

Data Structure

자료구조



최단 경로 문제



한국기술교육대학교
온라인평생교육원

학습내용

- 최단 경로 문제의 개념
- Dijkstra의 최단 경로 알고리즘
- Floyd의 최단 경로 알고리즘

학습목표

- 최단 경로 문제를 설명할 수 있다.
- Dijkstra의 최단 경로 알고리즘을 설명할 수 있다.
- Floyd의 최단 경로 알고리즘을 설명할 수 있다.

최단 경로 문제의 개념



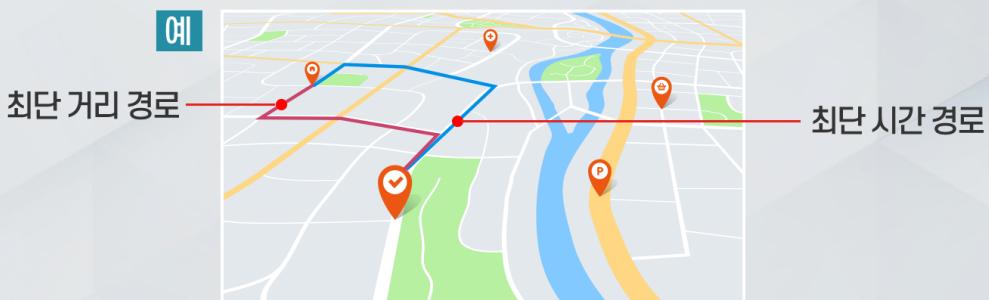
최단 경로 문제

최단 경로 문제의 개념

1 최단 경로 문제의 정의

최단 경로(Shortest Path) 문제

- 가중치 그래프에서 두 정점을 연결하는 여러 경로들 중에서 **간선들의 가중치 합이 최소**가 되는 경로를 찾는 문제
- “최단시간 경로”, “최단거리 경로” 등



최단 경로 문제

최단 경로 문제의 개념

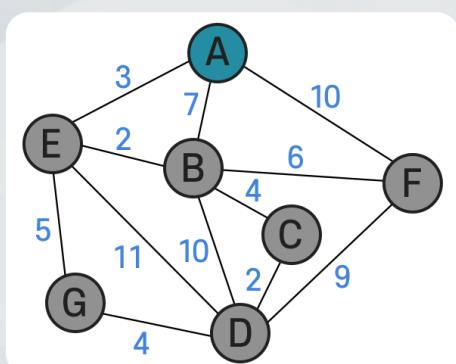
2 최단 경로 문제의 종류

문제 1

시작 정점에서 **도착 정점**까지의 최단 경로 길이

문제 2

시작 정점에서 **모든 정점**까지의 최단 경로 길이



Dijkstra 알고리즘

- (A→A) 최단 경로 A = 0
- (A→B) 최단 경로 A-E-B = 5
- (A→C) 최단 경로 A-E-B-C = 9
- (A→D) 최단 경로 A-E-B-C-D = 11
- (A→E) 최단 경로 A-E = 3
- (A→F) 최단 경로 A-F = 10
- (A→G) 최단 경로 A-E-G = 8

결과 : [0, 5, 9, 11, 3, 10, 8]



최단 경로 문제의 개념

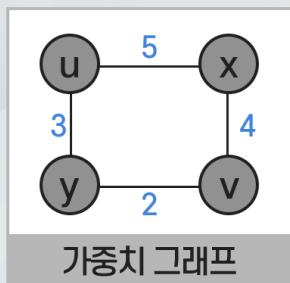
최단 경로 문제

최단 경로 문제의 개념

2 최단 경로 문제의 종류

문제 3

모든 정점 간의 최단 경로 길이



```
V = [ 'u', 'v', 'x', 'y' ]
W = [ [ 0, INF, 5, 3], #u
      [ INF, 0, 4, 2], #v
      [ 5, 4, 0, INF], #x
      [ 3, 2, INF, 0 ] ] #y
```

인접 행렬 표현

```
D = [ [ 0, 5, 5, 3],
      [ 5, 0, 4, 2],
      [ 5, 4, 0, 6],
      [ 3, 2, 6, 0 ] ]
```

계산 결과

모든 정점에서 모든 정점까지의 최단 경로 거리를 구하는 문제와 결과 예

» Floyd 알고리즘

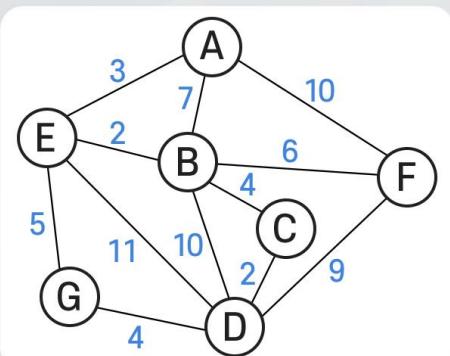
최단 경로 문제

최단 경로 문제의 개념

2 최단 경로 문제의 종류

● 가중치 그래프의 표현(인접 행렬)

» MST 문제 등



```
vertex = [ 'A', 'B', 'C', 'D', 'E', 'F', 'G' ]
weight = [[ None, 7, None, None, 3, 10, None ],
          [ 7, None, 4, 10, 2, 6, None ],
          [ None, 4, None, 2, None, None, None ],
          [ None, 10, 2, None, 11, 9, 4 ],
          [ 3, 2, None, 11, None, 13, 5 ],
          [ 10, 6, None, 9, 13, None, None ],
          [ None, None, None, 4, 5, None, None ] ]
```

가중치 그래프



최단 경로 문제의 개념

최단 경로 문제

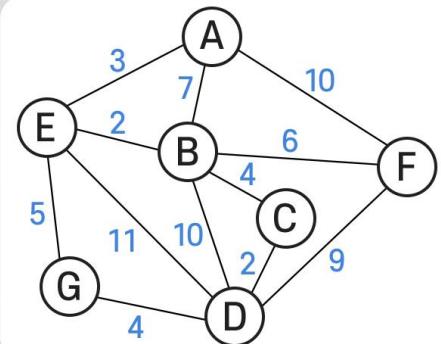
최단 경로 문제의 개념

2 최단 경로 문제의 종류

- 가중치 그래프의 표현(인접 행렬)

▶ 최단 경로 문제

간선이 없으면 가중치를 무한대로 처리



가중치 그래프

```
vertex = [ 'A', 'B', 'C', 'D', 'E', 'F', 'G' ]
weight = [[ 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, 13,   5 ],
          [ 10,   6, INF,  9,  13,   0, INF ],
          [ INF, INF, INF,  4,   5, INF,  0 ]]
```

Dijkstra의 최단 경로 알고리즘



최단 경로 문제

Dijkstra의 최단 경로 알고리즘

1 Dijkstra의 전략

◎ Dijkstra의 알고리즘의 개념

Dijkstra의 알고리즘

- 시작 정점에서 다른 모든 정점까지의 최단 경로 거리 계산하는 알고리즘
- Dijkstra의 전략
 - 탐욕적 기법 사용
 - 최단거리가 확정된 정점들과 간선으로 직접 연결된 정점들 중에서 가장 가까운 정점 u 를 선택
 - u 까지의 거리 확정
 - u 를 제외한 남은 정점들의 거리를 갱신

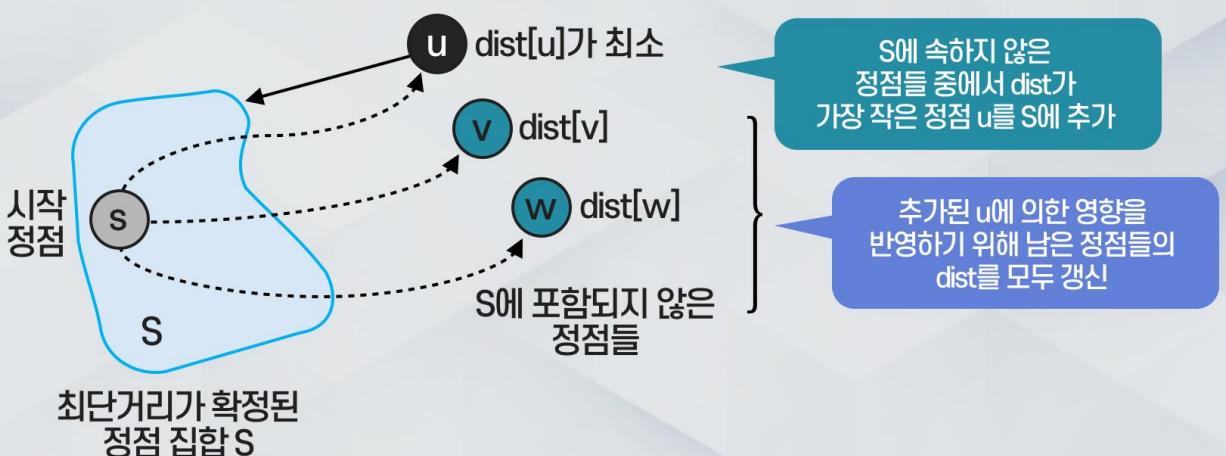
최단 경로 문제

Dijkstra의 최단 경로 알고리즘

1 Dijkstra의 전략

◎ Dijkstra의 알고리즘의 개념

▶ $\text{dist}[v]$: S 내의 정점만을 거쳐서 다른 정점 v 로 가는 최단 거리



Dijkstra의 최단 경로 알고리즘



최단 경로 문제

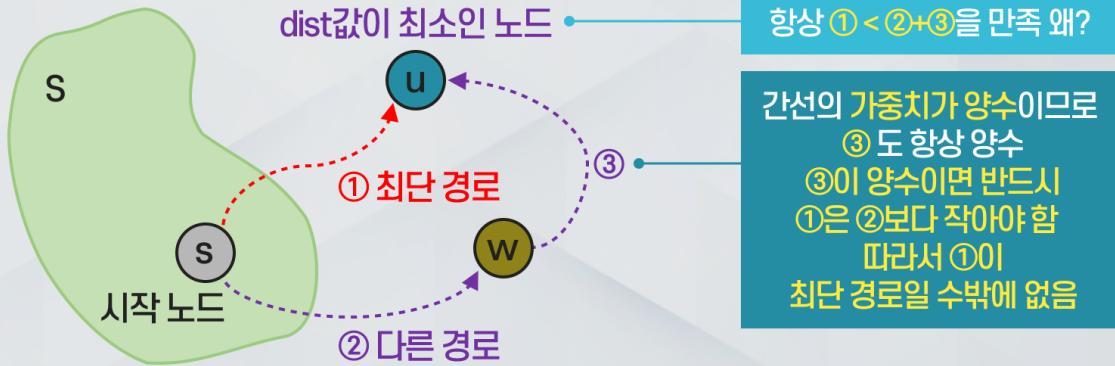
Dijkstra의 최단 경로 알고리즘

1 Dijkstra의 전략

● 최단 경로 증명

▶ $\text{dist}[u]$ 가 최소라면 $s \rightarrow u$ 의 거리를 확정할 수 있을까?

- $s \rightarrow \dots \rightarrow w \rightarrow u$ 가 최소일 수는 없을까?
- 간선의 가중치가 양수라면 → 성립



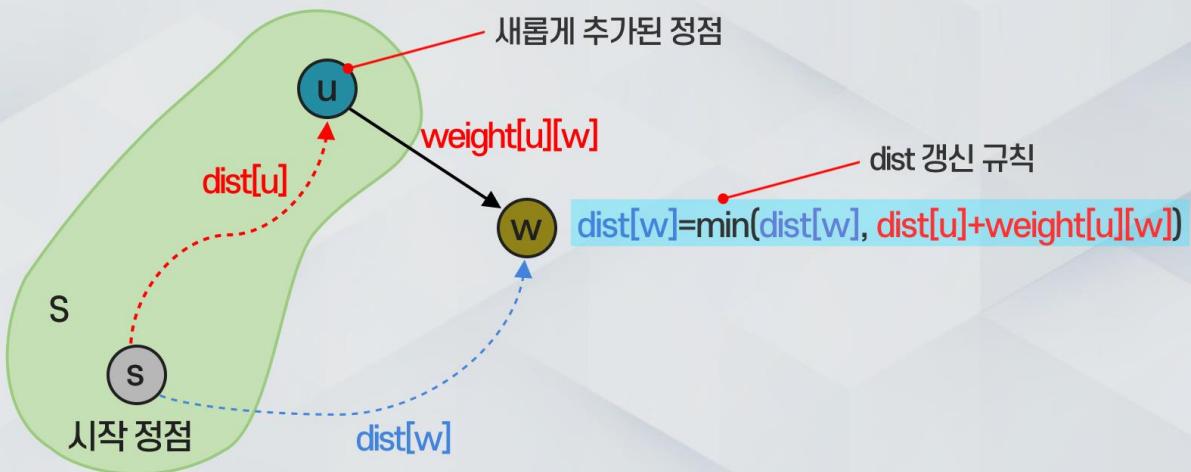
최단 경로 문제

Dijkstra의 최단 경로 알고리즘

1 Dijkstra의 전략

● dist 갱신

▶ 새로운 정점이 S에 추가되면 dist 갱신





Dijkstra의 최단 경로 알고리즘

최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

● 필요한 자료

dist[]

시작 정점으로부터의 거리

- 최초: 시작 정점과의 간선의 가중치

found[]

거리가 확정되었는지 여부

- 최초: 시작 정점만 True, 나머지는 False

path[]

이전 정점의 인덱스

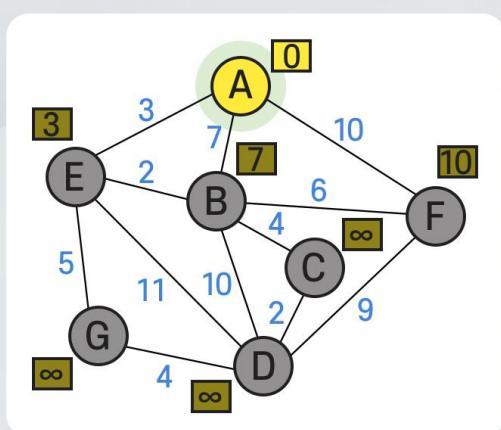
dist[]가 최소인 정점을 선택해 최단 거리를 확정하는 과정을 반복

최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

● 알고리즘 실행 과정: Step 1



$$S = \{ A \}$$

$$\begin{aligned}
 dist(A) &= w(A,A) = 0 \\
 dist(B) &= w(A,B) = 7 \\
 dist(C) &= w(A,C) = \infty \\
 dist(D) &= w(A,D) = \infty \\
 dist(E) &= w(A,E) = 3 \\
 dist(F) &= w(A,F) = 10 \\
 dist(G) &= w(A,G) = \infty
 \end{aligned}$$

dist[]: 시작 정점으로부터의 거리



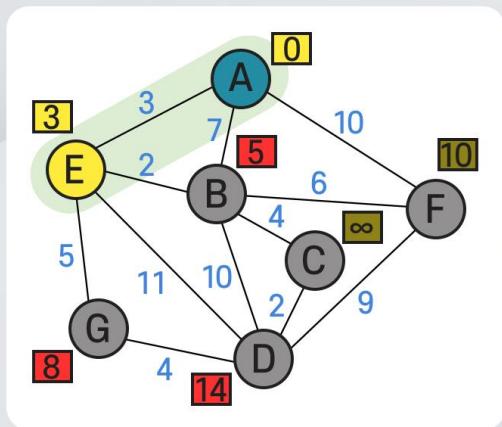
Dijkstra의 최단 경로 알고리즘

최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

- 알고리즘 실행 과정: Step 2



$$S = \{ A, E \}$$

$$\text{dist}(A)=0$$

$$\text{dist}(E)=w(A,E)=3$$

$$\text{dist}(B)=\min(\text{dist}(B), \text{dist}(E)+w(E,B))=\min(7, 3+2)=5$$

$$\text{dist}(C)=\min(\text{dist}(C), \text{dist}(E)+w(E,C))=\min(\infty, 3+\infty)=\infty$$

$$\text{dist}(D)=\min(\text{dist}(D), \text{dist}(E)+w(E,D))=\min(\infty, 3+11)=14$$

$$\text{dist}(F)=\min(\text{dist}(F), \text{dist}(E)+w(E,F))=\min(10, 3+\infty)=10$$

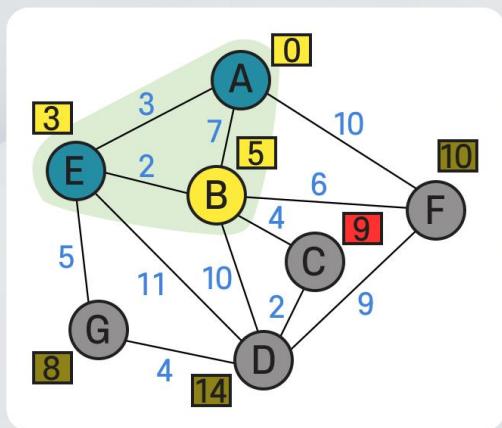
$$\text{dist}(G)=\min(\text{dist}(G), \text{dist}(E)+w(E,G))=\min(\infty, 3+5)=8$$

최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

- 알고리즘 실행 과정: Step 3



$$S = \{ A, E, B \}$$

$$\text{dist}(A)=0$$

$$\text{dist}(B)=5$$

$$\text{dist}(E)=3$$

$$\text{dist}(C)=\min(\text{dist}(C), \text{dist}(B)+w(B,C))=\min(\infty, 5+4)=9$$

$$\text{dist}(D)=\min(\text{dist}(D), \text{dist}(B)+w(B,D))=\min(14, 5+10)=14$$

$$\text{dist}(F)=\min(\text{dist}(F), \text{dist}(B)+w(B,F))=\min(10, 5+6)=10$$

$$\text{dist}(G)=\min(\text{dist}(G), \text{dist}(B)+w(B,G))=\min(8, 5+\infty)=8$$

Dijkstra의 최단 경로 알고리즘

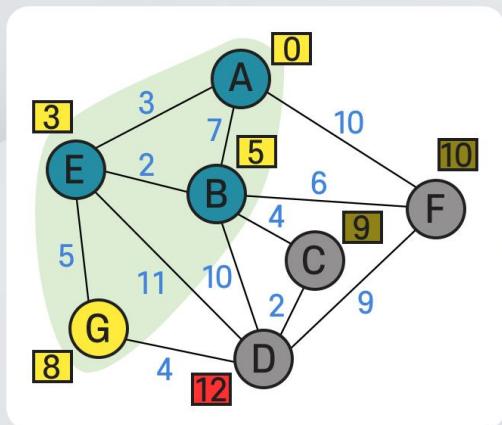


최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

- 알고리즘 실행 과정: Step 4



$$S = \{ A, E, B, G \}$$

$$\text{dist}(A)=0$$

$$\text{dist}(B)=5$$

$$\text{dist}(E)=3$$

$$\text{dist}(G)=8$$

$$\text{dist}(C)=\min(\text{dist}(C), \text{dist}(G)+w(G,C))=\min(9,8+\infty)=9$$

$$\text{dist}(D)=\min(\text{dist}(D), \text{dist}(G)+w(G,D))=\min(14,8+4)=12$$

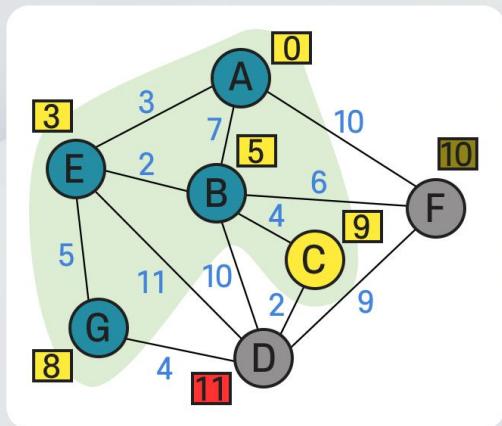
$$\text{dist}(F)=\min(\text{dist}(F), \text{dist}(G)+w(G,F))=\min(10,8+\infty)=10$$

최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

- 알고리즘 실행 과정: Step 5



$$S = \{ A, E, B, G, C \}$$

$$\text{dist}(A)=0$$

$$\text{dist}(B)=5$$

$$\text{dist}(E)=3$$

$$\text{dist}(G)=8$$

$$\text{dist}(D)=\min(\text{dist}(D), \text{dist}(C)+w(C,D))=\min(12,9+2)=11$$

$$\text{dist}(F)=\min(\text{dist}(F), \text{dist}(C)+w(C,F))=\min(10,9+\infty)=10$$



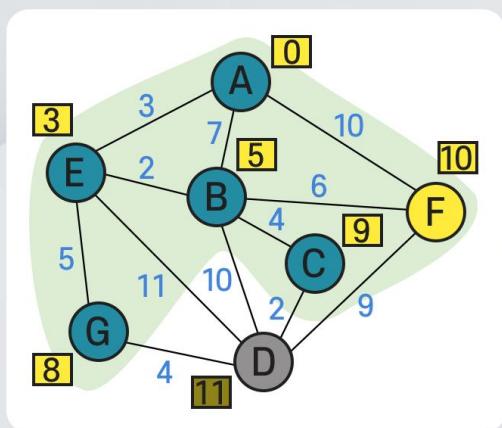
Dijkstra의 최단 경로 알고리즘

최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

- 알고리즘 실행 과정: Step 6



$$S = \{ A, E, B, G, C, F \}$$

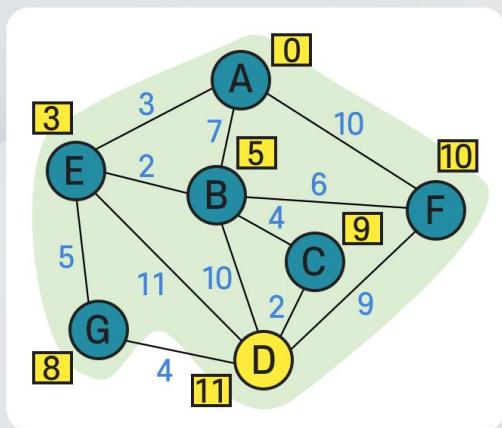
$$\begin{aligned} \text{dist}(A) &= 0 \\ \text{dist}(B) &= 5 \\ \text{dist}(C) &= 9 \\ \text{dist}(E) &= 3 \\ \text{dist}(F) &= 10 \\ \text{dist}(G) &= 8 \\ \text{dist}(D) &= \min(\text{dist}(D), \text{dist}(F) + w(F, D)) = \min(11, 10 + 9) = 11 \end{aligned}$$

최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

- 알고리즘 실행 과정: Step 7



$$S = \{ A, E, B, G, C, F, D \}$$

$$\begin{aligned} \text{dist}(A) &= 0 \\ \text{dist}(B) &= 5 \\ \text{dist}(C) &= 9 \\ \text{dist}(D) &= 11 \\ \text{dist}(E) &= 3 \\ \text{dist}(F) &= 10 \\ \text{dist}(G) &= 8 \end{aligned}$$

Dijkstra의 최단 경로 알고리즘



최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

```

def shortest_path_dijkstra(vertex, adj, s) :
    vsize = len(vertex)
    dist = list(adj[s])
    path = [s] * vsize
    found= [False] * vsize
    found[s] = True

    for i in range(vsize-1) :
        print("Step%2d: "%(i+1), dist)
        u = choose_vertex(dist, found)
        found[u] = True

        for w in range(vsize) :
            if not found[w] :
                if (dist[u] + adj[u][w] < dist[w]) :
                    dist[w] = dist[u] + adj[u][w]
                    path[w] = u

    return path

```

시작 정점과의 간선의 가중치(dist[s]=0)
 # 일단 모두 시작 정점으로 초기화
 # 최단거리 확정 여부(False로 초기화)
 # 최초에 s만 True 나머지는 False

 # 모든 정점을 포함해야 함
 # 단계별 dist[] 출력용
 # 최단 거리 정점 u
 # 이제 u까지의 거리는 확정됨

 # dist[] 갱신
 # 아직 거리가 확정되지 않은 정점들에 대해
 # u를 거치면 가깝다면...
 # dist[w] 갱신
 # w의 이전 정점은 u

 # 경로 출력

최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

● 시간 복잡도

Dijkstra

$O(n^2)$

- 주 반복문을 n 번 반복
- 내부 반복문을 $2n$ 번 반복



Dijkstra의 최단 경로 알고리즘



최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

```

def shortest_path_dijkstra(vertex, adj, s) :
    vsize = len(vertex)
    dist = list(adj[s])
    path = [s] * vsize
    found= [False] * vsize
    found[s] = True

    for i in range(vsize-1) :
        print("Step%2d: "%(i+1), dist)
        u = choose_vertex(dist, found)
        found[u] = True

        for w in range(vsize) :
            if not found[w] :
                if (dist[u] + adj[u][w] < dist[w]) :
                    dist[w] = dist[u] + adj[u][w]
                    path[w] = u

    return path

```

시작 정점과의 간선의 가중치(dist[s]=0)
 # 일단 모두 시작 정점으로 초기화
 # 최단거리 확정 여부(False로 초기화)
 # 최초에 s만 True 나머지는 False

 # 모든 정점을 포함해야 함
 # 단계별 dist[] 출력용
 # 최단 거리 정점 u
 # 이제 u까지의 거리는 확정됨

 # dist[] 갱신
 # 아직 거리가 확정되지 않은 정점들에 대해
 # u를 거치면 가깝다면...
 # dist[w] 갱신
 # w의 이전 정점은 u

 # 경로 출력

최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

● 시간 복잡도

Dijkstra

$O(n^2)$

- 주 반복문을 n 번 반복
- 내부 반복문을 $2n$ 번 반복
- 모든 정점 쌍의 최단 경로를 구하려면 n 번 반복 → $O(n^3)$



Dijkstra의 최단 경로 알고리즘



최단 경로 문제

Dijkstra의 최단 경로 알고리즘

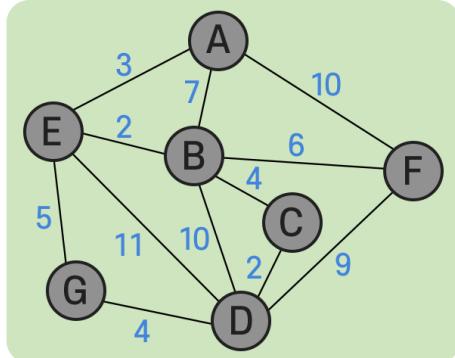
2 Dijkstra 알고리즘

● 테스트 프로그램

```

vertex = [ 'A', 'B', 'C', 'D', 'E', 'F', 'G' ]
weight = [
    [ 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, 13, 5 ],
    [ 10, 6, INF, 9, 13, 0, INF ],
    [ INF, INF, INF, 4, 5, INF, 0 ] ]
print("Shortest Path By Dijkstra Algorithm")
start = 0
path = shortest_path_dijkstra(vertex, weight, start)
...

```



최단 경로 문제

Dijkstra의 최단 경로 알고리즘

2 Dijkstra 알고리즘

● 테스트 프로그램

```

C:\WINDOWS\system32\cmd.exe
Shortest Path By Dijkstra Algorithm
Step 1: [0, 7, 999, 999, 3, 10, 999]
Step 2: [0, 5, 999, 14, 3, 10, 8]
Step 3: [0, 5, 9, 14, 3, 10, 8]
Step 4: [0, 5, 9, 12, 3, 10, 8]
Step 5: [0, 5, 9, 11, 3, 10, 8]
Step 6: [0, 5, 9, 11, 3, 10, 8]
Step 7: [0, 5, 9, 11, 3, 10, 8]
[최단경로: A->B] B <- E <- A
[최단경로: A->C] C <- B <- E <- A
[최단경로: A->D] D <- C <- B <- E <- A
[최단경로: A->E] E <- A
[최단경로: A->F] F <- A
[최단경로: A->G] G <- E <- A

```

각 단계별 dist[] 배열의 값 변화

A부터 모든 정점까지의 최단 경로



Floyd의 최단 경로 알고리즘

최단 경로 문제

Floyd의 최단 경로 알고리즘

1 Floyd의 전략

◎ Floyd 알고리즘의 개념

Floyd의 알고리즘

- 모든 정점 사이의 최단 경로 거리를 찾는 알고리즘
- Floyd-Warshall 알고리즘이라고도 함
- 동적 계획(Dynamic Programming) 전략을 이용
- 아이디어

- 모든 정점 사이의 최단 경로 거리를 구하려면 모든 정점을 거치는 상황을 고려
- 어떤 정점을 하나도 거치지 않는 (바로 가는) 경로에서부터 시작
- 정점을 하나씩 순차적으로 고려했을 때 경로를 갱신
- 최종적으로 모든 정점을 고려한 경로 거리를 구함

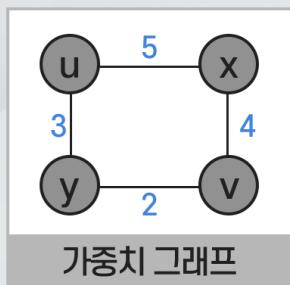
최단 경로 문제

Floyd의 최단 경로 알고리즘

1 Floyd의 전략

◎ 최종 결과→최단 거리 행렬

▶ 모든 정점에서 모든 정점까지의 거리 계산



```
V = [ 'u', 'v', 'x', 'y' ]
W = [ [ 0, INF, 5, 3], #u
      [ INF, 0, 4, 2], #v
      [ 5, 4, 0, INF], #x
      [ 3, 2, INF, 0] ] #y
```

인접 행렬 W

```
D = [ [ 0, 5, 5, 3],
      [ 5, 0, 4, 2],
      [ 5, 4, 0, 6],
      [ 3, 2, 6, 0] ]
```

최단 거리 행렬 D



Floyd의 최단 경로 알고리즘

최단 경로 문제

Floyd의 최단 경로 알고리즘

2 Floyd 알고리즘

그래프의 인접 행렬 표현 $W \rightarrow D$

- 최단거리 행렬 D
- $D^k[i][j] : 1 \sim k$ 번째 정점까지 만을 이용한 정점 $i \rightarrow j$ 의 최단 경로 거리
- 구하고자 하는 최적해: D^n

동적 계획 전략

- $D^0 \rightarrow D^1 \rightarrow D^2 \rightarrow \dots \rightarrow D^n$

최단 경로 문제

Floyd의 최단 경로 알고리즘

2 Floyd 알고리즘

일반적인 경우

- D^{k-1} 이 구해진 상태에서 k 번째 정점을 추가로 고려한 D^k 를 구하려고 함
- 두 가지 상황

- Case1: k 를 거치지 않는 경로

$$D^k[i][j] \leftarrow D^{k-1}[i][j]$$

- Case2: k 를 거치지 않는 경로

$$D^k[i][j] \leftarrow D^{k-1}[i][j] + D^{k-1}[k][j]$$

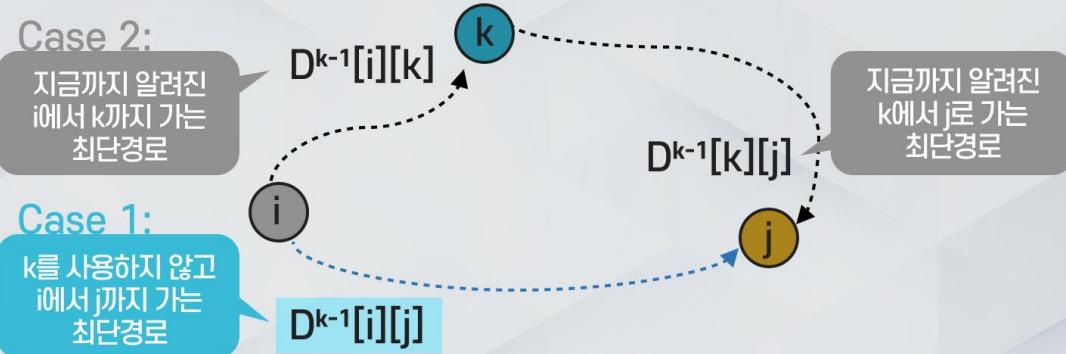
Floyd의 최단 경로 알고리즘



최단 경로 문제

Floyd의 최단 경로 알고리즘

2 Floyd 알고리즘



$$D^{k-1}[i][j] = \begin{cases} W[i][j] & k = 0 \\ \min(D^{k-1}[i][j], D^{k-1}[i][k] + D^{k-1}[k][j]) & \text{otherwise} \end{cases}$$

최단 경로 문제

Floyd의 최단 경로 알고리즘

2 Floyd 알고리즘

기반 상황

- D^0 : 아무런 정점을 거치지 않는 경로 거리 → 인접 행렬 W

일반 상황

- D^k : k까지의 정점만을 이용한 부분 문제의 최적해
- D^{k-1} 이 구해진 상태에서 k번째 정점을 고려할 때

- 후보 1: k를 거치지 않는 경로

$$D^k[i][j] \leftarrow D^{k-1}[i][j]$$

- 후보 2: k를 거치는 경로

$$D^k[i][j] \leftarrow D^{k-1}[i][k] + D^{k-1}[k][j]$$



Floyd의 최단 경로 알고리즘

최단 경로 문제

Floyd의 최단 경로 알고리즘

2 Floyd 알고리즘

● 알고리즘(유사코드)

- 2차원 배열 A를 이용하여 3중 반복을 하는 루프로 구성
- 배열 A의 초기값은 인접 행렬의 가중치

▶ Floyd-Warshall의 최단 경로 알고리즘

```
shortest_path_floyd()
for k ← 0 to n - 1
    for i ← 0 to n - 1
        for j ← 0 to n - 1
            A[i][j] = min(A[i][j], A[i][k] + A[k][j])
```

최단 경로 문제

Floyd의 최단 경로 알고리즘

2 Floyd 알고리즘

```
def shortest_path_floyd(vertex, adj) :
    vsize = len(vertex)                                # 정점의 개수

    A = list(adj)                                     # 2차원 배열(리스트의 리스트)의 복사
    for i in range(vsize) :
        A[i] = list(adj[i])

    for k in range(vsize) :                            # 모든 정점을 순서대로 고려함
        for i in range(vsize) :
            for j in range(vsize) :                    # 정점 i에서 정점 j까지의 거리
                if (A[i][k] + A[k][j] < A[i][j]) :
                    A[i][j] = A[i][k] + A[k][j]
    printA(A)                                         # 진행상황 출력용
```

Floyd의 최단 경로 알고리즘

최단 경로 문제

Floyd의 최단 경로 알고리즘

2 Floyd 알고리즘

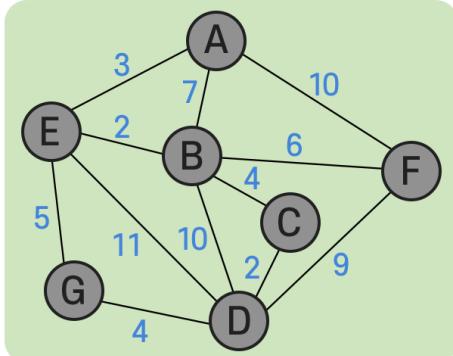
● 테스트 프로그램

```

vertex = [ 'A', 'B', 'C', 'D', 'E', 'F', 'G' ]
weight = [
    [ 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, 13, 5 ],
    [ 10, 6, INF, 9, 13, 0, INF ],
    [ INF, INF, INF, 4, 5, INF, 0 ] ]

print("Shortest Path By Dijkstra Algorithm")
shortest_path_floyd(vertex, weight)

```



최단 경로 문제

Floyd의 최단 경로 알고리즘

2 Floyd 알고리즘

● 실행결과

```

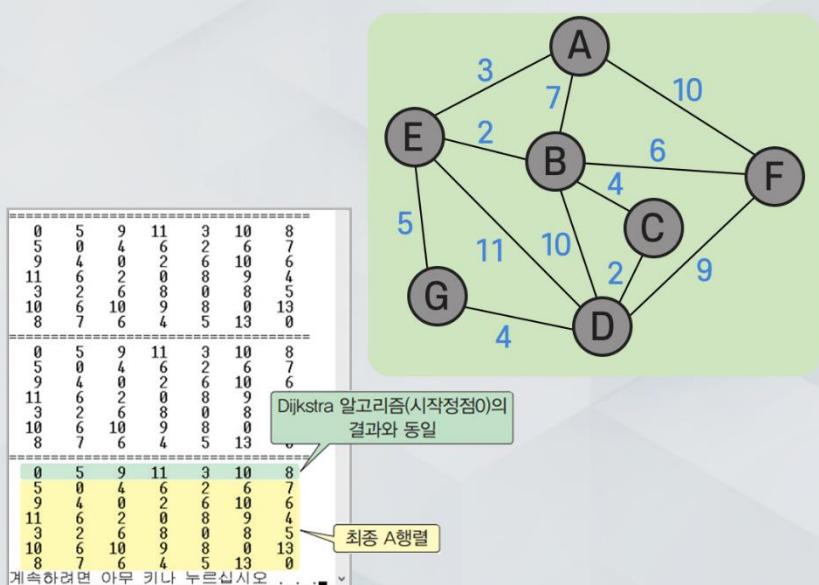
C:\WINDOWS\system32\cmd.exe - □ ×
=====
Shortest Path By Floyd's Algorithm
=====
0 7 INF INF 3 10 INF
7 0 4 10 2 6 INF
INF 4 0 2 INF INF INF
10 10 2 0 11 9 4
3 2 INF 11 0 13 5
10 6 INF 9 13 0 INF
INF INF INF 4 5 INF 0

0 7 11 17 3 10 INF
7 0 4 10 2 6 INF
11 4 0 2 6 10 INF
17 10 2 0 11 9 4
3 2 6 11 0 8 5
10 6 10 9 8 0 INF
INF INF INF 4 5 INF 0

0 7 11 13 3 10 INF
7 0 4 6 2 6 10
11 4 0 2 6 10 6
13 6 2 0 8 9 4
3 2 6 8 0 8 5
10 6 10 9 8 0 INF
INF INF INF 4 5 INF 0

0 7 11 13 3 10 17
7 0 4 6 2 6 10
11 4 0 2 6 10 6
13 6 2 0 8 9 4
3 2 6 8 0 8 5
10 6 10 9 8 0 13
17 10 6 4 5 13 0

```



Floyd의 최단 경로 알고리즘



최단 경로 문제

Floyd의 최단 경로 알고리즘

2 Floyd 알고리즘

● 시간 복잡도

Floyd-Warshall

$O(n^3)$

- 모든 정점 쌍의 최단 경로 거리를 구함
- 3중 반복문을 실행
- 매우 간결한 반복 구문을 사용

Dijkstra

$O(n^2)$

- 모든 정점 쌍의 최단 경로를 구하려면 n번 반복



$O(n^3)$