# Algorithms Final Assignment

응용통계학과 20202850 김지민

**1.**

**- code**

```c
#include <stdio.h>
#include <stdlib.h>
#define SIZE 10
#define INF -99999

// Global Array for the purpose of memoization.
int r[SIZE+1];
int s[SIZE+1];

int memorized_cut_rod(int p[], int n) {
    for (int i = 0; i <= n; i++) {
        r[i] = INF;
    }
    return memorized_cut_rod_aux(p, n, r);
}

int memorized_cut_rod_aux(int p[], int n, int r[]) {
    int q;
    if (r[n] >= 0)
        return r[n];
    if (n == 0)
        q = 0;
    else {
        q = INF;
        for (int i = 1; i <= n; i++) {
            int temp = p[i] + memorized_cut_rod_aux(p, n - i, r);
            if (temp >= q) {
                q = temp;
                s[n] = i;
            }
        }
    }
    r[n] = q;
    return q;
}


int main(void) {
    int price[] = { 0, 1, 4, 5, 7, 9, 11, 13, 13, 15, 16 };
    int n = sizeof(price) / sizeof(price[0]) - 1;

    for (int i = 1; i <= n; i++) {
        printf("length %d : Maximum obtained value is %d, ", i, memorized_cut_rod(price, i, r));
        int num = i;
        if (s[i] == num) {
            printf("optimal cut is ( %d )₩n", s[i]);
        }
        else {
            printf("( ");
            while (num != 0) {
                printf("%d ", s[num]);
                num -= s[num];
            }
            printf(")₩n");
        }
    }
    return 0;
}
```

- **result**

Microsoft Visual Studio 디버그 콘솔

```
length 1 : Maximum obtained value is 1, optimal cut is ( 1 )
length 2 : Maximum obtained value is 4, optimal cut is ( 2 )
length 3 : Maximum obtained value is 5, optimal cut is ( 3 )
length 4 : Maximum obtained value is 8, ( 2 2 )
length 5 : Maximum obtained value is 9, optimal cut is ( 5 )
length 6 : Maximum obtained value is 12, ( 2 2 2 )
length 7 : Maximum obtained value is 13, optimal cut is ( 7 )
length 8 : Maximum obtained value is 16, ( 2 2 2 2 )
length 9 : Maximum obtained value is 17, ( 7 2 )
length 10 : Maximum obtained value is 20, ( 2 2 2 2 2 )

C:₩Users₩jimin.DESKTOP-8V20QSQ₩source₩repos₩Project1₩Debug₩Project1.exe
).
```

**2.**

- **code**

```c
#include <stdio.h>
#include <stdlib.h>
#define INF 99999
#define SIZE 8
#define WHITE 0
#define GRAY 1
#define BLACK 2

// Queue
typedef struct Queue {
    int items[SIZE];
    int front;
    int rear;
} Queue;

int isEmpty(Queue* q) {
    if (q->rear == -1)
        return 1;
    else
        return 0;
}

void enqueue(Queue* q, int value) {
    if (q->rear == SIZE - 1)
        printf("₩nQueue is Full.");
    else {
        if (q->front == -1)
            q->front = 0;
        q->rear++;
        q->items[q->rear] = value;
    }
}
```

```c
int dequeue(Queue* q) {
    int item;
    if (isEmpty(q)) {
        printf("Queue is empty.");
        item = -1;
    }
    else {
        item = q->items[q->front];
        q->front++;
        if (q->front > q->rear) {
            q->front = q->rear = -1;
        }
    }
    return item;
}

int convert_int(char s) {
    switch (s) {
    case 'r': return 0;
    case 's': return 1;
    case 't': return 2;
    case 'u': return 3;
    case 'v': return 4;
    case 'w': return 5;
    case 'x': return 6;
    case 'y': return 7;
    default: return;
    }
}
char convert_char(int n) {
    switch (n) {
    case 0: return 'r';
    case 1: return 's';
    case 2: return 't';
    case 3: return 'u';
    case 4: return 'v';
    case 5: return 'w';
    case 6: return 'x';
    case 7: return 'y';
    default: return;
    }
}

typedef struct Node {
    char vertex;
    int d;
    int color;
    char parent;
    struct Node* link;
} Node;

Node* create_Node(char v) {
    Node* newNode = malloc(sizeof(Node));
    newNode->vertex = v;
    newNode->color = WHITE;
    newNode->d = INF;
    newNode->parent = NULL;
    return newNode;
};
```

```c
typedef struct Graph {
    int n;
    Node* node[SIZE];
    Node* adjLists[SIZE];
} Graph;

// BFS algorithm
void bfs(Graph* g, char start) {

    Queue* q = malloc(sizeof(Queue));
    q->front = -1;
    q->rear = -1;
    int s = convert_int(start);
    g->node[s]->color = GRAY;
    g->node[s]->d = 0;
    g->node[s]->parent = NULL;
    enqueue(q, s);
    printf("Visited %c | distance : %d ₩n", start, g->node[s]->d, g->node[s]->parent);
    while (!isEmpty(q)) {
        int cur = dequeue(q);

        Node* temp = g->adjLists[cur];

        while (temp != NULL) {
            char adj = temp->vertex;
            int a = convert_int(adj);
            if (g->node[a]->color == WHITE) {
                g->node[a]->color = GRAY;
                g->node[a]->d = g->node[cur]->d + 1;
                g->node[a]->parent = convert_char(cur);
                printf("Visited %c | distance : %d, parent : %c₩n", adj, g->node[a]->d, g->node[a]->parent);
                enqueue(q, a);
            }
            temp = temp->link;
        }
        g->node[cur]->color = BLACK;
    }
}

void addEdge(Graph* g, char start, char end) {
    // Add edge from start to end
    Node* newNode = create_Node(end);
    newNode->link = g->adjLists[convert_int(start)];
    g->adjLists[convert_int(start)] = newNode;

    // Add edge from end to start
    newNode = create_Node(start);
    newNode->link = g->adjLists[convert_int(end)];
    g->adjLists[convert_int(end)] = newNode;
}
```

```c
int main(void) {
    Graph* g = malloc(sizeof(Graph));
    g->n = SIZE;

    for (int i = 0; i < SIZE; i++) {
        g->adjLists[i] = NULL;
        g->node[i] = create_Node(convert_char(i));
    }

    addEdge(g, 'r', 's');
    addEdge(g, 'r', 'v');
    addEdge(g, 's', 'w');
    addEdge(g, 't', 'u');
    addEdge(g, 't', 'w');
    addEdge(g, 't', 'x');
    addEdge(g, 'u', 'x');
    addEdge(g, 'u', 'y');
    addEdge(g, 'w', 'x');
    addEdge(g, 'x', 'y');

    bfs(g, 's');

    return 0;
}
```

**- result**

Microsoft Visual Studio 디버그 콘솔

```
Visited s | distance : 0
Visited w | distance : 1, parent : s
Visited r | distance : 1, parent : s
Visited x | distance : 2, parent : w
Visited t | distance : 2, parent : w
Visited v | distance : 2, parent : r
Visited y | distance : 3, parent : x
Visited u | distance : 3, parent : x

C:₩Users₩jimin.DESKTOP-8V20QSQ₩source₩repos₩Project1₩Debug₩Project1.exe
)
```

**3.**

**- code**

```c
#include <stdio.h>
#define TRUE 1
#define FALSE 0
#define INF 99999
#define SIZE 5

int distance[SIZE];
char previous[SIZE];
int found[SIZE];

typedef struct Graph {
    int n;
    int adj_mat[SIZE][SIZE];
} Graph;

int convert_int(char s) {
    switch (s) {
    case 's': return 0;
    case 't': return 1;
    case 'x': return 2;
    case 'y': return 3;
    case 'z': return 4;
    default: return;
    }
}

char convert_char( int n) {
    switch (n) {
    case 0: return 's';
    case 1: return 't';
    case 2: return 'x';
    case 3: return 'y';
    case 4: return 'z';
    default: return;
    }
}
```

```c
void print_path(char start, char end) {
    char u = end;
    if (start == end) {
        printf("%c", start);
        return;
    }
    else {
        print_path(start, previous[convert_int(u)]);
        printf("->%c", u);
    }
}

int choose(int distance[], int n, int found[]) {
    int i, min, minpos;
    min = INF;
    minpos = -1;
    for (i = 0; i < n; i++)
        if (distance[i] < min && !found[i]) {
            min = distance[i];
            minpos = i;
        }
    return minpos;
}

void shortest_path(Graph* g, char start, char end) {
    int i, u, w;
    int s = convert_int(start);
    int e = convert_int(end);

    for (i = 0; i < g->n; i++) {
        distance[i] = g->adj_mat[s][i];
        found[i] = 0;
        previous[i] = start;
    }

    found[s] = 1;
    distance[s] = 0;

    for (i = 0; i < (g->n) - 1; i++) {
        u = choose(distance, g->n, found);
        found[u] = 1;
        if (u == e) {
            printf("shortest path : ");
            print_path(start, convert_char(u));
            printf("₩ndistance : %d₩n₩n", distance[u]);
        }
        for (w = 0; w < g->n; w++) {
            if (!found[w])
                if (distance[u] + g->adj_mat[u][w] < distance[w]) {
                    distance[w] = distance[u] + g->adj_mat[u][w];
                    previous[w] = convert_char(u);
                }
        }
    }
}
```

```c
int main(void) {
    Graph g = { 5,
    {{0, 3, INF, 5, INF},
    {INF, 0, 6, 5, INF},
    {INF, INF, 0, INF, 2},
    {INF, 1, 4, 0, 6},
    {3, INF, 7, INF, 0}}
    };

    printf("start vertex s, end vertex y₩n");
    shortest_path(&g, 's', 'y');
    printf("start vertex s, end vertex z₩n");
    shortest_path(&g, 's', 'z');

    return 0;
}
```

**- result**

Microsoft Visual Studio 디버그 콘솔

```
start vertex s, end vertex y
shortest path : s->y
distance : 5

start vertex s, end vertex z
shortest path : s->y->z
distance : 11


C:₩Users₩jimin.DESKTOP-8V20QSQ₩source₩repos₩Project1₩Debug₩Project1.exe(
).
```

**4.**

**- code**

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

#define INF 99999
#define FALSE 0
#define TRUE 1

typedef struct Edge {
    char start;
    char end;
    int w;  //weight of the edge
} Edge;

typedef struct Graph {
    int V;          // number of vertices
    int E;          // number of edges
    struct Edge edge[10];
} Graph;

int dis[5] = { INF, INF, INF, INF, INF };
char parent[5] = { 0, 0, 0, 0, 0 };

int convert_int(char s) {
    switch (s) {
    case 's': return 0;
    case 't': return 1;
    case 'x': return 2;
    case 'y': return 3;
    case 'z': return 4;
    }
}

void print_node(Graph* g) {
    printf("Distance    : ");
    for (int k = 0; k < g->V; k++) {
        if (dis[k] == INF)
            printf("INF ");
        else
            printf("%d ", dis[k]);
    }
    printf("₩n");
    printf("Predecessor : ");
    for (int k = 0; k < g->V; k++) {
        if (parent[k] == 0)
            printf("NIL ");
        else
            printf("%c ", parent[k]);
    }
    printf("₩n₩n");
}
```

```c
int bellmanford(Graph* g, char source) {

    int s, e, w;
    char arr[4] = { 's', 't', 'y', 'x'};
    dis[convert_int(source)] = 0;
    printf("              s t x y z\n");
    printf("=======================\n");
    int a = 0;
    char post, pre = 's';
    for (int i = 1; i <= g->V - 1; i++) {
        for (int j = 0; j < g->E; j++) {
            s = convert_int(g->edge[j].start);
            e = convert_int(g->edge[j].end);
            w = g->edge[j].w;

            if (dis[s] != INF && dis[e] > dis[s] + w) {
                post = g->edge[j].start;
                if (a < 5 && pre == arr[a] && pre != post) {
                    print_node(g);
                    a += 1;
                }
                pre = g->edge[j].start;
                dis[e] = dis[s] + w;
                parent[e] = g->edge[j].start;
            }
        }
    }
    print_node(g);

    for (int i = 0; i < g->E; i++) {
        s = convert_int(g->edge[i].start);
        e = convert_int(g->edge[i].end);
        w = g->edge[i].w;
        if (dis[s] != INF && dis[e] > dis[s] + w) {
            return FALSE;
        }
    }

    return TRUE;
}
```

```c
int main(void) {

    Graph* g = malloc(sizeof(Graph));
    g->V = 5;
    g->E = 10;

    g->edge[0].start = 's';
    g->edge[0].end = 't';
    g->edge[0].w = 5;

    g->edge[1].start = 's';
    g->edge[1].end = 'y';
    g->edge[1].w = 6;

    g->edge[2].start = 't';
    g->edge[2].end = 'x';
    g->edge[2].w = 5;

    g->edge[3].start = 't';
    g->edge[3].end = 'y';
    g->edge[3].w = 8;

    g->edge[4].start = 't';
    g->edge[4].end = 'z';
    g->edge[4].w = -4;

    g->edge[5].start = 'x';
    g->edge[5].end = 't';
    g->edge[5].w = -2;

    g->edge[6].start = 'y';
    g->edge[6].end = 'x';
    g->edge[6].w = -3;

    g->edge[7].start = 'y';
    g->edge[7].end = 'z';
    g->edge[7].w = 9;

    g->edge[8].start = 'z';
    g->edge[8].end = 's';
    g->edge[8].w = 2;

    g->edge[9].start = 'z';
    g->edge[9].end = 'x';
    g->edge[9].w = 4;

    if (bellmanford(g, 's') == TRUE)
        printf("algorithm returns TRUE.");
    else
        printf("algorithm returns FALSE.₩n");

    return 0;
}
```

**- result**

Microsoft Visual Studio 디버그 콘솔

```
              s t x y z
=========================
Distance    : 0 5 INF 6 INF
Predecessor : NIL s NIL s NIL

Distance    : 0 5 10 6 1
Predecessor : NIL s t s t

Distance    : 0 5 3 6 1
Predecessor : NIL s y s t

Distance    : 0 1 3 6 1
Predecessor : NIL x y s t

Distance    : -1 -1 1 5 -3
Predecessor : z x z s t

algorithm returns FALSE.

C:₩Users₩jimin.DESKTOP-8V20QSQ₩source₩repos₩Project1₩Debug₩Project1.exe
```

5.

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 0 | 15 | 30 | 20 | 35 |
| b | 15 | 0 | 40 | 10 | 45 |
| c | 30 | 40 | 0 | 15 | 25 |
| d | 20 | 10 | 15 | 0 | 20 |
| e | 35 | 45 | 25 | 20 | 0 |

Distance from start point

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

Minimum distance of two elements

$C(b, \{c\}) = d(b, c) + C(c, phi) = 40 + 30 = 70$

$C(b, \{d\}) = d(b, d) + C(d, phi) = 10 + 20 = 30$

$C(b, \{e\}) = d(b, e) + C(e, phi) = 45 + 35 = 80$

$C(c, \{b\}) = d(c, b) + C(b, phi) = 40 + 15 = 55$

$C(c, \{d\}) = d(c, d) + C(d, phi) = 15 + 20 = 35$

$C(c, \{e\}) = d(c, e) + C(e, phi) = 25 + 35 = 60$

$C(d, \{b\}) = d(d, b) + C(b, phi) = 10 + 15 = 25$

$C(d, \{c\}) = d(d, c) + C(c, phi) = 15 + 30 = 45$

$C(d, \{e\}) = d(d, e) + C(e, phi) = 20 + 35 = 55$

$C(e, \{b\}) = d(e, b) + C(b, phi) = 45 + 15 = 60$

$C(e, \{c\}) = d(e, c) + C(c, phi) = 25 + 30 = 55$

$C(e, \{d\}) = d(e, d) + C(d, phi) = 20 + 20 = 40$

Minimum distance of three elements

$C(b,\{c,d\}) = \min(d(b,c)+C(c,\{d\}),\ d(b,d)+C(d,\{c\}))$
$= \min(40+35,\ 10+45) = 55$

$C(b,\{c,e\}) = \min(d(b,c)+C(c,\{e\}),\ d(b,e)+C(e,\{c\}))$
$= \min(40+60,\ 80+55) = 100$

$C(b,\{d,e\}) = \min(d(b,d)+C(d,\{e\}),\ d(b,e)+C(e,\{d\}))$
$= \min(10+55,\ 45+40) = 65$

$C(c,\{b,d\}) = \min(d(c,b)+C(b,\{d\}),\ d(c,d)+C(d,\{b\}))$
$= \min(40+30,\ 15+25) = 40$

$C(c,\{b,e\}) = \min(d(c,b)+C(b,\{e\}),\ d(c,e)+C(e,\{b\}))$
$= \min(40+80,\ 25+60) = 85$

$C(c,\{d,e\}) = \min(d(c,d)+C(d,\{e\}),\ d(c,e)+C(e,\{d\}))$
$= \min(15+55,\ 25+40) = 65$

$C(d,\{b,c\}) = \min(d(d,b)+C(b,\{c\}),\ d(d,c)+C(c,\{b\}))$
$= \min(10+70,\ 15+55) = 70$

$C(d,\{b,e\}) = \min(d(d,b)+C(b,\{e\}),\ d(d,e)+C(e,\{b\}))$
$= \min(10+80,\ 20+60) = 80$

$C(d,\{c,e\}) = \min(d(d,c)+C(c,\{e\}),\ d(d,e)+C(e,\{c\}))$
$= \min(15+60,\ 20+55) = 75$

$C(e,\{b,c\}) = \min(d(e,b)+C(b,\{c\}),\ d(e,c)+C(c,\{b\}))$
$= \min(45+70,\ 25+55) = 80$

$C(e,\{b,d\}) = \min(d(e,b)+C(b,\{d\}),\ d(e,d)+C(d,\{b\}))$
$= \min(45+30,\ 20+25) = 45$

$C(e,\{c,d\}) = \min(d(e,c)+C(c,\{d\}),\ d(e,d)+C(d,\{c\}))$
$= \min(25+35,\ 20+45) = 60$

Minimum distance of four elements

$C(b,\{c,d,e\}) = \min(d(b,c)+C(c,\{d,e\}),\ d(b,d)+C(d,\{c,e\}),$
$d(b,e)+C(e,\{c,d\}))$
$= \min(40+65,\ 10+75,\ 45+60) = 85$

$C(c,\{b,d,e\}) = \min(d(c,b)+C(b,\{d,e\}),\ d(c,d)+C(d,\{b,e\}),$
$d(c,e)+C(e,\{b,d\}))$
$= \min(40+65,\ 15+80,\ 25+45) = 70$

$C(d,\{b,c,e\}) = \min(d(d,b)+C(b,\{c,e\}),\ d(d,c)+C(c,\{b,e\}),$
$d(d,e)+C(e,\{b,c\}))$
$= \min(10+100,\ 15+85,\ 20+80) = 100$

$C(e,\{b,c,d\}) = \min(d(e,b)+C(b,\{c,d\}),\ d(e,c)+C(c,\{b,d\}),$
$d(e,d)+C(d,\{b,c\}))$
$= \min(45+55,\ 25+40,\ 20+70) = 65$

Final step

$C(a,\{b,c,d,e\}) = \min(d(a,b)+C(b,\{c,d,e\}),$
$d(a,c)+C(c,\{b,d,e\}),$
$d(a,d)+C(d,\{b,c,e\}),$
$d(a,e)+C(e,\{b,c,d\}))$
$= \min(15+85,\ 30+70,\ 20+100,\ 35+65) = 100$

the optimal cost : 100

There are three cases where the value is 100,
so there are three possible routes.

There are two cases.

So one more case is added.

1st Optimal route : $d(a,b) + C(b,\{c,d,e\})$

① $d(d,c) + C(c,\{e\})$

② $d(d,e) + C(e,\{c\})$

$= d(a,b) + d(b,d) + \underline{C(d,\{c,e\})}$

$= d(a,b) + d(b,d) + \underline{d(d,c) + C(c,\{e\})}$ ①

$= d(a,b) + d(b,d) + d(d,c) + d(c,e) + d(e,phi)$

$\therefore\ a \to b \to d \to c \to e \to a$

2nd optimal route : $d(a,b) + C(b,\{c,d,e\})$

$= d(a,b) + d(b,d) + \underline{C(d,\{c,e\})}$

$= d(a,b) + d(b,d) + \underline{d(d,e) + C(e,\{c\})}$ ②

$= d(a,b) + d(b,d) + d(d,e) + d(e,c) + d(c,phi)$

$\therefore\ a \to b \to d \to e \to c \to a$

3rd optimal route : $d(a,c) + C(c,\{b,d,e\})$

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

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

$= d(a,c) + d(c,e) + d(e,d) + d(d,b) + d(b,phi)$

$\therefore\ a \to c \to e \to d \to b \to a$

4th optimal route : $d(a,e) + C(e,\{b,c,d\})$

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

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

$= d(a,e) + d(e,c) + d(c,d) + d(d,b) + d(b,phi)$

$\therefore\ a \to e \to c \to d \to b \to a$

There are four cases in total.