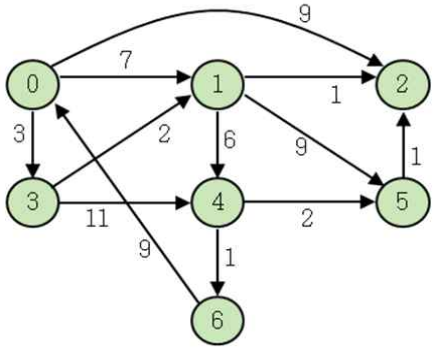
자료구조 14주차 과제

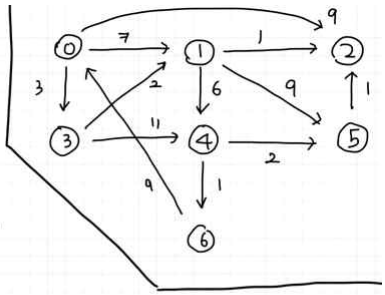
제출일: 2022-12-08

학번/이름: 2016121150 / 윤준영

문1) 위 그래프에 대하여 노드 0에서 다른 노드로의 대한 shortest path의 길이를 모두 구하려 한다. 이를 위하여 교재의 Dijkstra 알고리즘을 사용할 때 distance 배열의 원소들의 값의 변화를 순서대로 모두 쓰시오. (3점)



vertex	0	1	2	3	4	5	6	
distance	0	7	9	(3)	MAX	MAX	MAX	V_0, V_3 : 3
found	1	0	0	0	0	0	0	
distance	0	(5)	9	3	14	MAX	MAX	V_0, V_3, V_1 : 5
found	1	0	0	1	0	0	0	
distance	0	5	(6)	3	11	14	MAX	V_0, V_3, V_1, V_2 : 6
found	1	1	0	1	0	0	0	
distance	0	5	6	3	(11)	14	MAX	V_0, V_3, V_1, V_4 : 11
found	1	1	1	1	0	0	0	
distance	0	5	6	3	11	13	(12)	V_0, V_3, V_1, V_4, V_6 : 12
found	1	1	1	1	1	0	0	
distance	0	5	6	3	11	(13)	12	V_0, V_3, V_1, V_4, V_5 : 13
found	1	1	1	1	1	0	1	
distance	0	5	6	3	11	13	12	
found	1	1	1	1	1	1	1	



문2) 다음 함수를 사용하여 문1)의 모든 쌍의 두 노드 사이의 거리를 모두 출력하려고 한다.

```
void allcosts(int cost[][MAX_VERTICES],
              int distance[][MAX_VERTICES], int n){
/* 각 정점에서 다른 모든 정점으로의 거리 계산,cost:인접 행렬, distance:거리값*/
    int i, j, k;
    for (i=0; i<n; i++)
       for (j=0; j<n; j++)
          distance[i][j] = cost[i][j];
    for (k=0; k<n; k++)
       for (i=0; i<n; i++)
          for (j=0; j<n; j++)
             if (distance[i][k]+distance[k][j] < distance[i][j]) {
                distance[i][j] = distance[i][k] + distance[k][j];
             }
}
```

1) 경로가 있을 경우에 서로 다른 두 노드 사이의 거리를 모두 출력하는 main 함수를 작성하시오.(2점)

```
23  int main(void) {
24      int cost[7][7] = {{0, 7, 9, 3, MAX, MAX, MAX},
25                        {MAX, 0, 1, MAX, 6, 9, MAX},
26                        {MAX, MAX, 0, MAX, MAX, MAX, MAX},
27                        {MAX, 2, MAX, 0, 11, MAX, MAX},
28                        {MAX, MAX, MAX, MAX, 0, 2, 1},
29                        {MAX, MAX, 1, MAX, MAX, 0, MAX},
30                        {9, MAX, MAX, MAX, MAX, MAX, 0}};
31      int dist[7][7];
32      int n = 7;
33      for (int i=0; i<n; i++)
34          for (int j=0; j<n; j++)
35              if (i!=j && cost[i][j]!=MAX)
36                  printf("%d->%d : %d\n", i, j, cost[i][j]);
37      printf("\nallcosts\n");
38      allcosts(cost, dist, n);
39      printf("\nallcosts\n");
40      for (int i=0; i<n; i++)
41          for (int j=0; j<n; j++)
42              if (i!=j && dist[i][j]!=MAX)
43                  printf("%d->%d : %d\n", i, j, dist[i][j]);
44      return 0;
45  }
```

cost allcosts행렬과 main 함수를 위와 같이 작성하였다. MAX값은 INT_MAX로 설정하였다. 따라서 두 노드 사이의 경로가 없거나 출발과 도착이 같으면 제외하고 출력하였다. 출력 결과는 다음과 같다.

```
0->1 : 7
0->2 : 9
0->3 : 3
1->2 : 1
1->4 : 6
1->5 : 9
3->1 : 2
3->4 : 11
4->5 : 2
4->6 : 1
5->2 : 1
6->0 : 9
```

2) 위 함수의 if 문에 다음 문장을 추가하여, distance가 갱신될 때마다 내용이 출력되도록 하고,

0에서 5까지의 최단거리가 계산되는 과정을 설명하시오. (2점)

printf("%d-%d-%d:%d\n",i,k,j,distance[i][j]);

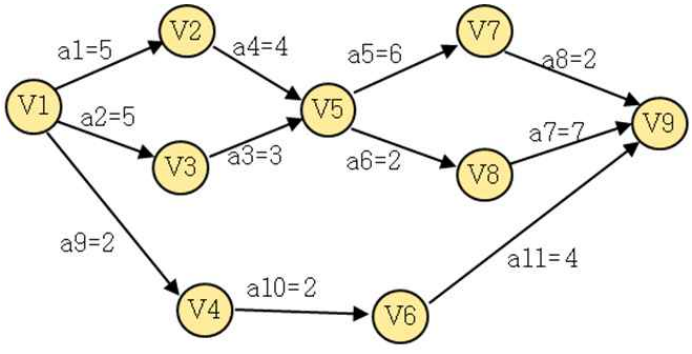
```
6   void allcosts(int cost[][MAX_VERTICES],
7                 int distance[][MAX_VERTICES], int n){
8   /* 각 정점에서 다른 모든 정점으로의 거리 계산,cost:인접 행렬, distance:거리값*/
9       int i, j, k;
10      for (i=0; i<n; i++)
11          for (j=0; j<n; j++)
12              distance[i][j] = cost[i][j];
13      for (k=0; k<n; k++)
14          for (i=0; i<n; i++)
15              for (j=0; j<n; j++)
16                  if (distance[i][k]+distance[k][j] < distance[i][j])
17                      if (distance[i][k] != MAX && distance[k][j] != MAX){
18                          distance[i][j] = distance[i][k] + distance[k][j];
19                          printf("%d-%d-%d:%d\n",i,k,j,distance[i][j]);
20                      }
21  }
```

위와 같이 allcosts 함수를 수정하였다. 직접 연결되어 있지 않을 경우 MAX 값을 INT_MAX로 설정하였는데, 더하는 과정에서 overflow가 발생하기 때문에 distance[i][k]와 distance[k][j] 둘 중 하나가 MAX일 경우 연결될 수 없으므로 distance[i][j]가 갱신되지 않게 설정하였다. 실행 결과는 다음과 같다.

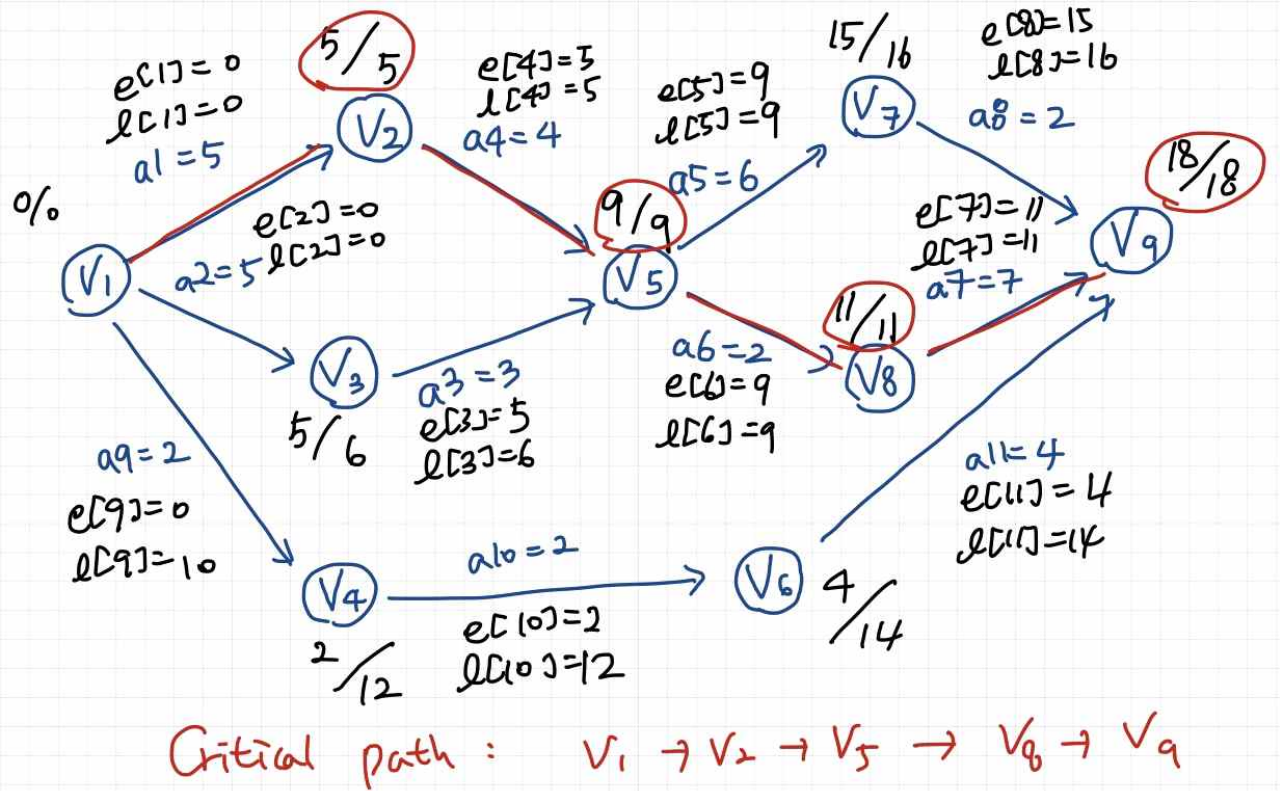
```
allcosts    6-1-5:25
6-0-1:16    0-3-1:5     1-4-6:7
6-0-2:18    0-3-2:6     3-4-5:10
6-0-3:12    0-3-4:11    3-4-6:9
0-1-2:8     0-3-5:14    6-4-5:22
0-1-4:13    6-3-1:14    4-5-2:3
0-1-5:16    6-3-2:15    1-6-0:16
3-1-2:3     6-3-4:20    1-6-3:19
3-1-4:8     6-3-5:23    3-6-0:18
3-1-5:11    0-4-5:13    4-6-0:10
6-1-2:17    0-4-6:12    4-6-1:15
6-1-4:22    1-4-5:8     4-6-3:13
```

0에서 5까지의 최단거리가 계산되는 과정은 MAX에서 0-1-5를 통해 이동하는 cost 7+9=16으로 갱신된 후, 3-5의 최단거리(3-1-5)가 11로 갱신된 후, 0-3-5를 통해 이동하는 cost 3+11=14로 갱신된다. 다음에 0 -4-5를 통해 이동하는 cost 11+2=13으로 최종적으로 갱신된다. 여기서 0-4의 경로는 0-3-4의 루트를 통해 갱신된 최소 cost이다.

문3) 다음 AOE(Activity on Edge) 그래프에 대하여 물음에 답하시오.



1) 위 그래프의 critical path를 쓰시오.(2점)



2) 전체 완료 시간을 지연시키지 않는 노드 V4의 latest event occurrence time을 구하시오.(1점)

$$le[V_4] = l[a_9] + \text{duration of } [a_9]$$

$$= 12$$