

알고리즘 3차 과제

그리디 - 거스름돈, 프림알고리즘

2020. 10. 22. 목

컴퓨터공학과

2019305059

이현수

■ 첫 번째 과제: 거스름돈에 대한 최소 동전수 제시

(코드-복사)

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

//타입 A 함수

```
void CoinChange_A(int cash) {  
    int change = cash; //거스름돈 총액  
    int n500 = 0, n100 = 0, n50 = 0, n10 = 0, n5 = 0, n1 = 0; //액면가별 개수 저장할 변수들  
  
    while (change >= 500){ //남은 거스름돈이 500원 이상일 경우  
        change -= 500;  
        n500++;  
    }  
    while (change >= 100){ }{ //남은 거스름돈이 100원 이상일 경우  
        change -= 100;  
        n100++;  
    }  
    while (change >= 50){ }{ //남은 거스름돈이 50원 이상일 경우  
        change -= 50;  
        n50++;  
    }  
    while (change >= 10){ }{ //남은 거스름돈이 10원 이상일 경우  
        change -= 10;  
        n10++;  
    }  
    while (change >= 5){ }{ //남은 거스름돈이 5원 이상일 경우  
        change -= 5;  
        n5++;  
    }  
    while (change >= 1){ }{ //남은 거스름돈이 1원 이상일 경우  
        change -= 1;  
        n1++;  
    }  
    printf("[규정A]Wn");  
    printf("500원 : %d, 100원 : %d, 50원 : %d, 10원 : %d, 5원 : %d, 1원 : %dWn",  
           n500, n100, n50, n10, n5, n1); //액면가별 개수 출력  
    printf("총 동전개수 : %dWn", n500 + n100 + n50 + n10 + n5 + n1);  
}
```

//타입 B 함수

```
void CoinChange_B(int cash) {  
    int change = cash; //거스름돈 총액  
    int n500 = 0, n130 = 0, n100 = 0, n50 = 0, n10 = 0, n5 = 0, n1 = 0; //액면가별 개수 저장할 변수들  
  
    while (change >= 500){ }{ //남은 거스름돈이 500원 이상일 경우  
        change -= 500;  
        n500++;  
    }  
    while (change >= 130){ }{ //남은 거스름돈이 130원 이상일 경우  
        change -= 130;  
        n130++;  
    }  
    while (change >= 100){ }{ //남은 거스름돈이 100원 이상일 경우  
        change -= 100;  
        n100++;  
    }  
    while (change >= 50){ }{ //남은 거스름돈이 50원 이상일 경우  
        change -= 50;  
        n50++;  
    }  
    while (change >= 10){ }{ //남은 거스름돈이 10원 이상일 경우  
        change -= 10;  
        n10++;  
    }
```

```

    }
    while (change >= 5){ ){ //남은 거스름돈이 5원 이상일 경우
        change -= 5;
        n5++;
    }
    while (change >= 1){ ){ //남은 거스름돈이 1원 이상일 경우
        change -= 1;
        n1++;
    }
    printf("[규정B]Wn");
    printf("500원 : %d, 130원 : %d, 100원 : %d, 50원 : %d, 10원 : %d, 5원 : %d, 1원 : %dWn",
        n500, n130, n100, n50, n10, n5, n1); //액면가별 개수 출력
    printf("총 동전개수 : %dWn", n500 + n130 + n100 + n50 + n10 + n5 + n1);
}

int main(void)
{
    int inputChange;
    printf("거스름돈 액수를 입력하세요 : ");
    scanf_s("%d", &inputChange); //거스름돈 입력
    printf("Wn");

    CoinChange_A(inputChange); //타입 A 함수 호출
    printf("Wn");
    CoinChange_B(inputChange); //타입 B 함수 호출
}

```

(코드-캡처)

```
1  #include<stdio.h>
2
3  void CoinChange_A(int cash) {
4      int change = cash;
5      int n500 = 0, n100 = 0, n50 = 0, n10 = 0, n5 = 0, n1 = 0;
6
7      while (change >= 500){
8          change -= 500;
9          n500++;
10     }
11     while (change >= 100){
12         change -= 100;
13         n100++;
14     }
15     while (change >= 50){
16         change -= 50;
17         n50++;
18     }
19     while (change >= 10){
20         change -= 10;
21         n10++;
22     }
23     while (change >= 5){
24         change -= 5;
25         n5++;
26     }
27     while (change >= 1){
28         change -= 1;
29         n1++;
30     }
31     printf("[규정A]\n");
32     printf("500원 : %d, 100원 : %d, 50원 : %d, 10원 : %d, 5원 : %d, 1원 : %d\n",
33           n500, n100, n50, n10, n5, n1);
34     printf("총 동전개수 : %d\n", n500 + n100 + n50 + n10 + n5 + n1);
35 }
36
37 void CoinChange_B(int cash) {
38     int change = cash;
39     int n500 = 0, n130 = 0, n100 = 0, n50 = 0, n10 = 0, n5 = 0, n1 = 0;
40
41     while (change >= 500){
42         change -= 500;
43         n500++;
44     }
45     while (change >= 130) {
46         change -= 130;
47         n130++;
48     }
49     while (change >= 100){
50         change -= 100;
51         n100++;
52     }
53     while (change >= 50){
54         change -= 50;
55         n50++;
56     }
57     while (change >= 10){
58         change -= 10;
59         n10++;
60     }
61     while (change >= 5){
62         change -= 5;
63         n5++;
64     }
65     while (change >= 1){
66         change -= 1;
67         n1++;
68     }
69     printf("[규정B]\n");
70     printf("500원 : %d, 130원 : %d, 100원 : %d, 50원 : %d, 10원 : %d, 5원 : %d, 1원 : %d\n",
71           n500, n130, n100, n50, n10, n5, n1);
72     printf("총 동전개수 : %d\n", n500 + n130 + n100 + n50 + n10 + n5 + n1);
73 }
74
75 int main(void)
76 {
77     int inputChange;
78     printf("거스름돈 액수를 입력하세요 : ");
79     scanf_s("%d", &inputChange);
80     printf("\n");
81
82     CoinChange_A(inputChange);
83     printf("\n");
84     CoinChange_B(inputChange);
85 }
```

(실행)

 Microsoft Visual Studio 디버그 콘솔

거스름돈 액수를 입력하세요 : 630

[규정A]


500원 : 1, 100원 : 1, 50원 : 0, 10원 : 3, 5원 : 0, 1원 : 0

총 동전개수 : 5

[규정B]

500원 : 1, 130원 : 1, 100원 : 0, 50원 : 0, 10원 : 0, 5원 : 0, 1원 : 0

총 동전개수 : 2

 Microsoft Visual Studio 디버그 콘솔

거스름돈 액수를 입력하세요 : 3420

[규정A]


500원 : 6, 100원 : 4, 50원 : 0, 10원 : 2, 5원 : 0, 1원 : 0

총 동전개수 : 12

[규정B]

500원 : 6, 130원 : 3, 100원 : 0, 50원 : 0, 10원 : 3, 5원 : 0, 1원 : 0

총 동전개수 : 12

 Microsoft Visual Studio 디버그 콘솔

거스름돈 액수를 입력하세요 : 5700

[규정A]


500원 : 11, 100원 : 2, 50원 : 0, 10원 : 0, 5원 : 0, 1원 : 0

총 동전개수 : 13

[규정B]

500원 : 11, 130원 : 1, 100원 : 0, 50원 : 1, 10원 : 2, 5원 : 0, 1원 : 0

총 동전개수 : 15

 Microsoft Visual Studio 디버그 콘솔

거스름돈 액수를 입력하세요 : 10000

[규정A]


500원 : 20, 100원 : 0, 50원 : 0, 10원 : 0, 5원 : 0, 1원 : 0

총 동전개수 : 20

[규정B]

500원 : 20, 130원 : 0, 100원 : 0, 50원 : 0, 10원 : 0, 5원 : 0, 1원 : 0

총 동전개수 : 20

 Microsoft Visual Studio 디버그 콘솔

거스름돈 액수를 입력하세요 : 20380

[규정A]

500원 : 40, 100원 : 3, 50원 : 1, 10원 : 3, 5원 : 0, 1원 : 0

총 동전개수 : 47

[규정B]

500원 : 40, 130원 : 2, 100원 : 1, 50원 : 0, 10원 : 2, 5원 : 0, 1원 : 0

총 동전개수 : 45

■ 두 번째 과제: 최소신장트리 도출(Prim 알고리즘 적용)

(코드-복사)

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

//그래프 정점 이름이 담긴 배열
char vertexName[] =
{ 'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z' };

typedef struct { //그래프 구조체
    int V;
    int E;
    int** M;
}graphType;
typedef graphType* graphPtr;

int** InintMatrix(int Row, int Ncol, int Value) //그래프 인접행렬 동적할당
{
    int** m;
    int i, j;
    m = malloc(Row * sizeof(int*));
    for (i = 0; i < Row; i++) {
        m[i] = malloc(Ncol * sizeof(int));
    }
    for (i = 0; i < Row; i++) {
        for (j = 0; j < Ncol; j++) {
            m[i][j] = Value;
        }
    }
    return m;
}

graphPtr InintGraph(int V) //그래프 초기화
{
    graphPtr G = malloc(sizeof(graphType));
    G->V = V; //정점 개수 V
    G->E = 0; //간선 개수 0
    G->M = InintMatrix(V, V, 0);
    return G;
}

void InsertEdge(graphPtr graph, int V1, int V2, int weight) //graph 정점 V1, V2를 가중치 weight인 간선 추가
{
    if (graph->M[V1][V2] == 0 && graph->M[V2][V1] == 0) //간선이 연결되었지 않을 경우
    {
        graph->E++; //간선 개수 1 추가
        graph->M[V1][V2] = weight;
        graph->M[V2][V1] = weight; //무향그래프이므로 (V1, V2), (V2, V1)에 weight 저장
    }
    else //이미 다른값이 있을 경우 나중에 들어온값을 갱신
    {
        graph->M[V1][V2] = weight;
        graph->M[V2][V1] = weight;
    }
}

void FreeMatrix(graphPtr graph){ //그래프 해체
    int i;
    for (i = 0; i < graph->V; i++) {
        free(graph->M[i]);
    }
    free(graph->M);
}
```

```

void PrintGraph(graphPtr graph){//그래프 출력 함수
    int i, j;
    printf("\n");
    printf(" ");
    for (i = 0; i < graph->V; i++)
    {
        printf("%2c ", vertexName[i]); //정점이름 출력
    }
    printf("\n");
    for (i = 0; i < graph->V; i++) {
        printf("%c ", vertexName[i]);
        for (j = 0; j < graph->V; j++) {
            printf("%02d ", graph->M[i][j]); //2칸 간격으로 빈공간은 0으로 채워넣음
        }
        printf("\n");
    }
    printf("\n");
}

/////////////////////////////////////////////////// ↓프림알고리즘 관련 코드
int searchMinVertex(int size, int* Distance, int* Selected)
{
    //최소신장트리에 속하지 않은 점 v중 Distance[v]가 최소인 점 찾아 반환하는 함수
    int i, minVertex;
    for (i = 0; i < size; i++)
    {
        if (Selected[i] == 0 && Distance[i] != -1)//MST에 속하지 않으면서 Distance가 무한대가 아닐경우
        {
            minVertex = i; //거리가 최소라고 가정->다음 반복문에서 기준점
            break;
        }
    }
    for (i = 0; i < size; i++) //MST에 속하지않고, Distance가 무한대가 아니면서 거리가 가장짧은 점 찾기
    {
        if (Selected[i] == 0 && Distance[i] != -1 && (Distance[i] < Distance[minVertex])) minVertex = i;
    }
    return minVertex; //최소점 반환
}

void updateDistance(graphPtr graph, int* Distance, int* Selected, int* LinkVertex, int minVertex)
{
    //최소신장트리에 속하지 않은 각점 중 minVertex점과의 간선 가중치가 Distance에 있는 값보다 작으면 갱신하는 함수
    int i;
    for (i = 0; i < graph->V; i++)
    {
        if (Selected[i] == 0) //MST에 속하지 않는점만 거리 갱신
        {
            if (Distance[i] == -1) //Distance가 무한대일 경우
            {
                //최소점,minVertex점(최근에 MST에 추가된점)과 i점과 간선이 있을 때
                if (graph->M[i][minVertex] > 0)
                {
                    Distance[i] = graph->M[i][minVertex]; //Distance에 추가
                    LinkVertex[i] = minVertex;//추가한 간선거리가 어떤 점과의 간선인지저장
                }
            }
            else //Distance가 무한대가 아니고 거리가 있을경우
            {
                //최소점과 i점이 연결되었으면서 그 거리가 Distance에 있는 거리보다 작을 때 갱신
                if (graph->M[i][minVertex] != 0 && graph->M[i][minVertex] < Distance[i])
                {
                    Distance[i] = graph->M[i][minVertex];
                    LinkVertex[i] = minVertex;
                }
            }
        }
    }
}

```

```

graphPtr Prim(graphPtr graph) //프림 알고리즘 함수
{
    int i;
    int minVertex; //간선거리가 최소점 변수
    int intStartVertex; //시작점변수
    char chStartVertex; //시작점입력변수

    graphPtr MST = InintGraph(graph->V); //최소신장 트리용 인접행렬 그래프

    int* Distance = malloc(sizeof(int) * graph->V); //간선가중치 저장 동적배열
    int* LinkVertex = malloc(sizeof(int) * graph->V); //그 간선가중치와 연결된 정점 저장
    int* Selected = malloc(sizeof(int) * graph->V); //최소신장트리에 포함된 정점인지 아닌지 저장(포함됨=1, 안됨=0)

    for (i = 0; i < graph->V; i++) Distance[i] = -1; //초기에 모두 -1로 저장(-1은 ∞의미)
    for (i = 0; i < graph->V; i++) Selected[i] = 0; //초기에 모두 0으로 저장->아무것도 포함안됐기때문

    printf("시작 정점을 입력하세요 : "); scanf_s("%c", &chStartVertex); //시작정점 대문자 알파벳으로 입력
    intStartVertex = chStartVertex - 65; //입력받은 문자를 숫자로 변환 A는 0, B는 1 ... 변환

    Distance[intStartVertex] = 0; //임의의 시작점 intStartVertex Distance 0으로 저장
    Selected[intStartVertex] = 1; // 임의의 시작점을 Selected배열에 1로 저장해 최소신장트리 정점으로 추가

    for (i = 0; i < graph->V; i++) //시작점이 아닌 점에대해 배열 Distance 초기화
    {
        if (i != intStartVertex && graph->M[intStartVertex][i] > 0) //자기자신이 아니고 간선이 존재하면
        {
            Distance[i] = graph->M[intStartVertex][i]; //Distance 배열에 가중치 저장
            LinkVertex[i] = intStartVertex; //그 가중치를 가지는 간선이 어떤 점과 연결됐는지 저장
        }
    }

    while (MST->E < graph->V - 1) //최소신장 트리 간선개수가 입력그래프 (정점개수-1) 경우 반복 종료
    {
        minVertex = searchMinVertex(graph->V, Distance, Selected); //MST에 속하지 않으면서 거리최소인점찾기
        Selected[minVertex] = 1; //최소점minVertex가 MST에 포함됐다는 표시
        InsertEdge(MST, minVertex, LinkVertex[minVertex], Distance[minVertex]); //최소신장트리에 추가

        updateDistance(graph, Distance, Selected, LinkVertex, minVertex); //minVertex점에 대한 거리갱신
    }

    free(Distance);
    free(LinkVertex);
    free(Selected); //동적배열한 배열 3개 해체
    return MST; //최소신장트리 MST 반환
}

void SetEdge(graphPtr graph) //각정점마다 임의의 4개정점과 간선 갖게하는 함수
{
    srand(time(NULL)); //난수생성을 위한 선언
    int* countEdge = malloc(sizeof(int) * graph->V); //각 정점이 가지는 간선개수 저장배열
    int i, j, edge, m, value;
    for (i = 0; i < graph->V; i++) countEdge[i] = 0; //모두 0으로 초기화
    for (i = 0; i < graph->V; i++)
    {
        m = 0;
        while (countEdge[i] < 4) //각 정점이 가지는 간선개수가 4개미만이면 무한반복
        {
            edge = rand() % graph->V; //연결한 정점 랜덤값으로 저장
            while(edge==i) edge = rand() % graph->V; //연결할 정점이 자기자신이 아닐때까지 무한반복
            value = rand() % 50; //간선의 가중치 랜덤하게 저장
            if (countEdge[edge] == 4 && graph->M[edge][i]!=0)
            { //i점과 추가할 점이 이미 개수가 4개이지만 이미간선이 있어서 간선을 갱신하는 경우
                InsertEdge(graph, edge, i, value);
                m++;
            }
        }
    }
}

```



```

    else if(countEdge[edge]<4 && graph->M[edge][i]==0)
    { //i점과 추가할 점이 이미 개수가 4개미만, 간선이 없는경우
        InsertEdge(graph, edge, i, value);
        m++;
        countEdge[i] += 1;
        countEdge[edge] += 1; //각각 점의 간선개수 추가
    }
    else if (countEdge[edge] < 4 && graph->M[edge][i] != 0) {
        //i점과 추가할 점이 이미 개수가 4개미만, 간선이 있는경우 간선가중치만 갱신
        m++;
        InsertEdge(graph, edge, i, value);
    }
    if (m == 4)break; //저장,갱신 동작이 4번이 되면 반복문 빠져나감.
    //a-b, b-a가 나왔을 때 마지막으로 나온 b-a의 가중치를저장->정점중에 3개이하간선 가능
}

}

int main(void)
{
    graphType* Graph = InintGraph(26); //입력 그래프 만들기

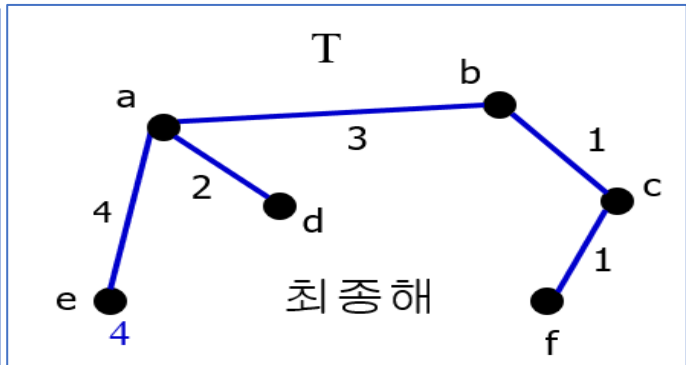
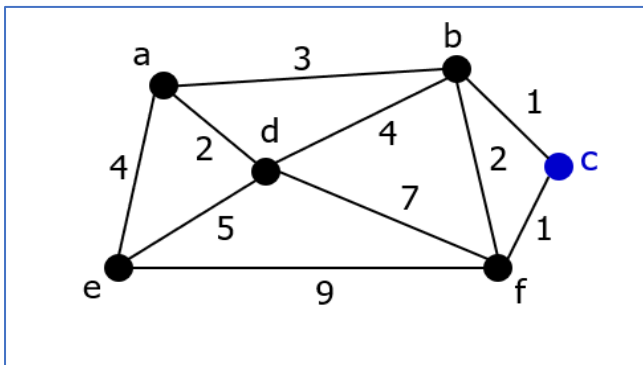
    SetEdge(Graph); //간선 만들기
    PrintGraph(Graph); //입력 그래프 출력

    graphType* MST= Prim(Graph); //프림알고리즘 적용
    PrintGraph(MST); //최소신장트리 용 그래프 해체
    printf("간선 수 : %d\n", MST->E); //최소신장트리 용 그래프 간선 개수

    FreeMatrix(Graph); //입력 그래프 해체
    FreeMatrix(MST); //최소신장트리 용 그래프 해체
}

```

(증명)



```

int main(void)
{
    graphType* Graph = InintGraph(6);
    InsertEdge(Graph, 0, 1, 3);
    InsertEdge(Graph, 1, 2, 1);
    InsertEdge(Graph, 2, 5, 1);
    InsertEdge(Graph, 4, 5, 9);
    InsertEdge(Graph, 4, 0, 4);
    InsertEdge(Graph, 1, 5, 2);
    InsertEdge(Graph, 0, 3, 2);
    InsertEdge(Graph, 3, 4, 5);
    InsertEdge(Graph, 1, 3, 4);
    InsertEdge(Graph, 3, 5, 7);
    //SetEdge(Graph);
    PrintGraph(Graph);

    graphType* MST= Prim(Graph);
    PrintGraph(MST);
    printf("간선 수 : %d\n", MST->E);

    FreeMatrix(Graph);
    FreeMatrix(MST);
}

```

Microsoft Visual Studio 디버그 콘솔

	A	B	C	D	E	F
A	00	03	00	02	04	00
B	03	00	01	04	00	02
C	00	01	00	00	00	01
D	02	04	00	00	05	07
E	04	00	00	05	00	09
F	00	02	01	07	09	00

시작 정점을 입력하세요 : E

	A	B	C	D	E	F
A	00	03	00	02	04	00
B	03	00	01	00	00	00
C	00	01	00	00	00	01
D	02	00	00	00	00	00
E	04	00	00	00	00	00
F	00	00	01	00	00	00

간선 수 : 5

ppt에 있는 예제를 실행시켜
보니 정상적으로 동작한다.

(소스-캡처)

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<time.h>
4
5  char vertexName[] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z' };
6  typedef struct
7  {
8      int V;
9      int E;
10     int** M;
11 }graphType;
12 typedef graphType* graphPtr;
13
14 int** InintMatrix(int Row, int Ncol, int Value) {
15     int** m;
16     int i, j;
17     m = malloc(Row * sizeof(int*));
18     for (i = 0; i < Row; i++) {
19         m[i] = malloc(Ncol * sizeof(int));
20     }
21     for (i = 0; i < Row; i++) {
22         for (j = 0; j < Ncol; j++) {
23             m[i][j] = Value;
24         }
25     }
26     return m;
27 }
28
29 graphPtr InintGraph(int V) {
30     graphPtr G = malloc(sizeof(graphType));
31     G->V = V;
32     G->E = 0;
33     G->M = InintMatrix(V, V, 0);
34     return G;
35 }
36
37 void InsertEdge(graphPtr graph, int V1, int V2, int weight) {
38     if (graph->M[V1][V2] == 0 && graph->M[V2][V1] == 0)
39     {
40         graph->E++;
41         graph->M[V1][V2] = weight;
42         graph->M[V2][V1] = weight;
43     }
44     else
45     {
46         graph->M[V1][V2] = weight;
47         graph->M[V2][V1] = weight;
48     }
49 }
50
51 void FreeMatrix(graphPtr graph) {
52     int i;
53     for (i = 0; i < graph->V; i++) {
54         free(graph->M[i]);
55     }
56     free(graph->M);
57 }
58
59 void PrintGraph(graphPtr graph)
60 {
61     int i, j;
62     printf("\n");
63     printf(" ");
64     for (i = 0; i < graph->V; i++)
65     {
66         printf("%2c ", vertexName[i]);
67     }
68     printf("\n");
69     for (i = 0; i < graph->V; i++) {
70         printf("%c ", vertexName[i]);
71         for (j = 0; j < graph->V; j++) {
72             printf("%02d ", graph->M[i][j]);
73         }
74         printf("\n");
75     }
76     printf("\n");
}
```

```

77     }
78
79     int searchMinVertex(int size, int* Distance, int* Selected)
80     {
81         int i, minVertex;
82         for (i = 0; i < size; i++)
83         {
84             if (Selected[i] == 0 && Distance[i] != -1)
85             {
86                 minVertex = i;
87                 break;
88             }
89         }
90         for (i = 0; i < size; i++)
91         {
92             if (Selected[i] == 0 && Distance[i] != -1 && (Distance[i] < Distance[minVertex])) minVertex = i;
93         }
94         return minVertex;
95     }
96
97     void updateDistance(graphPtr graph, int* Distance, int* Selected, int* LinkVertex, int minVertex)
98     {
99         int i;
100         for (i = 0; i < graph->V; i++)
101         {
102             if (Selected[i] == 0)
103             {
104                 if (Distance[i] == -1)
105                 {
106                     if (graph->M[i][minVertex] > 0)
107                     {
108                         Distance[i] = graph->M[i][minVertex];
109                         LinkVertex[i] = minVertex;
110                     }
111                 }
112                 else
113                 {
114                     if (graph->M[i][minVertex] != 0 && graph->M[i][minVertex] < Distance[i])
115                     {
116                         Distance[i] = graph->M[i][minVertex];
117                         LinkVertex[i] = minVertex;
118                     }
119                 }
120             }
121         }
122     }
123
124     graphPtr Prim(graphPtr graph)
125     {
126         int i;
127         int minVertex;
128         int intStartVertex;
129         char chStartVertex;
130
131         graphPtr MST = InintGraph(graph->V);
132
133         int* Distance = malloc(sizeof(int) * graph->V);
134         int* LinkVertex = malloc(sizeof(int) * graph->V);
135         int* Selected = malloc(sizeof(int) * graph->V);
136
137         for (i = 0; i < graph->V; i++) Distance[i] = -1;
138         for (i = 0; i < graph->V; i++) Selected[i] = 0;
139
140         printf("시작 정점을 입력하세요 : "); scanf_s("%c", &chStartVertex);
141         intStartVertex = chStartVertex - 65;
142
143         Distance[intStartVertex] = 0;
144         Selected[intStartVertex] = 1;
145
146         for (i = 0; i < graph->V; i++)
147         {
148             if (i != intStartVertex && graph->M[intStartVertex][i] > 0)
149             {
150                 Distance[i] = graph->M[intStartVertex][i];
151                 LinkVertex[i] = intStartVertex;
152             }
153         }
154
155         while (MST->E < graph->V - 1)
156         {
157             minVertex = searchMinVertex(graph->V, Distance, Selected);
158             Selected[minVertex] = 1;
159             InsertEdge(MST, minVertex, LinkVertex[minVertex], Distance[minVertex]);
160
161             updateDistance(graph, Distance, Selected, LinkVertex, minVertex);

```

```

162     }
163
164     free(Distance);
165     free(LinkVertex);
166     free(Selected);
167     return MST;
168 }
169
170 void SetEdge(graphPtr graph)
171 {
172     srand(time(NULL));
173     int* countEdge = malloc(sizeof(int) * graph->V);
174     int i, j, edge, m, value;
175     for (i = 0; i < graph->V; i++) countEdge[i] = 0;
176     for (i = 0; i < graph->V; i++)
177     {
178         m = 0;
179         while (countEdge[i] < 4)
180         {
181             edge = rand() % graph->V;
182             while(edge==i) edge = rand() % graph->V;
183             value = rand() % 50;
184             if (countEdge[edge] >= 4 && graph->M[edge][i]!=0)
185             {
186                 InsertEdge(graph, edge, i, value);
187                 m++;
188             }
189             else if(countEdge[edge]<4 && graph->M[edge][i]==0)
190             {
191                 InsertEdge(graph, edge, i, value);
192                 m++;
193                 countEdge[i] += 1;
194                 countEdge[edge] += 1;
195             }
196             else if (countEdge[edge] < 4 && graph->M[edge][i] != 0) {
197                 m++;
198                 InsertEdge(graph, edge, i, value);
199             }
200             if (m == 4)break;
201         }
202     }
203 }
204
205 int main(void)
206 {
207     graphType* Graph = InintGraph(26);
208     SetEdge(Graph);
209     PrintGraph(Graph);
210
211     graphType* MST= Prim(Graph);
212     PrintGraph(MST);
213     printf("간선 수 : %d\n", MST->E);
214
215     FreeMatrix(Graph);
216     FreeMatrix(MST);
217 }

```

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	00	43	00	00	00	00	00	29	00	00	00	00	00	13	00	00	00	00	00	00	00	00	00	00	00	00
B	43	00	17	00	00	00	00	00	00	00	21	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	17	00	00	00	00	37	00	00	00	00	00	00	00	00	00	00	00	00	00	26	00	00	00	40	00
D	00	00	00	00	00	00	00	00	00	00	00	21	00	34	00	00	00	00	14	00	01	00	00	00	00	00
E	00	00	00	00	02	00	08	00	00	31	00	34	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	02	00	00	00	00	00	04	00	00	00	38	00	00	00	00	45	00	00	00	00	00	00	00
G	00	37	00	00	00	00	00	17	00	00	00	07	00	00	00	00	00	00	00	00	00	00	00	00	00	12
H	29	00	00	08	00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	00	00	00	00	00
I	00	00	00	00	00	17	00	00	00	00	00	00	25	00	00	00	00	00	00	00	00	00	36	17	00	00
J	00	00	00	00	00	00	00	00	00	02	00	00	00	33	00	00	00	00	00	00	11	00	36	00	00	00
K	00	21	00	31	04	00	00	00	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
L	00	00	00	00	00	00	00	00	00	00	00	00	02	00	00	00	00	29	00	00	00	39	00	00	11	
M	00	00	21	34	00	07	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	29	00	00	00	
N	13	00	00	00	00	00	00	25	00	00	02	00	00	04	00	00	00	00	00	00	00	00	00	00	00	00
O	00	00	34	00	38	00	00	00	33	00	00	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00
P	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	37	11	00	25	00	00	00	00	00	32	
Q	00	00	00	00	00	00	00	00	00	00	00	00	00	00	37	00	00	00	06	00	29	00	00	00	21	
R	00	00	00	00	00	00	00	00	00	00	00	00	00	11	00	00	00	00	00	22	07	00	00	00	00	
S	00	00	14	00	00	00	01	00	00	00	29	00	00	00	00	00	00	00	35	00	00	00	00	00	00	
T	00	00	00	00	45	00	00	00	00	00	00	00	00	25	06	00	35	00	00	00	00	00	00	00	00	
U	00	26	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	05	00	31	00	00	
V	00	00	00	00	00	00	00	00	11	00	00	00	00	29	22	00	00	05	00	00	00	00	00	00	00	
W	00	00	00	00	00	00	00	36	00	00	39	29	00	00	00	00	07	00	00	00	00	00	00	00	00	
X	00	00	00	00	00	00	00	17	36	00	00	00	00	00	00	00	00	00	00	31	00	00	00	45	00	
Y	00	00	40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	45	00	00	
Z	00	00	00	00	00	12	00	00	00	00	11	00	00	32	21	00	00	00	00	00	00	00	00	00	00	

시작 정점을 입력하세요 : K

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	00	00	00	00	00	00	00	00	00	00	00	00	13	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	17	00	00	00	00	00	00	00	21	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	17	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	40	00
D	00	00	00	00	00	00	00	00	00	00	00	21	00	00	00	00	00	00	00	01	00	00	00	00	00	00
E	00	00	00	00	02	00	08	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	02	00	00	00	00	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
G	00	00	00	00	00	00	00	17	00	00	00	07	00	00	00	00	00	00	00	00	00	00	00	00	00	12
H	00	00	00	08	00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	00	00	00	00	00	00
I	00	00	00	00	00	17	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	17	00	00	
J	00	00	00	00	00	00	00	00	00	02	00	00	00	00	00	00	00	00	00	00	11	00	00	00	00	
K	00	21	00	00	04	00	00	00	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
L	00	00	00	00	00	00	00	00	00	00	00	02	00	00	00	00	00	00	00	00	00	00	00	00	11	
M	00	00	21	00	07	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
N	13	00	00	00	00	00	00	00	00	00	02	00	00	04	00	00	00	00	00	00	00	00	00	00	00	00
O	00	00	00	00	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00
P	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	11	00	00	00	00	00	00	00	00	00	00
Q	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	06	00	00	00	00	00	00	21	
R	00	00	00	00	00	00	00	00	00	00	00	00	00	11	00	00	00	00	00	22	07	00	00	00	00	00
S	00	00	00	00	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
T	00	00	00	00	00	00	00	00	00	00	00	00	00	00	06	00	00	00	00	00	00	00	00	00	00	00
U	00	00	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	05	00	00	00	00	00
V	00	00	00	00	00	00	00	00	11	00	00	00	00	29	22	00	00	05	00	00	00	00	00	00	00	00
W	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	07	00	00	00	00	00	00	00	00	00
X	00	00	00	00	00	00	00	17	36	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Y	00	00	40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Z	00	00	00	00	00	12	00	00	00	00	11	00	00	32	21	00	00	00	00	00	00	00	00	00	00	00

간선 수 : 25

D:\wc\드라이브대체\바탕화면\4학기 공부\4학기\Debug\4학기.exe(프로세스 14368개)이(가) 디버깅이 중지될 때 콘솔을 자동으로 닫으려면 [도구] -> [옵션] -> [디버깅] > [디버깅이 창을 닫으려면 아무 키나 누르세요]...

(소감)

첫번째 동전개수 과제는 쉬웠지만, 두번째 과제는 많이 어려웠다. 인터넷에서 Selected 배열 사용 등 아이디어를 얻어서 ppt에 나와있는 의사코드로 되있는 알고리즘을 보면서 만들어 나갔다. 여러 예제를 만들어서 잘 동작하는지 해봤는데 오류가 나서 5번정도 수정끝에 완벽하게 동작되서 뿌듯했다.

(이름짓기에 참고한 자료 출처)

<https://blog.naver.com/mtrca/89632235>

이름짓는 법 [변수, 함수, 필드 등] | PRG개발 / 컴퓨터,네트워크

2009. 9. 23. 15:17

<https://blog.naver.com/mtrca/89632235> 복사

번역하기

1. 변수 이름 짓는 법

① 헝가리안 표기법 : 변수이름 앞에 변수의 타입을 첨가하고 데이터타입은 반드시 소문자를 사용 **무따기**
예) intStartNum [int_데이터타입, StartNum_변수이름]

② 카멜케이스 (변수 이름을 지을 때 사용) : 첫 문자를 제외한 모든 단어의 시작에 대문자사용_낙타등처럼 울퉁불퉁해지죠~~ **무따기**
예) intStartNum

2. 함수 이름 짓는 법

① 파스칼케이스 (함수이름을 지을때 사용) : 첫 문자를 포함한 모든 단어의 시작에 대문자사용 **무따기**
예) ReserveNumber()

변수 이름은 카멜케이스 방법을 이용했다.