Data Structure #13

Shortest Path

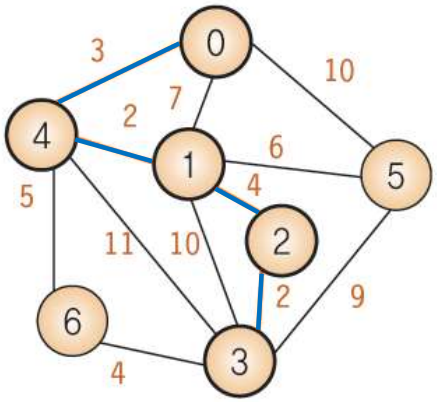
2020년도 1학기

Intro.

- **실습주제 소개**
  - Dijkstra 최단경로 알고리즘
- **실습수업 문제**
  - Dijkstra 최단경로 알고리즘
- **기말 시험**
  - 6. 19 (금) 10:00 – 12:00 IT/BT 202호

최단 경로(shortest path)

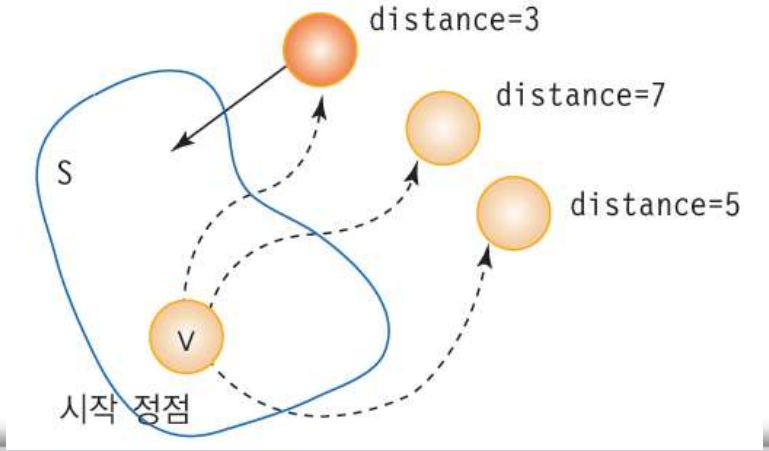
- 네트워크에서 정점 u와 정점 v를 연결하는 경로 중에서 간선들의 가중치 합이 최소가 되는 경로
- 간선의 가중치는 비용, 거리, 시간 등
- 정점 0에서 정점 3으로 가는 최단 경로 문제
  - 인접행렬에서 간선이 없는 노드쌍의 가중치는 ∞ 임
  - 0,4,1,2,3이 최단 경로
  - 최단경로 길이는 3+2+4+2=11



	0	1	2	3	4	5	6
0	0	7	∞	∞	3	10	∞
1	7	0	4	10	2	6	∞
2	∞	4	0	2	∞	∞	∞
3	∞	10	2	0	11	9	4
4	3	2	∞	11	0	∞	5
5	10	6	∞	9	∞	0	∞
6	∞	∞	∞	4	5	∞	0

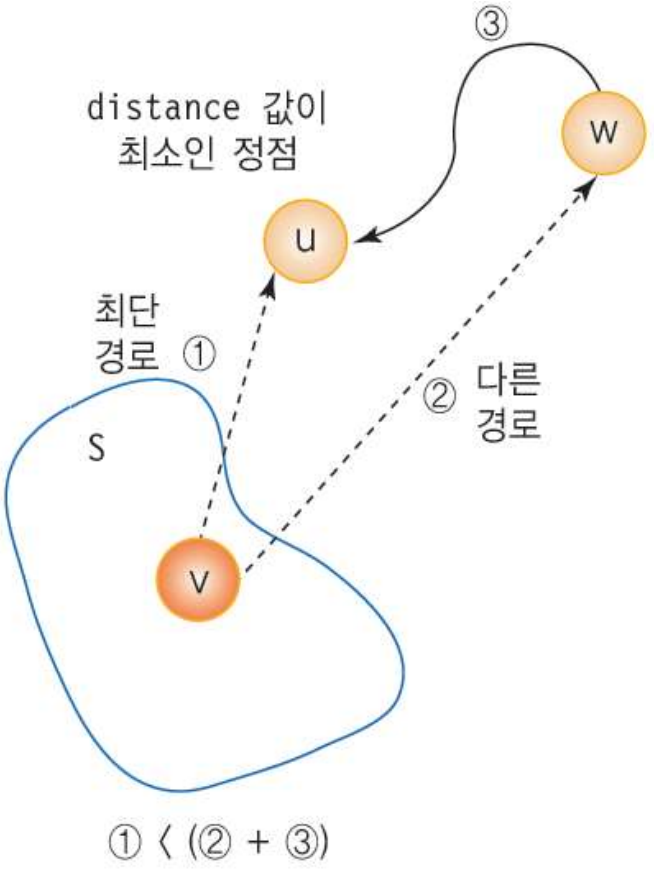
Dijkstra의 최단경로 알고리즘

- **하나의 시작 정점으로부터 모든 다른 정점까지의 최단 경로 찾음**
- **집합 S**
  - 시작 정점 v로부터의 최단경로가 이미 발견된 정점들의 집합
- **distance 배열**
  - 최단경로가 알려진 정점들만을 이용한 다른 정점들까지의 최단경로 길이
  - distance 배열의 초기값(시작 정점 v)
    - distance[v] = 0
    - 다른 정점에 대한 distance 값은 시작정점과 해당 정점간의 가중치 값
- **매 단계에서 가장 distance 값이 작은 정점을 S에 추가**



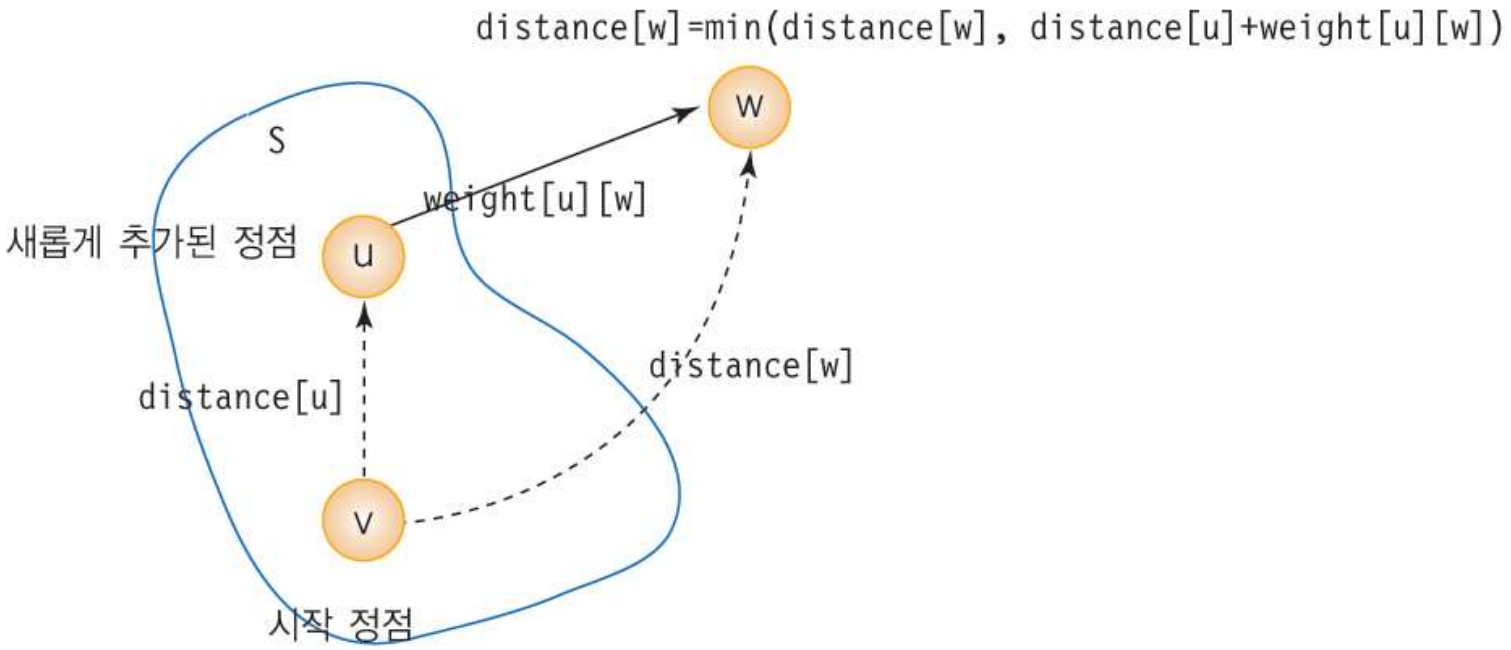
Dijkstra의 최단경로 알고리즘

- distance 값이 가장 작은 정점을 u라고 하자. 그러면 시작 정점 v에서 정점 u까지의 최단거리는 경로 ①이 된다.
- 정점 w를 거쳐서 정점 u로 가는 가상적인 경로가 있다고 가정해보자. 그러면 정점 v에서 정점 u까지의 거리는 정점 v에서 정점 w까지의 거리 ②와 정점 w에서 정점 u로 가는 거리③을 합한 값이 된다.
- 그러나 경로 ②는 경로 ①보다 항상 길 수 밖에 없다. 왜냐하면 현재 distance 값이 가장 작은 정점은 u이기 때문이다.
- 따라서 매 단계에서 distance 값이 가장 작은 정점들을 추가해나가면 시작 정점에서 모든 정점까지의 최단거리를 구할 수 있다.

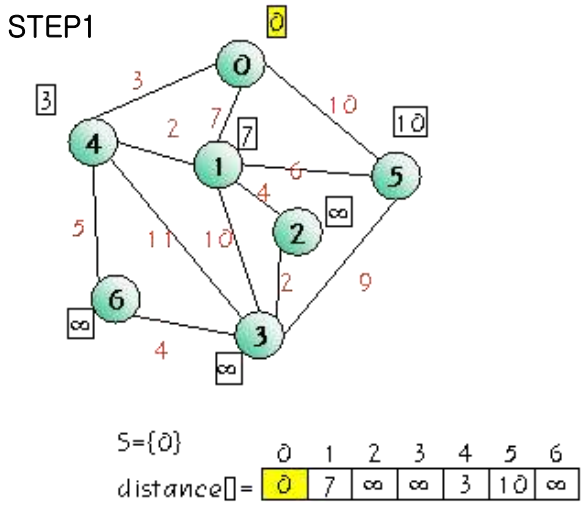


Dijkstra의 최단경로 알고리즘

- 새로운 정점이 S에 추가되면 distance값 갱신



Dijkstra의 최단경로 알고리즘 예시

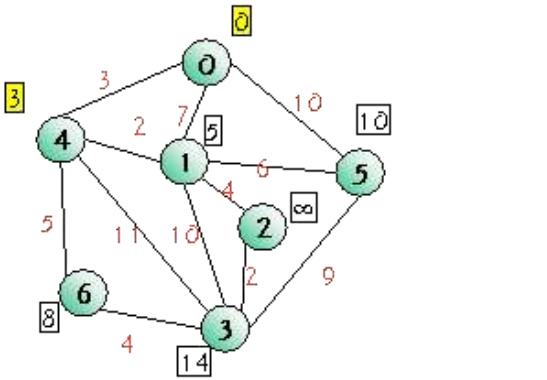


- S={ 0 }

- distance[0] = weight[0][0] = 0
- distance[1] = weight[0][1] = 7
- distance[2] = weight[0][2] = ∞
- distance[3] = weight[0][3] = ∞
- distance[4] = weight[0][4] = 3
- distance[5] = weight[0][5] = 10
- distance[6] = weight[0][6] = ∞

Dijkstra의 최단경로 알고리즘 예시

STEP2



S={0, 4}

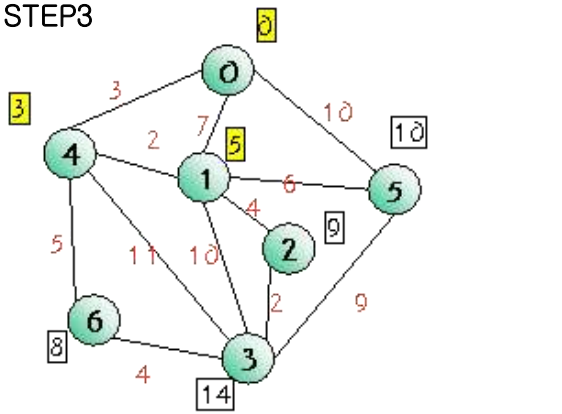
	0	1	2	3	4	5	6
distance[]=	0	5	∞	14	3	10	8

- S={ 0, 4 }
- distance[0] = 0
- distance[1] = min(distance[1], distance[4]+weight[4][1])
  = min(7, 3+2) = 5
- distance[2] = min(distance[2], distance[4]+weight[4][2])
  = ∞
- distance[3] = min(distance[3], distance[4]+weight[4][3])
  = min(∞, 3+11) = 14
- distance[4] = 3
- distance[5] = min(distance[5], distance[4]+weight[4][5])
  = min(10, 3+ ∞) = 10
- distance[6] = min(distance[6], distance[4]+weight[4][6])
  = min(∞, 3+5) = 8

Dijkstra의 최단경로 알고리즘 예시

- S={ 0, 4, 1 }

STEP3

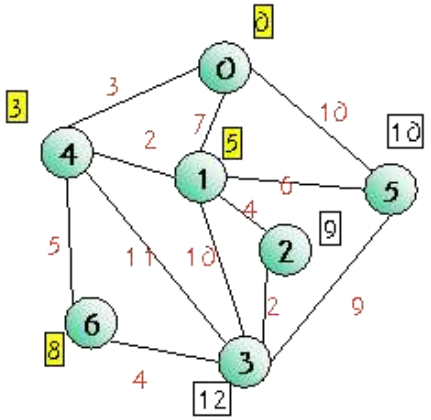


S={∂, 4, 1}

	∂	1	2	3	4	5	6
distance[]=	∂	5	9	14	3	10	8

- S={ 0, 4, 1, 6 }

STEP4



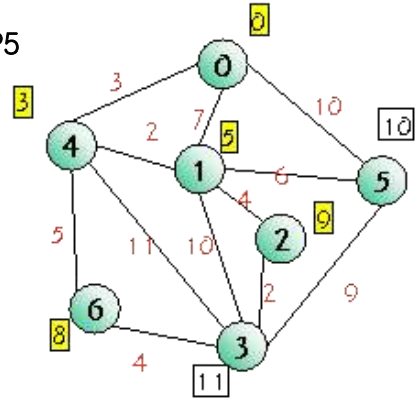
S={∂, 4, 1, 6}

	∂	1	2	3	4	5	6
distance[]=	∂	5	9	12	3	10	8

Dijkstra의 최단경로 알고리즘 예시

- S={ 0, 4, 1, 6, 2 }

STEP5

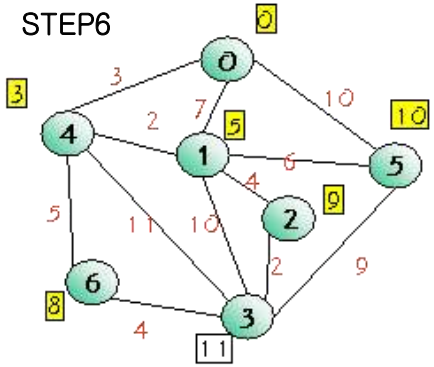


S={0, 4, 1, 6, 2}

	0	1	2	3	4	5	6
distance[]=	0	5	9	11	3	10	8

- S={ 0, 4, 1, 6, 2, 5 }

STEP6



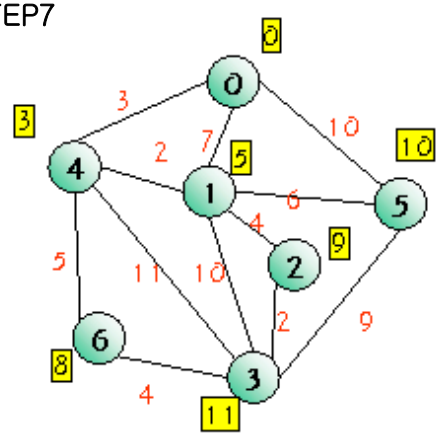
S={0, 4, 1, 6, 2, 5}

	0	1	2	3	4	5	6
distance[]=	0	5	9	11	3	10	8

Dijkstra의 최단경로 알고리즘 예시

- S={ 0, 4, 1, 6, 2, 5, 3 }

STEP7



S={0, 4, 1, 6, 2, 5, 3}

distance[]=

0	1	2	3	4	5	6
0	5	9	11	3	10	8

Dijkstra의 최단경로 알고리즘

```
// 입력: 가중치 그래프 G, 가중치는 음수가 아님.
// 출력: distance 배열, distance[u]는 v에서 u까지의 최단 거리이다.
shortest_path(G, v)

S←{v}
for 각 정점 w∈G do
        distance[w]←weight[v][w];
while 모든 정점이 S에 포함되지 않으면 do
        u←집합 S에 속하지 않는 정점 중에서 최소 distance 정점;
        S←S∪{u}
        for u에 인접하고 S에 없는 각 정점 z do
                if distance[u]+weight[u][z] < distance[z]
                        then distance[z]←distance[u]+weight[u][z];
```

Dijkstra의 최단경로 프로그램 구현 (개인 실습 #1)

– 사용 변수

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

#define TRUE 1
#define FALSE 0
#define NODES                7              /* 노드의 수 */
#define INF    9999                         /* 무한 값(연결이 없는 경우) */

/* 네트워크의 인접행렬 */
int cost[NODES][NODES]={
                { 0,   7, INF, INF,   3,   10, INF },
                { 7,   0,    4,  10,    2,    6, INF },
                { INF,4,     0,    2, INF, INF, INF },
                { INF, 10, 2,      0,  11,    9,    4 },
                { 3,      2, INF, 11,   0,  INF,    5 },
                { 10,    6, INF,  9, INF,      0, INF },
                { INF, INF, INF, 4,    5, INF,     0 }};

int distance[NODES];          /* 시작노드로부터의 최단경로 거리 */
int found[NODES];             /* 방문한 노드 표시 */
```

Dijkstra의 최단경로 프로그램 구현

- 출력함수

```
void print_distance()
{
        int i;
        for(i=0;i<NODES;i++){
                printf("%d ", distance[i]);
        }
        printf("₩n");
}
```

Dijkstra의 최단경로 프로그램 구현

– 최소비용 선택

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

Dijkstra의 최단경로 프로그램 구현

– 최단경로 탐색 함수

```
void shortest_path(int start, int cost[][NODES],
                           int distance[], int n, int found[]) /* 시작노드 start  */
{
    int i, u, w;
  /* 초기화 */
/* 시작노드 방문 표시 */

}
```

Dijkstra의 최단경로 프로그램 구현

- 메인함수

```
void main()
{
        shortest_path(0, cost, distance, NODES, found);
}
```

제출

- **제출**
  - 개인 실습 (#1)
    - 오늘 자정까지 제출 (~ 2020/6/12 23:59)
    - dijkstra