

알고리즘 2021 보고서

보고서 제출서약서

나는 숭실대학교 컴퓨터학부의 일원으로 명예를 지키면서 생활하고 있습니다.

나는 보고서를 작성하면서 다음과 같은 사항을 준수하였음을 엄숙히 서약합니다.

1. 나는 자력으로 보고서를 작성하였습니다.

1.1. 나는 동료의 보고서를 베끼지 않았습니다.

1.2. 나는 비공식적으로 얻은 해답/해설을 기초로 보고서를 작성하지 않았습니다.

2. 나는 보고서에서 참조한 문헌의 출처를 밝혔으며 표절하지 않았습니다. (나는 특히 인터넷에서 다운로드한 내용을 보고서에 거의 그대로 복사하여 사용하지 않았습니다.)

3. 나는 보고서를 제출하기 전에 동료에게 보여주지 않았습니다.

4. 나는 보고서의 내용을 조작하거나 날조하지 않았습니다.

과목	알고리즘 2021
과제명	Prim Algorithm과 Kruskal Algorithm을 이용한 MST찾기
담당교수	최 재 영 교 수
제출인	컴퓨터학부 20162449 김상현 [307]
제출일	2021년 11월 4일

차 례

1장 프로젝트 동기/목적

2장 설계/구현 아이디어

3장 수행결과(구현 화면 포함)

4장 디버깅

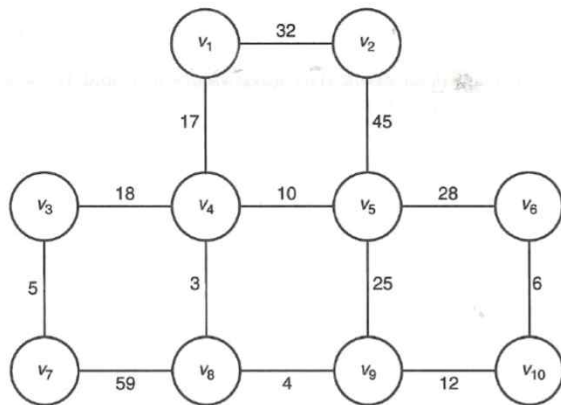
5장 소스코드(+주석)

1. 프로젝트 동기 및 목적

프림 알고리즘을 이해하고 MST (최소 신장 트리)를 구할 수 있도록 구현한다.

크루스칼 알고리즘을 이해하고 MST (최소 신장 트리)를 구할 수 있도록 구현한다.

2. 설계 구현 아이디어 (2번 문제)



주어진 그래프를 프로그래밍 상에서 구현하기 위해서 행렬로 표현한다.

변환 후 인접행렬로 나타냄 ->

```
int weight[MAX_VERTICES_PRIM][MAX_VERTICES_PRIM] = { // 신장트리의 거리와 모양을 배열로 표현, INF는 바로갈 수 있는 경로가 없음을 뜻한다.
    { 0, 32, INF, 17, INF, INF, INF, INF, INF, INF },
    { 32, 0, INF, INF, 45, INF, INF, INF, INF, INF },
    { INF, INF, 0, 18, INF, INF, 5, INF, INF, INF },
    { 17, INF, 18, 0, 10, INF, INF, 3, INF, INF },
    { INF, 45, INF, 10, 0, 28, INF, INF, 25, INF },
    { INF, INF, INF, INF, 28, 0, INF, INF, INF, 6 },
    { INF, INF, 5, INF, INF, INF, 0, 59, INF, INF },
    { INF, INF, INF, 3, INF, INF, 59, 0, 4, INF },
    { INF, INF, INF, INF, 25, INF, INF, 4, 0, 12 },
    { INF, INF, INF, INF, INF, 6, INF, INF, 12, 0 }
};
```

먼저 **프림 알고리즘**을 간단히 그림으로 이해 해 보려고 한다. (덧장)

정점의 개수 만큼의 크기의 dist[] 배열을 선언한다. 정점간의 거리를 저장하는데 더 길이가 짧은 간선이 존재한다면 업데이트 하게 된다.

1. dist[] 배열은 시작 정점에 대해서 0으로 초기화 하고 나머지 모든 값을 INF 값으로 넣어준다.
2. 시작 정점에서 이어져 있는 모든 간선들의 가중치를 dist배열에 입력한다. 이제 배열에 들어있는 값 중 제일 작은 값을 찾고 그 가중치를 선택한다.
3. 선택한 가중치에 해당하는 정점에서 이어져 있는 간선들의 가중치를 다시 dist 배열에 입력한다. 이 때 이미 선택이 되었던 정점의 경우는 넣어도 의미 없으므로 넣지 않도록 하자. 또한 dist배열에 이미 들어있는 값이 새롭게 찾은 가중치의 값보다 큰 경우 dist배열의 값을 업데이트 해준다.
4. 위 과정을 반복하고 (총 정점의 개수 - 1) 만큼 반복하게 되면 모든 정점을 선택하였고 종료하게 된다.

dist[i]	1	2	3	4	5	6	7	8	9	10	MST
	0	32	17								(1,4)
			18	10				3			(4,8)
							39		4		(8,9)
				10						12	(5,4)
		32	18		28	39			12		(9,10)
					6						(10,6)
		32	18			59					(4,3)
							5				(3,7)
	32										(1,2)

<- 직접 손으로 표현한 MST 과정

```
int get_min_vertex(int n)
{
    int v, i; // 정점의 정보를 저장할 변수 v, 반복문을 위한 변수 i
    for (i = 0; i < n; i++)
    {
        if (selected[i] == FALSE) {
            v = i; // 아직 선택되지 않은 정점의 번호를 v에 저장, 각 함수 실행별 0부터 n-1까지 차례대로 저장된다.
            break;
        }
    }

    // 위에서 선택된 정점이 과연 최소거리를 지니고 있는 정점인지를 확인한다.
    for (i = 0; i < n; i++)
    {
        // 선택되지 않은 정점들을 순회하면서 최소거리를 가진 정점을 찾아낸다.
        if (selected[i] == FALSE && (dist[i] < dist[v]))
            v = i; // 더 적은 거리가 존재한다면 해당 정점을 저장한다.
    }

    return(v); // 최소의 거리를 갖는 정점이 선택됐으므로 정점 번호를 리턴한다.
}
```

get_min_vertex() : 각 반복마다 최소 거리를 갖는 정점을 찾는 함수

```
void prim(int s, int n){
    make_arr(n);
    int i, u, v;
    for (u = 0; u < n; u++){ // dist배열과 selected배열의 정보를 초기화
        dist[u] = INF;
        selected[u] = FALSE;
    }
    dist[s] = 0; // 시작정점과 시작정점간의 거리는 0이다, 자기자신을 순환하는 경로는 없다고 가정한다.
    int min = 0;
    for (i = 0; i < n; i++){
        // 리턴된 정점 번호를 u에 저장한다. u는 최소거리를 가지는 정점.
        u = get_min_vertex(n);
        selected[u] = TRUE; // 해당 정점을 선택했다고 표시.

        for (v = 0; v < n; v++){ // 새로운 정점에서 얻은 이어진 가중치 정보 dist배열 업데이트
            // 선택된 u 정점을 기준으로 정점(u)과 연결되어 있는 정점까지의 거리를 각각 비교.
            if (weight[u][v] != INF) { // 정점 u와 연결이 되어있고
                // 아직 선택되지 않았으며 해당 변(weight[u][v])의 길이가 기존의 dist[v] 값보다 작다면
                if (selected[v] == FALSE && weight[u][v] < dist[v])
                    dist[v] = weight[u][v]; // dist[v]의 값을 갱신.
            }
        }
        min += dist[u]; // 최소거리 산출
        printf_set(dist[u]);
        printf("선택한 가중치: %d\n-->", dist[u]);
        printf("현재 최소값: %d\n", min);
    }
    printf("\n최소값: %d\n", min);
}
```

prim() 함수 : get_min_vertex() 함수에서 찾은 정점에서 얻는 새로운 가중치 정보를 이미 선택한 정점이 아니고 기존의 dist 배열 값보다 작으면 업데이트 해준다.

다음은 **Kruscal** 알고리즘이다. 크루스칼 알고리즘의 가장 기초의 개념은 간선 가중치를 오름차순으로 정렬하여 가중치가 낮은 정점부터 차례로 연결 시키는 것이다. 물론 제일 중요한 것은 정점을 연결하였을 때 아직 연결하지 못한 정점들이 남아 있을 때는 사이클이 생기면 안된다는 것이다.

먼저 모든 간선들을 가중치와 시작 정점, 끝 정점을 모두 int 형식을 담을 수 있는 구조체에 담는다.

```
typedef struct {
    int src; //시작정점 번호
    int dest; //끝 정점 번호
    int weight; //가중치
}forSort;

forSort arr[MAX_VER] = {0}; // 간선 별로 시작 정점과 끝 정점 , 가중치를 저장하기 위한 구조체 배열
```

이제 C library 함수 qsort() 함수를 이용하여 arr 구조체 배열을 weight (가중치) 기준으로 오름차순 정렬한다.

```
while (edge_count < (n - 1)) {
    if (i < index) {
        min = arr[i].weight;
        uiset = find_set(arr[i].src);
        vset = find_set(arr[i].dest);
        if (uiset != vset) {
            edge_count++;
            union_set(uiset, vset);
            mincost += min;
            print_set(min);
            printf("선택한 가중치: %d\t-->", min);
            printf("현재 최소값 : %d\n", mincost);
        }
        i++;
    }
}
```

Kruscal() 함수에서의 최종 반복문 :

- find_set() 함수를 이용하여 arr[] 구조체에 저장되어있는 시작 정점, 끝 정점 두 개의 정점의 제일 위에 있는 부모 노드를 찾게 된다.
- union_set() 함수를 이용하여 두 개가 속한 집합이 다를 때 두 집합을 합치고 최소 비용 트리에 포함하여 가중치 값을 계산하는데 이용된다.

```
int find_set(int vertex) {
    int parent_value, p, i = -1;
    for (i = vertex; (parent_value = parent[i]) >= 0; i = parent_value)
        ; // 원소의 parent를 찾아감
    p = i; //parent를 발견해냄.
    for (i = vertex; (parent_value = parent[i]) >= 0; i = parent_value)
        parent[i] = p; // 원소의 parent
    return p;
}
```

find_set() 함수 : 사이클이 형성되는지 확인하는 함수이다.

첫 for 문에서는 parent[]의 값이 음수가 나올 때까지 확인한다. 음수가 나오면 해당 정점을 포함한 트리의 최상위 부모라는 뜻이기에 그 값을 저장한다.

두 번째 for문에서는 그 최상위 부모에 연결되어 있는 노드들의 parent[]를 모두 저장한 값으로 바꾸어 최상

위 부모를 알려준다. 또한 해당 저장 값을 리턴하여 최상위 정점의 index를 반환한다.

```
void union_set(int r1, int r2) {
    if (num[r1] < num[r2]) {
        parent[r1] = r2; //r2 = 부모노드가 될것 r1 = 자식
        num[r2] += num[r1]; // r2의 갯수가 r1의 집합의 갯수만큼 늘어남.
    }
    else {
        parent[r2] = r1;
        num[r1] += num[r2];
    }
}
```

union_set() 함수 : 두 트리를 연결하기 위해서 실행되는 함수이다.

연결시킬 두 개의 정점 r1과 r2 중 집합 내부의 정점의 수가 더 큰 쪽을 기준으로 해서 그쪽에 작은 트리를 연결시킨다. 또한 연결되는 쪽의 정점의 부모를 연결할 정점으로 바꾸어 주어야하고, 연결된 트리의 집합의 개수도 두 정점의 num의 값을 더해준다.

3. 수행결과 (문제 1번 / 문제 2번 / 문제 3번)

문제 1번 .

```
Microsoft Visual Studio 디버그 콘솔
1. 프림 알고리즘 2. 크루스칼 알고리즘 : 1

-----
프림 알고리즘을 이용한 최소 신장 트리 찾기
선택한 가중치: 0      ->현재 최소값 : 0
(1, 4) : 선택한 가중치: 17      ->현재 최소값 : 17
(4, 8) : 선택한 가중치: 3       ->현재 최소값 : 20
(8, 9) : 선택한 가중치: 4       ->현재 최소값 : 24
(4, 5) : 선택한 가중치: 10      ->현재 최소값 : 34
(9, 10) : 선택한 가중치: 12     ->현재 최소값 : 46
(6, 10) : 선택한 가중치: 6      ->현재 최소값 : 52
(3, 4) : 선택한 가중치: 18      ->현재 최소값 : 70
(3, 7) : 선택한 가중치: 5       ->현재 최소값 : 75
(1, 2) : 선택한 가중치: 32      ->현재 최소값 : 107

최소값 : 107

C:\Users\#tticj\source\repos\SUB6\Debug\SUB6.exe(프로세스 5328개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

* 처음 시작 정점이 포함된 트리는 가중치가 0이기 때문에 간선을 나타내는 시작 정점과 끝 정점이 없다.

문제 2번.

```
case 1:
    printf("프림 알고리즘을 이용한 최소 신장 트리 찾기\n");
    //prim(0, 10); // 정점 개수가 7개인 그래프에서 0번 정점을 출력
    prim(8, 10);
    break;
```

main() 함수에서 시작 정점을 8로 바꾸고 실행하였다. (뒷장)

```
Microsoft Visual Studio 디버그 콘솔
1. 프림 알고리즘 2. 크루스칼 알고리즘 : 1

-----
프림 알고리즘을 이용한 최소 신장 트리 찾기
선택한 가중치: 0      -->현재 최소값 : 0
(8 , 9) : 선택한 가중치: 4      -->현재 최소값 : 4
(4 , 8) : 선택한 가중치: 3      -->현재 최소값 : 7
(4 , 5) : 선택한 가중치: 10     -->현재 최소값 : 17
(9 , 10) : 선택한 가중치: 12    -->현재 최소값 : 29
(6 , 10) : 선택한 가중치: 6     -->현재 최소값 : 35
(1 , 4) : 선택한 가중치: 17     -->현재 최소값 : 52
(3 , 4) : 선택한 가중치: 18     -->현재 최소값 : 70
(3 , 7) : 선택한 가중치: 5      -->현재 최소값 : 75
(1 , 2) : 선택한 가중치: 32     -->현재 최소값 : 107

최소값 : 107

C:\Users\#tticj\source\repos\SUB6\Debug\SUB6.exe(프로세스 3448개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

프림 알고리즘으로 찾아낸 최소 신장트리 (MST) 는 어떤 정점에서 시작해도 똑같다.

문제 3번.

```
선택 Microsoft Visual Studio 디버그 콘솔
1. 프림 알고리즘 2. 크루스칼 알고리즘 : 2

-----
크루스칼 알고리즘을 이용한 최소 신장 트리 찾기
간선들의 가중치에 따른 정렬 :
(3 , 7) : 3
(7 , 8) : 4
(2 , 6) : 5
(5 , 9) : 6
(3 , 4) : 10
(8 , 9) : 12
(0 , 3) : 17
(2 , 3) : 18
(4 , 8) : 25
(4 , 5) : 28
(0 , 1) : 32
(1 , 4) : 45
(6 , 7) : 58
(4 , 8) : 선택한 가중치: 3      -->현재 최소값 : 3
(8 , 9) : 선택한 가중치: 4      -->현재 최소값 : 7
(3 , 7) : 선택한 가중치: 5      -->현재 최소값 : 12
(6 , 10) : 선택한 가중치: 6     -->현재 최소값 : 18
(4 , 5) : 선택한 가중치: 10     -->현재 최소값 : 28
(9 , 10) : 선택한 가중치: 12    -->현재 최소값 : 40
(1 , 4) : 선택한 가중치: 17     -->현재 최소값 : 57
(3 , 4) : 선택한 가중치: 18     -->현재 최소값 : 75
(1 , 2) : 선택한 가중치: 32     -->현재 최소값 : 107

최소값 : 107

C:\Users\#tticj\source\repos\SUB6\Debug\SUB6.exe(프로세스 10740개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

* 가중치에 따른 오름차순 정렬한 arr[] 에 대해서도 출력하였다.

4. 디버깅

The screenshot shows the Visual Studio 2019 IDE with the following components:

- Source Code (kruskal.cpp):**

```

57 //크루스칼알고리즘 전 가중치를 기준으로 오름차순으로 정렬을 위한 함수
58 int sort_for_kruskal(int n);
59 void kruskal(int n);
60
61
62 //-----함수 및 전역 변수 및 구조체
63
64 //-----
65
66 int get_min_vertex(int n){ ... }
67 void prim(int s, int n){ ... }
68
69 //-----프림알고리즘 사용 함수 끝
70
71 int compare(const void* a, const void* b){ ... }
72 void init_set(int n){ ... }
73 int find_set(int vertex){
74     int parent_value, p, i = -1;
75     for (i = vertex; (parent_value = parent[i]) >= 0; i = parent_value)
76         ; // 원소의 parent를 찾아가
77     p = i; //parent를 발견했음.
78     for (i = vertex; (parent_value = parent[i]) >= 0; i = parent_value)
79         parent[i] = p; // 원소의 parent
80     return p;
81 }
82 void union_set(int r1, int r2){
83     if (num[r1] < num[r2]){
84         parent[r1] = r2; //r2 = 부모노드가 될것 r1 = 자식
85         num[r2] += num[r1]; // r2의 갯수가 r1의 집합의 갯수만큼 늘어남.
86     }
87     else{
88         parent[r2] = r1;
89         num[r1] += num[r2];
90     }
91 }
92
93 int main(){
94     int n;
95     while (scanf("%d", &n) != EOF){
96         int arr[100][3];
97         int mincost = 0;
98         int index = 0;
99         sort_for_kruskal(n);
100         init_set(n);
101         while (edge_count < (n - 1)){
102             if (i < index){
103                 min = arr[i].weight;
104                 uset = find_set(arr[i].src);
105                 vset = find_set(arr[i].dest);
106                 if (uset != vset){ // 같은 시간 1ms 이하
107                     edge_count++;
108                     union_set(uset, vset);
109                     mincost += min;
110                     printf("선택한 가중치: %d\n", min);
111                     printf("현재 최소값: %d\n", mincost);
112                     i++;
113                 }
114             }
115         }
116         printf("최소값: %d\n", mincost);
117     }
118     return 0;
119 }
120
121 //-----크루스칼 알고리즘 사용 함수 끝
122
123 void main(){
124     int n;
125     while (scanf("%d", &n) != EOF){
126         int arr[100][3];
127         int mincost = 0;
128         int index = 0;
129         sort_for_kruskal(n);
130         init_set(n);
131         while (edge_count < (n - 1)){
132             if (i < index){
133                 min = arr[i].weight;
134                 uset = find_set(arr[i].src);
135                 vset = find_set(arr[i].dest);
136                 if (uset != vset){ // 같은 시간 1ms 이하
137                     edge_count++;
138                     union_set(uset, vset);
139                     mincost += min;
140                     printf("선택한 가중치: %d\n", min);
141                     printf("현재 최소값: %d\n", mincost);
142                     i++;
143                 }
144             }
145         }
146         printf("최소값: %d\n", mincost);
147     }
148     return 0;
149 }
150
151 //-----크루스칼 알고리즘 사용 함수 끝
152
153 void main(){
154     int n;
155     while (scanf("%d", &n) != EOF){
156         int arr[100][3];
157         int mincost = 0;
158         int index = 0;
159         sort_for_kruskal(n);
160         init_set(n);
161         while (edge_count < (n - 1)){
162             if (i < index){
163                 min = arr[i].weight;
164                 uset = find_set(arr[i].src);
165                 vset = find_set(arr[i].dest);
166                 if (uset != vset){ // 같은 시간 1ms 이하
167                     edge_count++;
168                     union_set(uset, vset);
169                     mincost += min;
170                     printf("선택한 가중치: %d\n", min);
171                     printf("현재 최소값: %d\n", mincost);
172                     i++;
173                 }
174             }
175         }
176         printf("최소값: %d\n", mincost);
177     }
178     return 0;
179 }
180
181 //-----크루스칼 알고리즘 사용 함수 끝
182
183 void main(){
184     int n;
185     while (scanf("%d", &n) != EOF){
186         int arr[100][3];
187         int mincost = 0;
188         int index = 0;
189         sort_for_kruskal(n);
190         init_set(n);
191         while (edge_count < (n - 1)){
192             if (i < index){
193                 min = arr[i].weight;
194                 uset = find_set(arr[i].src);
195                 vset = find_set(arr[i].dest);
196                 if (uset != vset){ // 같은 시간 1ms 이하
197                     edge_count++;
198                     union_set(uset, vset);
199                     mincost += min;
200                     printf("선택한 가중치: %d\n", min);
201                     printf("현재 최소값: %d\n", mincost);
202                     i++;
203                 }
204             }
205         }
206         printf("최소값: %d\n", mincost);
207     }
208     return 0;
209 }
210
211 //-----크루스칼 알고리즘 사용 함수 끝
212
213 void main(){
214     int n;
215     while (scanf("%d", &n) != EOF){
216         int arr[100][3];
217         int mincost = 0;
218         int index = 0;
219         sort_for_kruskal(n);
220         init_set(n);
221         while (edge_count < (n - 1)){
222             if (i < index){
223                 min = arr[i].weight;
224                 uset = find_set(arr[i].src);
225                 vset = find_set(arr[i].dest);
226                 if (uset != vset){ // 같은 시간 1ms 이하
227                     edge_count++;
228                     union_set(uset, vset);
229                     mincost += min;
230                     printf("선택한 가중치: %d\n", min);
231                     printf("현재 최소값: %d\n", mincost);
232                     i++;
233                 }
234             }
235         }
236         printf("최소값: %d\n", mincost);
237     }
238     return 0;
239 }
240
241 //-----크루스칼 알고리즘 사용 함수 끝
242
243 void main(){
244     int n;
245     while (scanf("%d", &n) != EOF){
246         int arr[100][3];
247         int mincost = 0;
248         int index = 0;
249         sort_for_kruskal(n);
250         init_set(n);
251         while (edge_count < (n - 1)){
252             if (i < index){
253                 min = arr[i].weight;
254                 uset = find_set(arr[i].src);
255                 vset = find_set(arr[i].dest);
256                 if (uset != vset){ // 같은 시간 1ms 이하
257                     edge_count++;
258                     union_set(uset, vset);
259                     mincost += min;
260                     printf("선택한 가중치: %d\n", min);
261                     printf("현재 최소값: %d\n", mincost);
262                     i++;
263                 }
264             }
265         }
266         printf("최소값: %d\n", mincost);
267     }
268     return 0;
269 }
270
271 //-----크루스칼 알고리즘 사용 함수 끝
272
273 void main(){
274     int n;
275     while (scanf("%d", &n) != EOF){
276         int arr[100][3];
277         int mincost = 0;
278         int index = 0;
279         sort_for_kruskal(n);
280         init_set(n);
281         while (edge_count < (n - 1)){
282             if (i < index){
283                 min = arr[i].weight;
284                 uset = find_set(arr[i].src);
285                 vset = find_set(arr[i].dest);
286                 if (uset != vset){ // 같은 시간 1ms 이하
287                     edge_count++;
288                     union_set(uset, vset);
289                     mincost += min;
290                     printf("선택한 가중치: %d\n", min);
291                     printf("현재 최소값: %d\n", mincost);
292                     i++;
293                 }
294             }
295         }
296         printf("최소값: %d\n", mincost);
297     }
298     return 0;
299 }
300
301 //-----크루스칼 알고리즘 사용 함수 끝
302
303 void main(){
304     int n;
305     while (scanf("%d", &n) != EOF){
306         int arr[100][3];
307         int mincost = 0;
308         int index = 0;
309         sort_for_kruskal(n);
310         init_set(n);
311         while (edge_count < (n - 1)){
312             if (i < index){
313                 min = arr[i].weight;
314                 uset = find_set(arr[i].src);
315                 vset = find_set(arr[i].dest);
316                 if (uset != vset){ // 같은 시간 1ms 이하
317                     edge_count++;
318                     union_set(uset, vset);
319                     mincost += min;
320                     printf("선택한 가중치: %d\n", min);
321                     printf("현재 최소값: %d\n", mincost);
322                     i++;
323                 }
324             }
325         }
326         printf("최소값: %d\n", mincost);
327     }
328     return 0;
329 }
330
331 //-----크루스칼 알고리즘 사용 함수 끝
332
333 void main(){
334     int n;
335     while (scanf("%d", &n) != EOF){
336         int arr[100][3];
337         int mincost = 0;
338         int index = 0;
339         sort_for_kruskal(n);
340         init_set(n);
341         while (edge_count < (n - 1)){
342             if (i < index){
343                 min = arr[i].weight;
344                 uset = find_set(arr[i].src);
345                 vset = find_set(arr[i].dest);
346                 if (uset != vset){ // 같은 시간 1ms 이하
347                     edge_count++;
348                     union_set(uset, vset);
349                     mincost += min;
350                     printf("선택한 가중치: %d\n", min);
351                     printf("현재 최소값: %d\n", mincost);
352                     i++;
353                 }
354             }
355         }
356         printf("최소값: %d\n", mincost);
357     }
358     return 0;
359 }
360
361 //-----크루스칼 알고리즘 사용 함수 끝
362
363 void main(){
364     int n;
365     while (scanf("%d", &n) != EOF){
366         int arr[100][3];
367         int mincost = 0;
368         int index = 0;
369         sort_for_kruskal(n);
370         init_set(n);
371         while (edge_count < (n - 1)){
372             if (i < index){
373                 min = arr[i].weight;
374                 uset = find_set(arr[i].src);
375                 vset = find_set(arr[i].dest);
376                 if (uset != vset){ // 같은 시간 1ms 이하
377                     edge_count++;
378                     union_set(uset, vset);
379                     mincost += min;
380                     printf("선택한 가중치: %d\n", min);
381                     printf("현재 최소값: %d\n", mincost);
382                     i++;
383                 }
384             }
385         }
386         printf("최소값: %d\n", mincost);
387     }
388     return 0;
389 }
390
391 //-----크루스칼 알고리즘 사용 함수 끝
392
393 void main(){
394     int n;
395     while (scanf("%d", &n) != EOF){
396         int arr[100][3];
397         int mincost = 0;
398         int index = 0;
399         sort_for_kruskal(n);
400         init_set(n);
401         while (edge_count < (n - 1)){
402             if (i < index){
403                 min = arr[i].weight;
404                 uset = find_set(arr[i].src);
405                 vset = find_set(arr[i].dest);
406                 if (uset != vset){ // 같은 시간 1ms 이하
407                     edge_count++;
408                     union_set(uset, vset);
409                     mincost += min;
410                     printf("선택한 가중치: %d\n", min);
411                     printf("현재 최소값: %d\n", mincost);
412                     i++;
413                 }
414             }
415         }
416         printf("최소값: %d\n", mincost);
417     }
418     return 0;
419 }
420
421 //-----크루스칼 알고리즘 사용 함수 끝
422
423 void main(){
424     int n;
425     while (scanf("%d", &n) != EOF){
426         int arr[100][3];
427         int mincost = 0;
428         int index = 0;
429         sort_for_kruskal(n);
430         init_set(n);
431         while (edge_count < (n - 1)){
432             if (i < index){
433                 min = arr[i].weight;
434                 uset = find_set(arr[i].src);
435                 vset = find_set(arr[i].dest);
436                 if (uset != vset){ // 같은 시간 1ms 이하
437                     edge_count++;
438                     union_set(uset, vset);
439                     mincost += min;
440                     printf("선택한 가중치: %d\n", min);
441                     printf("현재 최소값: %d\n", mincost);
442                     i++;
443                 }
444             }
445         }
446         printf("최소값: %d\n", mincost);
447     }
448     return 0;
449 }
450
451 //-----크루스칼 알고리즘 사용 함수 끝
452
453 void main(){
454     int n;
455     while (scanf("%d", &n) != EOF){
456         int arr[100][3];
457         int mincost = 0;
458         int index = 0;
459         sort_for_kruskal(n);
460         init_set(n);
461         while (edge_count < (n - 1)){
462             if (i < index){
463                 min = arr[i].weight;
464                 uset = find_set(arr[i].src);
465                 vset = find_set(arr[i].dest);
466                 if (uset != vset){ // 같은 시간 1ms 이하
467                     edge_count++;
468                     union_set(uset, vset);
469                     mincost += min;
470                     printf("선택한 가중치: %d\n", min);
471                     printf("현재 최소값: %d\n", mincost);
472                     i++;
473                 }
474             }
475         }
476         printf("최소값: %d\n", mincost);
477     }
478     return 0;
479 }
480
481 //-----크루스칼 알고리즘 사용 함수 끝
482
483 void main(){
484     int n;
485     while (scanf("%d", &n) != EOF){
486         int arr[100][3];
487         int mincost = 0;
488         int index = 0;
489         sort_for_kruskal(n);
490         init_set(n);
491         while (edge_count < (n - 1)){
492             if (i < index){
493                 min = arr[i].weight;
494                 uset = find_set(arr[i].src);
495                 vset = find_set(arr[i].dest);
496                 if (uset != vset){ // 같은 시간 1ms 이하
497                     edge_count++;
498                     union_set(uset, vset);
499                     mincost += min;
500                     printf("선택한 가중치: %d\n", min);
501                     printf("현재 최소값: %d\n", mincost);
502                     i++;
503                 }
504             }
505         }
506         printf("최소값: %d\n", mincost);
507     }
508     return 0;
509 }
509
510 //-----크루스칼 알고리즘 사용 함수 끝
511
512 void main(){
513     int n;
514     while (scanf("%d", &n) != EOF){
515         int arr[100][3];
516         int mincost = 0;
517         int index = 0;
518         sort_for_kruskal(n);
519         init_set(n);
520         while (edge_count < (n - 1)){
521             if (i < index){
522                 min = arr[i].weight;
523                 uset = find_set(arr[i].src);
524                 vset = find_set(arr[i].dest);
525                 if (uset != vset){ // 같은 시간 1ms 이하
526                     edge_count++;
527                     union_set(uset, vset);
528                     mincost += min;
529                     printf("선택한 가중치: %d\n", min);
530                     printf("현재 최소값: %d\n", mincost);
531                     i++;
532                 }
533             }
534         }
535         printf("최소값: %d\n", mincost);
536     }
537     return 0;
538 }
539
540 //-----크루스칼 알고리즘 사용 함수 끝
541
542 void main(){
543     int n;
544     while (scanf("%d", &n) != EOF){
545         int arr[100][3];
546         int mincost = 0;
547         int index = 0;
548         sort_for_kruskal(n);
549         init_set(n);
550         while (edge_count < (n - 1)){
551             if (i < index){
552                 min = arr[i].weight;
553                 uset = find_set(arr[i].src);
554                 vset = find_set(arr[i].dest);
555                 if (uset != vset){ // 같은 시간 1ms 이하
556                     edge_count++;
557                     union_set(uset, vset);
558                     mincost += min;
559                     printf("선택한 가중치: %d\n", min);
560                     printf("현재 최소값: %d\n", mincost);
561                     i++;
562                 }
563             }
564         }
565         printf("최소값: %d\n", mincost);
566     }
567     return 0;
568 }
569
570 //-----크루스칼 알고리즘 사용 함수 끝
571
572 void main(){
573     int n;
574     while (scanf("%d", &n) != EOF){
575         int arr[100][3];
576         int mincost = 0;
577         int index = 0;
578         sort_for_kruskal(n);
579         init_set(n);
580         while (edge_count < (n - 1)){
581             if (i < index){
582                 min = arr[i].weight;
583                 uset = find_set(arr[i].src);
584                 vset = find_set(arr[i].dest);
585                 if (uset != vset){ // 같은 시간 1ms 이하
586                     edge_count++;
587                     union_set(uset, vset);
588                     mincost += min;
589                     printf("선택한 가중치: %d\n", min);
590                     printf("현재 최소값: %d\n", mincost);
591                     i++;
592                 }
593             }
594         }
595         printf("최소값: %d\n", mincost);
596     }
597     return 0;
598 }
599
600 //-----크루스칼 알고리즘 사용 함수 끝
601
602 void main(){
603     int n;
604     while (scanf("%d", &n) != EOF){
605         int arr[100][3];
606         int mincost = 0;
607         int index = 0;
608         sort_for_kruskal(n);
609         init_set(n);
610         while (edge_count < (n - 1)){
611             if (i < index){
612                 min = arr[i].weight;
613                 uset = find_set(arr[i].src);
614                 vset = find_set(arr[i].dest);
615                 if (uset != vset){ // 같은 시간 1ms 이하
616                     edge_count++;
617                     union_set(uset, vset);
618                     mincost += min;
619                     printf("선택한 가중치: %d\n", min);
620                     printf("현재 최소값: %d\n", mincost);
621                     i++;
622                 }
623             }
624         }
625         printf("최소값: %d\n", mincost);
626     }
627     return 0;
628 }
629
630 //-----크루스칼 알고리즘 사용 함수 끝
631
632 void main(){
633     int n;
634     while (scanf("%d", &n) != EOF){
635         int arr[100][3];
636         int mincost = 0;
637         int index = 0;
638         sort_for_kruskal(n);
639         init_set(n);
640         while (edge_count < (n - 1)){
641             if (i < index){
642                 min = arr[i].weight;
643                 uset = find_set(arr[i].src);
644                 vset = find_set(arr[i].dest);
645                 if (uset != vset){ // 같은 시간 1ms 이하
646                     edge_count++;
647                     union_set(uset, vset);
648                     mincost += min;
649                     printf("선택한 가중치: %d\n", min);
650                     printf("현재 최소값: %d\n", mincost);
651                     i++;
652                 }
653             }
654         }
655         printf("최소값: %d\n", mincost);
656     }
657     return 0;
658 }
659
660 //-----크루스칼 알고리즘 사용 함수 끝
661
662 void main(){
663     int n;
664     while (scanf("%d", &n) != EOF){
665         int arr[100][3];
666         int mincost = 0;
667         int index = 0;
668         sort_for_kruskal(n);
669         init_set(n);
670         while (edge_count < (n - 1)){
671             if (i < index){
672                 min = arr[i].weight;
673                 uset = find_set(arr[i].src);
674                 vset = find_set(arr[i].dest);
675                 if (uset != vset){ // 같은 시간 1ms 이하
676                     edge_count++;
677                     union_set(uset, vset);
678                     mincost += min;
679                     printf("선택한 가중치: %d\n", min);
680                     printf("현재 최소값: %d\n", mincost);
681                     i++;
682                 }
683             }
684         }
685         printf("최소값: %d\n", mincost);
686     }
687     return 0;
688 }
689
690 //-----크루스칼 알고리즘 사용 함수 끝
691
692 void main(){
693     int n;
694     while (scanf("%d", &n) != EOF){
695         int arr[100][3];
696         int mincost = 0;
697         int index = 0;
698         sort_for_kruskal(n);
699         init_set(n);
700         while (edge_count < (n - 1)){
701             if (i < index){
702                 min = arr[i].weight;
703                 uset = find_set(arr[i].src);
704                 vset = find_set(arr[i].dest);
705                 if (uset != vset){ // 같은 시간 1ms 이하
706                     edge_count++;
707                     union_set(uset, vset);
708                     mincost += min;
709                     printf("선택한 가중치: %d\n", min);
710                     printf("현재 최소값: %d\n", mincost);
711                     i++;
712                 }
713             }
714         }
715         printf("최소값: %d\n", mincost);
716     }
717     return 0;
718 }
719
720 //-----크루스칼 알고리즘 사용 함수 끝
721
722 void main(){
723     int n;
724     while (scanf("%d", &n) != EOF){
725         int arr[100][3];
726         int mincost = 0;
727         int index = 0;
728         sort_for_kruskal(n);
729         init_set(n);
730         while (edge_count < (n - 1)){
731             if (i < index){
732                 min = arr[i].weight;
733                 uset = find_set(arr[i].src);
734                 vset = find_set(arr[i].dest);
735                 if (uset != vset){ // 같은 시간 1ms 이하
736                     edge_count++;
737                     union_set(uset, vset);
738                     mincost += min;
739                     printf("선택한 가중치: %d\n", min);
740                     printf("현재 최소값: %d\n", mincost);
741                     i++;
742                 }
743             }
744         }
745         printf("최소값: %d\n", mincost);
746     }
747     return 0;
748 }
749
750 //-----크루스칼 알고리즘 사용 함수 끝
751
752 void main(){
753     int n;
754     while (scanf("%d", &n) != EOF){
755         int arr[100][3];
756         int mincost = 0;
757         int index = 0;
758         sort_for_kruskal(n);
759         init_set(n);
760         while (edge_count < (n - 1)){
761             if (i < index){
762                 min = arr[i].weight;
763                 uset = find_set(arr[i].src);
764                 vset = find_set(arr[i].dest);
765                 if (uset != vset){ // 같은 시간 1ms 이하
766                     edge_count++;
767                     union_set(uset, vset);
768                     mincost += min;
769                     printf("선택한 가중치: %d\n", min);
770                     printf("현재 최소값: %d\n", mincost);
771                     i++;
772                 }
773             }
774         }
775         printf("최소값: %d\n", mincost);
776     }
777     return 0;
778 }
779
780 //-----크루스칼 알고리즘 사용 함수 끝
781
782 void main(){
783     int n;
784     while (scanf("%d", &n) != EOF){
785         int arr[100][3];
786         int mincost = 0;
787         int index = 0;
788         sort_for_kruskal(n);
789         init_set(n);
790         while (edge_count < (n - 1)){
791             if (i < index){
792                 min = arr[i].weight;
793                 uset = find_set(arr[i].src);
794                 vset = find_set(arr[i].dest);
795                 if (uset != vset){ // 같은 시간 1ms 이하
796                     edge_count++;
797                     union_set(uset, vset);
798                     mincost += min;
799                     printf("선택한 가중치: %d\n", min);
800                     printf("현재 최소값: %d\n", mincost);
801                     i++;
802                 }
803             }
804         }
805         printf("최소값: %d\n", mincost);
806     }
807     return 0;
808 }
809
810 //-----크루스칼 알고리즘 사용 함수 끝
811
812 void main(){
813     int n;
814     while (scanf("%d", &n) != EOF){
815         int arr[100][3];
816         int mincost = 0;
817         int index = 0;
818         sort_for_kruskal(n);
819         init_set(n);
820         while (edge_count < (n - 1)){
821             if (i < index){
822                 min = arr[i].weight;
823                 uset = find_set(arr[i].src);
824                 vset = find_set(arr[i].dest);
825                 if (uset != vset){ // 같은 시간 1ms 이하
826                     edge_count++;
827                     union_set(uset, vset);
828                     mincost += min;
829                     printf("선택한 가중치: %d\n", min);
830                     printf("현재 최소값: %d\n", mincost);
831                     i++;
832                 }
833             }
834         }
835         printf("최소값: %d\n", mincost);
836     }
837     return 0;
838 }
839
840 //-----크루스칼 알고리즘 사용 함수 끝
841
842 void main(){
843     int n;
844     while (scanf("%d", &n) != EOF){
845         int arr[100][3];
846         int mincost = 0;
847         int index = 0;
848         sort_for_kruskal(n);
849         init_set(n);
850         while (edge_count < (n - 1)){
851             if (i < index){
852                 min = arr[i].weight;
853                 uset = find_set(arr[i].src);
854                 vset = find_set(arr[i].dest);
855                 if (uset != vset){ // 같은 시간 1ms 이하
856                     edge_count++;
857                     union_set(uset, vset);
858                     mincost += min;
859                     printf("선택한 가중치: %d\n", min);
860                     printf("현재 최소값: %d\n", mincost);
861                     i++;
862                 }
863             }
864         }
865         printf("최소값: %d\n", mincost);
866     }
867     return 0;
868 }
869
870 //-----크루스칼 알고리즘 사용 함수 끝
871
872 void main(){
873     int n;
874     while (scanf("%d", &n) != EOF){
875         int arr[100][3];
876         int mincost = 0;
877         int index = 0;
878         sort_for_kruskal(n);
879         init_set(n);
880         while (edge_count < (n - 1)){
881             if (i < index){
882                 min = arr[i].weight;
883                 uset = find_set(arr[i].src);
884                 vset = find_set(arr[i].dest);
885                 if (uset != vset){ // 같은 시간 1ms 이하
886                     edge_count++;
887                     union_set(uset, vset);
888                     mincost += min;
889                     printf("선택한 가중치: %d\n", min);
890                     printf("현재 최소값: %d\n", mincost);
891                     i++;
892                 }
893             }
894         }
895         printf("최소값: %d\n", mincost);
896     }
897     return 0;
898 }
899
900 //-----크루스칼 알고리즘 사용 함수 끝
901
902 void main(){
903     int n;
904     while (scanf("%d", &n) != EOF){
905         int arr[100][3];
906         int mincost = 0;
907         int index = 0;
908         sort_for_kruskal(n);
909         init_set(n);
910         while (edge_count < (n - 1)){
911             if (i < index){
912                 min = arr[i].weight;
913                 uset = find_set(arr[i].src);
914                 vset = find_set(arr[i].dest);
915                 if (uset != vset){ // 같은 시간 1ms 이하
916                     edge_count++;
917                     union_set(uset, vset);
918                     mincost += min;
919                     printf("선택한 가중치: %d\n", min);
920                     printf("현재 최소값: %d\n", mincost);
921                     i++;
922                 }
923             }
924         }
925         printf("최소값: %d\n", mincost);
926     }
927     return 0;
928 }
929
930 //-----크루스칼 알고리즘 사용 함수 끝
931
932 void main(){
933     int n;
934     while (scanf("%d", &n) != EOF){
935         int arr[100][3];
936         int mincost = 0;
937         int index = 0;
938         sort_for_kruskal(n);
939         init_set(n);
940         while (edge_count < (n - 1)){
941             if (i < index){
942                 min = arr[i].weight;
943                 uset = find_set(arr[i].src);
944                 vset = find_set(arr[i].dest);
945                 if (uset != vset){ // 같은 시간 1ms 이하
946                     edge_count++;
947                     union_set(uset, vset);
948                     mincost += min;
949                     printf("선택한 가중치: %d\n", min);
950                     printf("현재 최소값: %d\n", mincost);
951                     i++;
952                 }
953             }
954         }
955         printf("최소값: %d\n", mincost);
956     }
957     return 0;
958 }
959
960 //-----크루스칼 알고리즘 사용 함수 끝
961
962 void main(){
963     int n;
964     while (scanf("%d", &n) != EOF){
965         int arr[100][3];
966         int mincost = 0;
967         int index = 0;
968         sort_for_kruskal(n);
969         init_set(n);
970         while (edge_count < (n - 1)){
971             if (i < index){
972                 min = arr[i].weight;
973                 uset = find_set(arr[i].src);
974                 vset = find_set(arr[i].dest);
975                 if (uset != vset){ // 같은 시간 1ms 이하
976                     edge_count++;
977                     union_set(uset, vset);
978                     mincost += min;
979                     printf("선택한 가중치: %d\n", min);
980                     printf("현재 최소값: %d\n", mincost);
981                     i++;
982                 }
983             }
984         }
985         printf("최소값: %d\n", mincost);
986     }
987     return 0;
988 }
989
990 //-----크루스칼 알고리즘 사용 함수 끝
991
992 void main(){
993     int n;
994     while (scanf("%d", &n) != EOF){
995         int arr[100][3];
996         int mincost = 0;
997         int index = 0;
998         sort_for_kruskal(n);
999         init_set(n);
1000         while (edge_count < (n - 1)){
1001             if (i < index){
1002                 min = arr[i].weight;
1003                 uset = find_set(arr[i].src);
1004                 vset = find_set(arr[i].dest);
1005                 if (uset != vset){ // 같은 시간 1ms 이하
1006                     edge_count++;
1007                     union_set(uset, vset);
1008                     mincost += min;
1009                     printf("선택한 가중치: %d\n", min);
1010                     printf("현재 최소값: %d\n", mincost);
1011                     i++;
1012                 }
1013             }
1014         }
1015         printf("최소값: %d\n", mincost);
1016     }
1017     return 0;
1018 }
1019
1020 //-----크루스칼 알고리즘 사용 함수 끝
1021
1022 void main(){
1023     int n;
1024     while (scanf("%d", &n) != EOF){
1025         int arr[100][3];
1026         int mincost = 0;
1027         int index = 0;
1028         sort_for_kruskal(n);
1029         init_set(n);
1030         while (edge_count < (n - 1)){
1031             if (i < index){
1032                 min = arr[i].weight;
1033                 uset = find_set(arr[i].src);
1034                 vset = find_set(arr[i].dest);
1035                 if (uset != vset){ // 같은 시간 1ms 이하
1036                     edge_count++;
1037                     union_set(uset, vset);
1038                     mincost += min;
1039                     printf("선택한 가중치: %d\n", min);
1040                     printf("현재 최소값: %d\n", mincost);
1041                     i++;
1042                 }
1043             }
1044         }
1045         printf("최소값: %d\n", mincost);
1046     }
1047     return 0;
1048 }
1049
1050 //-----크루스칼 알고리즘 사용 함수 끝
1051
1052 void main(){
1053     int n;
1054     while (scanf("%d", &n) != EOF){
1055         int arr[100][3];
1056         int mincost = 0;
1057         int index = 0;
1058         sort_for_kruskal(n);
1059         init_set(n);
1060         while (edge_count < (n - 1)){
1061             if (i < index){
1062                 min = arr[i].weight;
1063                 uset = find_set(arr[i].src);
1064                 vset = find_set(arr[i].dest);
1065                 if (uset != vset){ // 같은 시간 1ms 이하
1066                     edge_count++;
1067                     union_set(uset, vset);
1068                     mincost += min;
1069                     printf("선택한 가중치: %d\n", min);
1070                     printf("현재 최소값: %d\n", mincost);
1071                     i++;
1072                 }
1073             }
1074         }
1075         printf("최소값: %d\n", mincost);
1076     }
1077     return 0;
1078 }
1079
1080 //-----크루스칼 알고리즘 사용 함수 끝
1081
1082 void main(){
1083     int n;
1084     while (scanf("%d", &n) != EOF){
1085         int arr[100][3];
1086         int mincost = 0;
1087         int index = 0;
1088         sort_for_kruskal(n);
1089         init_set(n);
1090         while (edge_count < (n - 1)){
1091             if (i < index){
1092                 min = arr[i].weight;
1093                 uset = find_set(arr[i].src);
1094                 vset = find_set(arr[i].dest);
1095                 if (uset != vset){ // 같은 시간 1ms 이하
1096                     edge_count++;
1097                     union_set(uset, vset);
1098                     mincost += min;
1099                     printf("선택한 가중치: %d\n", min);
1100                     printf("현재 최소값: %d\n", mincost);
1101                     i++;
1102                 }
1103             }
1104         }
1105         printf("최소값: %d\n", mincost);
1106     }
1107     return 0;
1108 }
1109
1110 //-----크루스칼 알고리즘 사용 함수 끝
1111
1112 void main(){
1113     int n;
1114     while (scanf("%d", &n) != EOF){
1115         int arr[100][3];
1116         int mincost = 0;
1117         int index = 0;

```


5. 소스 코드

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0
#define MAX_VER 100 //여기서는 모든 간선을
정렬전에 넣어야함으로 간선의 최대 수보다 크게
정의한다
#define MAX_VERTICES_PRIM 10 // 신장트리
의 정점 개수
#define INF 1000L // 경로가 없는 INFINITE를
표현하기 위해 정의된 상수 INF
int n;
weight[MAX_VERTICES_PRIM][MAX_VERTICES_
PRIM] = { // 신장트리의 거리와 모양을 배열로
표현, INF는 바로갈 수 있는 경로가 없음을 뜻한
다.
    {0, 32, INF, 17, INF, INF, INF, INF, INF, INF},
    {32, 0, INF, INF, 45, INF, INF, INF, INF, INF},
    {INF, INF, 0, 18, INF, INF, 5, INF, INF, INF},
    {17, INF, 18, 0, 10, INF, INF, 3, INF, INF},
    {INF, 45, INF, 10, 0, 28, INF, INF, 25, INF},
    {INF, INF, INF, INF, 28, 0, INF, INF, INF, 6},
    {INF, INF, 5, INF, INF, INF, 0, 59, INF, INF},
    {INF, INF, INF, 3, INF, INF, 59, 0, 4, INF},
    {INF, INF, INF, INF, 25, INF, INF, 4, 0, 12},
    {INF, INF, INF, INF, INF, 6, INF, INF, 12, 0}
};
int n;
temp_weight[MAX_VERTICES_PRIM][MAX_VERTI
CES_PRIM];
int selected[MAX_VERTICES_PRIM]; // 선택된
정점의 정보를 담을 배열, 선택이 됐는지 안됐는
지를 표시한다.
int dist[MAX_VERTICES_PRIM]; // 최소의 거리
정보만을 담는 배열, 새로운 최소거리가 나올 때
마다 갱신된다.
int parent[MAX_VER]; // 각 정점 부모의 노드가
저장된다, 음수 이면 그 원소가 최상위 부모라는
뜻이고 음수와 함께 표시된 숫자가 자식의 수다.
int num[MAX_VER]; // 각 정점들의 집합의 개수
를 저장한다.
typedef struct {
    int src; //시작정점 번호
    int dest; //끝 정점 번호
    int weight; //가중치
}forSort;
forSort arr[MAX_VER] = {0}; // 간선 별로 시작
정점과 끝 정점, 가중치를 저장하기 위한 구조체
배열
// 간선들을 구조체 내에 저장하기 위한 함수
int make_arr(int n);
// 최소 dist[v]값을 갖는 정점을 반환
int get_min_vertex(int n);
// Prim, s는 시작 정점, n 은 정점의 수
```

```
void prim(int s, int n);
// 오름차순 qsort 에 쓰이는 비교함수
int compare(const void * a, const void * b);
// 초기화 용 함수
void init_set(int n);
//원소의 루트 노드를 찾아가서 사이클이 형성되
는지 확인하기위한 함수
int find_set(int vertex);
//사이클이 형성되지 않는 두개의 트리집합을 연
결, 단 숫자가 더 작은 쪽이 큰 쪽의 자식으로 들
어가는 것이 좋다.
void union_set(int r1, int r2);
//가중치를 가지고 해당 가중치를 가지는 정점 2
개를 찾아서 프린트하는 함수
void print_set(int);
//크루스칼알고리즘 전 가중치를 기준으로 오름차
순으로 정렬을 위한 함수
int sort_for_krusal(int n);
void kruskal(int n);
/
-----함수 및 전역 변수
및 구조체 선언 끝
//-----
int get_min_vertex(int n)
{
    int v, i; // 정점의 정보를 저장할 변수 v, 반
복문을 위한 변수 i
    for (i = 0; i < n; i++)
    {
        if (selected[i] == FALSE) {
            v = i; // 아직 선택되지 않은 정점
의 번호를 v에 저장, 각 함수 실행별 0부터 n -
1까지 차례대로 저장된다.
            break;
        }
    }
    // 위에서 선택된 정점이 과연 최소거리를 지
니고 있는 정점인지를 확인한다.
    for (i = 0; i < n; i++)
    {
        // 선택되지 않은 정점들을 순회하면서
최소거리를 가진 정점을 찾아낸다.
        if (selected[i] == FALSE && (dist[i] <
dist[v]))
            v = i; // 더 적은 거리가 존재한다
면 해당 정점을 저장한다.
    }
    return(v); // 최소의 거리를 갖는 정점이 선
택됐으므로 정점 번호를 리턴한다.
}
void prim(int s, int n){
    make_arr(n);
    int i, u, v;
    for (u = 0; u < n; u++){ // dist배열과
selected배열의 정보를 초기화
        dist[u] = INF;
```



```

arr[index].weight);
        index++;
    }
}
return index;
}
int sort_for_krusal(int n) {
    int index = make_arr(n);
    qsort(arr, index, sizeof(forSort), compare);
    //-->arr배열에 저장된 가중치가 오름차순으로 정
    리
    printf("간선들의 가중치에 따른 정렬 :Wn");
    for (int i =0; i < index; i ++) { // -->
        qsort확인
        if (arr[i].weight == INF)
            break;
        printf("(%d , %d) : %dWn", arr[i].src,
arr[i].dest, arr[i].weight);
    }
    return index;
}
void kruskal(int n) {
    int i, j;
    //int index = 0;
    int edge_count =0;
    int min, mincost =0;
    int uset, vset; //정점 u와 정점 v의 집합 번
    호
    int index = sort_for_krusal(n);
    init_set(n);
    i =0;
    while (edge_count < (n -1)) {
        if (i < index) {
            min = arr[i].weight;
            uset = find_set(arr[i].src);
            vset = find_set(arr[i].dest);
            if (uset != vset) {
                edge_count++;
                union_set(uset, vset);
                mincost += min;
                print_set(min);
                printf("선택한          가중치:
%dWt-->", min);
                printf("현재 최소값 : %dWn",
mincost);
            }
            i++;
        }
    }
    printf("Wn최소값 : %d Wn", mincost);
}
//-----
-----크루스칼 알고리즘
사용 함수 끝
void main()
{

```

```

int input;
printf("1. 프림 알고리즘 2. 크루스칼 알고리
즘 : "); scanf("%d", &input);

printf("Wn-----
-----Wn");

switch (input)
{
    case 1:
        printf("프림 알고리즘을 이용한 최소 신
장 트리 찾기Wn");
        //prim(0, 10); // 정점 개수가 7개인 그
래프에서 0번 정점을 출발하여 얻을 수 있는 최
소비용신장트리를 찾아라.
        prim(8,10);
        break;
    case 2 :
        printf("크루스칼 알고리즘을 이용한 최소
신장 트리 찾기Wn");
        kruskal(10);
        break;
    default: printf("1 혹은 2 의 숫자만 입력해주
세요 Wn");
        break;
}

}
/*****
****
** End Line
*****/

```