < 8; i++) {
            if (linked_mat[front][i] == 1) {
                if (node[i].color == WHITE) {
                    node[i].color = GRAY;
                    node[i].d = node[front].d + 1;
                    node[i].pre = front;
                    queue[r++] = i;
                }
            }
        }

        node[front].color = BLACK;
    }
```

```
    printf("Num  Dis  Pre\n");

    for (int i = 0; i < 8; i++) {
        printf(" %d    %d    %d\n", i, node[i].d, node[i].pre);
    }

    return 0;
}
```

Execution Result

r : 0, s : 1, t : 2, u : 3

v : 4, w : 5, x : 6, y : 7

```
Num  Dis  Pre
 0    1    1
 1    0   -1
 2    2    5
 3    3    2
 4    2    0
 5    1    1
 6    2    5
 7    3    6
```

3. Dijkstra Algorithm

Program Code

```c
#include <stdio.h>

int vertex[5][5] = {
    {1e9,   3, 1e9,   5, 1e9},
    {1e9, 1e9,   6,   2, 1e9},
    {1e9, 1e9, 1e9, 1e9,   2},
    {1e9,   1,   4, 1e9,   6},
    {  3, 1e9,   7, 1e9, 1e9}
};

int dis[5];
int vis[5];

int choose() {
    int i, min, minpos;
    min = 1e9;
    minpos = -1;

    for (i = 0; i < 5; i++) {

        if (dis[i] < min && vis[i] == -1) {
            min = dis[i];
            minpos = i;
        }
    }
    return minpos;
}
```

```c
void shortest_path(int start) {
    for (int i = 0; i < 5; i++) {
        dis[i] = vertex[start][i];
        vis[i] = -1;
    }

    vis[start] = 0;
    dis[start] = 0;
    for (int i = 0; i < 4; i++) {
        int u = choose();

        vis[u] = i;
        for (int w = 0; w < 5; w++) {

            if (vis[w] == -1) {
                if (dis[u] + vertex[u][w] < dis[w]) {
                    dis[w] = dis[u] + vertex[u][w];
                }
            }
        }
    }
}
```

```c
int main() {
    shortest_path(0);

    printf("Node : y, Cost : %d\n3", dis[3]);

    int print = 3;
    while (print != 0) {
        print = vis[print];
        printf(" <- %d", print);
    }


    printf("\nNode : z, Cost : %d\n4", dis[4]);

    print = 4;
    while (print != 0) {
        print = vis[print];
        printf(" <- %d", print);
    }

    return 0;
}
```

Execution Result

s : 0, t : 1, x : 2, y : 3, z : 4

```
Node : y, Cost : 5
3 <- 1 <- 0
Node : z, Cost : 11
4 <- 3 <- 1 <- 0
```

4. Bellman-Ford Algorithm

Program Code

```c
#include <stdio.h>

typedef struct vertex {
    int d;
    int pre;
}vertex;

int main() {
    int flag = 1;
    vertex node[5];

    int edge[5][5] = {
        {1e9,   5, 1e9,   6, 1e9},
        {1e9, 1e9,   5,   8,  -4},
        {1e9,  -2, 1e9, 1e9, 1e9},
        {1e9, 1e9,  -3, 1e9,   9},
        {  2, 1e9,   4, 1e9, 1e9}
    };

    for (int i = 0; i < 5; i++) {
        node[i].d = 1e9;
        node[i].pre = -1;
    }

    node[0].d = 0;
```

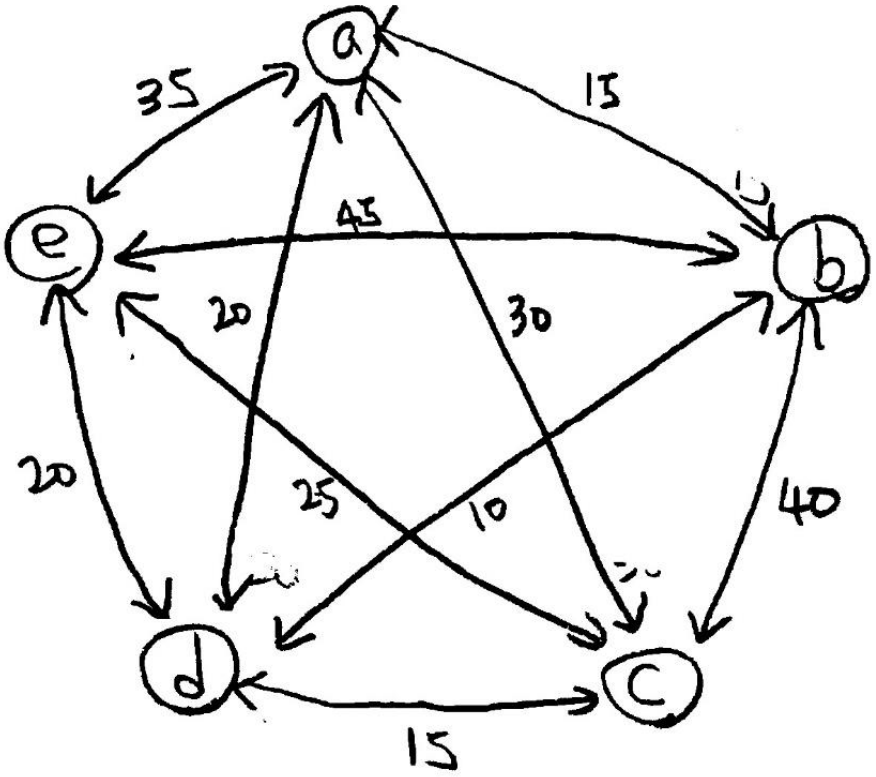
```
    node[0].d = 0;

    for (int v = 1; v < 5; v++) {
        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (edge[i][j] != 1e9) {
                    if (node[i].d != 1e9 && node[i].d + edge[i][j] < node[j].d) {
                        node[j].d = node[i].d + edge[i][j];
                        node[j].pre = i;
                    }
                }
            }
        }

        for (int i = 0; i < 5; i++) {
            if (node[i].d == 1e9 && node[i].pre == -1)
                printf("Node : %d, Distance : INF, Predecessor : NIL\n", i);
            else if (node[i].d == 1e9)
                printf("Node : %d, Distance : INF, Predecessor : %3d\n", i, node[i].pre);
            else if (node[i].pre == -1)
                printf("Node : %d, Distance : %3d, Predecessor : NIL\n", i, node[i].d);
            else
                printf("Node : %d, Distance : %3d, Predecessor : %3d\n", i, node[i].d, node[i].pre);
        }
        printf("\n\n");
    }

    for (int i = 0; i < 5 && flag; i++) {
        for (int j = 0; j < 5 && flag; j++) {
            if (edge[i][j] != 1e9) {
                if (node[i].d != 1e9 && node[i].d + edge[i][j] < node[j].d) {
                    flag = 0;
                }
            }
        }
    }

    if (flag) {
        printf("Negative-Weight Cycle exists");
    }
    else
        printf("Available to use Bellman-Ford Algorithm");

    return 0;
```

Execution Result

s : 0, t : 1, x : 2, y : 3, z : 4

```
Node : 0, Distance :   0, Predecessor : NIL
Node : 1, Distance :   5, Predecessor :   0
Node : 2, Distance :   3, Predecessor :   3
Node : 3, Distance :   6, Predecessor :   0
Node : 4, Distance :   1, Predecessor :   1


Node : 0, Distance :   0, Predecessor : NIL
Node : 1, Distance :   1, Predecessor :   2
Node : 2, Distance :   3, Predecessor :   3
Node : 3, Distance :   6, Predecessor :   0
Node : 4, Distance :   1, Predecessor :   1


Node : 0, Distance :  -1, Predecessor :   4
Node : 1, Distance :   1, Predecessor :   2
Node : 2, Distance :   1, Predecessor :   4
Node : 3, Distance :   6, Predecessor :   0
Node : 4, Distance :  -3, Predecessor :   1


Node : 0, Distance :  -1, Predecessor :   4
Node : 1, Distance :  -1, Predecessor :   2
Node : 2, Distance :   1, Predecessor :   4
Node : 3, Distance :   5, Predecessor :   0
Node : 4, Distance :  -3, Predecessor :   1


Negative-Weight Cycle exists
```

5. TSP



$C(b,phi) = 15$, $C(c,phi) = 30$, $C(d,phi) = 20$, $C(e,phi) = 35$

$C(b,\{c\}) = d(b,c) + C(c,phi) = 70$, $C(b,\{d\}) = d(b,d) + C(d,phi) = 30$

$C(b,\{e\}) = d(b,e) + C(e,phi) = 80$, $C(c,\{b\}) = d(c,b) + C(b,phi) = 55$

$C(c,\{d\}) = d(c,d) + C(d,phi) = 35$, $C(c,\{e\}) = d(c,e) + C(e,phi) = 60$

$C(d,\{b\}) = d(d,b) + C(b,phi) = 25$, $C(d,\{c\}) = d(d,c) + C(c,phi) = 45$

$C(d,\{e\}) = d(d,e) + C(e,phi) = 55$, $C(e,\{b\}) = d(e,b) + C(b,phi) = 60$

$C(e,\{c\}) = d(e,c) + C(c,phi) = 55$, $C(e,\{d\}) = d(e,d) + C(d,phi) = 40$

$C(b,\{c,d\}) = \min(d(b,c) + C(c,\{d\}),\ d(b,d) + C(d,\{b\})) = \min(95, 35) = 35$

$C(b,\{c,e\}) = \min(d(b,c) + C(c,\{e\}),\ d(b,e) + C(e,\{c\})) = \min(100, 100) = 100$

$C(b,\{d,e\}) = \min(d(b,d) + C(d,\{e\}),\ d(b,e) + C(e,\{d\})) = \min(65, 85) = 65$

$C(c,\{b,d\}) = \min(d(c,b) + C(b,\{d\}),\ d(c,d) + C(d,\{b\})) = \min(90, 40) = 40$

$C(c,\{b,e\}) = \min(d(c,b) + C(b,\{e\}),\ d(c,e) + C(e,\{b\})) = \min(120, 85) = 85$

$C(c,\{d,e\}) = \min(d(c,d) + C(d,\{e\}),\ d(c,e) + C(e,\{d\})) = \min(90, 65) = 65$

$C(d,\{b,c\}) = \min(d(d,b) + C(b,\{c\}),\ d(d,c) + C(c,\{b\})) = \min(80, 90) = 90$

$C(d,\{b,e\}) = \min(d(d,b) + C(b,\{e\}),\ d(d,e) + C(e,\{b\})) = \min(90, 80) = 80$

$C(d,\{c,e\}) = \min(d(d,c) + C(c,\{e\}),\ d(d,e) + C(e,\{c\})) = \min(95, 95) = 95$

$C(e,\{b,c\}) = \min(d(e,b) + C(b,\{c\}),\ d(e,c) + C(c,\{b\})) = \min(115, 80) = 80$

$C(e,\{b,d\}) = \min(d(e,b) + C(b,\{d\}),\ d(e,d) + C(d,\{b\})) = \min(95, 45) = 45$

$C(e,\{c,d\}) = \min(d(e,c) + C(c,\{d\}),\ d(e,d) + C(d,\{c\})) = \min(80, 65) = 80$

$C(b,\{c,d,e\}) = \min(d(b,c) + C(c,\{d,e\}),\ d(b,d) + C(d,\{c,e\}),\ d(b,e) + C(e,\{c,d\})) = \min(105, 85, 105) = 85$

$C(c,\{b,d,e\}) = \min(d(c,b) + C(b,\{d,e\}),\ d(c,d) + C(d,\{b,e\}),\ d(c,e) + C(e,\{b,d\})) = \min(105, 95, 70) = 70$

$C(d,\{b,c,e\}) = \min(d(d,b) + C(b,\{c,e\}),\ d(d,c) + C(c,\{b,e\}),\ d(d,e) + C(e,\{b,c\})) = \min(75, 100, 100) = 75$

$C(e,\{b,c,d\}) = \min(d(e,b) + C(b,\{c,d\}),\ d(e,c) + C(c,\{b,d\}),\ d(e,d) + C(d,\{b,c\})) = \min(80, 65, 90) = 65$

$C(a,\{b,c,d,e\}) = \min(d(a,b) + C(b,\{c,d,e\}),\ d(a,c) + C(c,\{b,d,e\}),\ d(a,d) + C(d,\{b,c,e\}),\ d(a,e) + C(e,\{b,c,d\})) = \min(100, 100, 95, 100) = 95$

Shortest Route : A – D – B – C – E - A