동적 프로그래밍 방법_2

2020년도 2학기 최 희석

목차

- 스트링 편집 거리 문제
- 모든 정점 간의 최단 경로(플로이드)

스트링 편집 거리 문제

스트링 편집 거리 문제의 개념과 원리

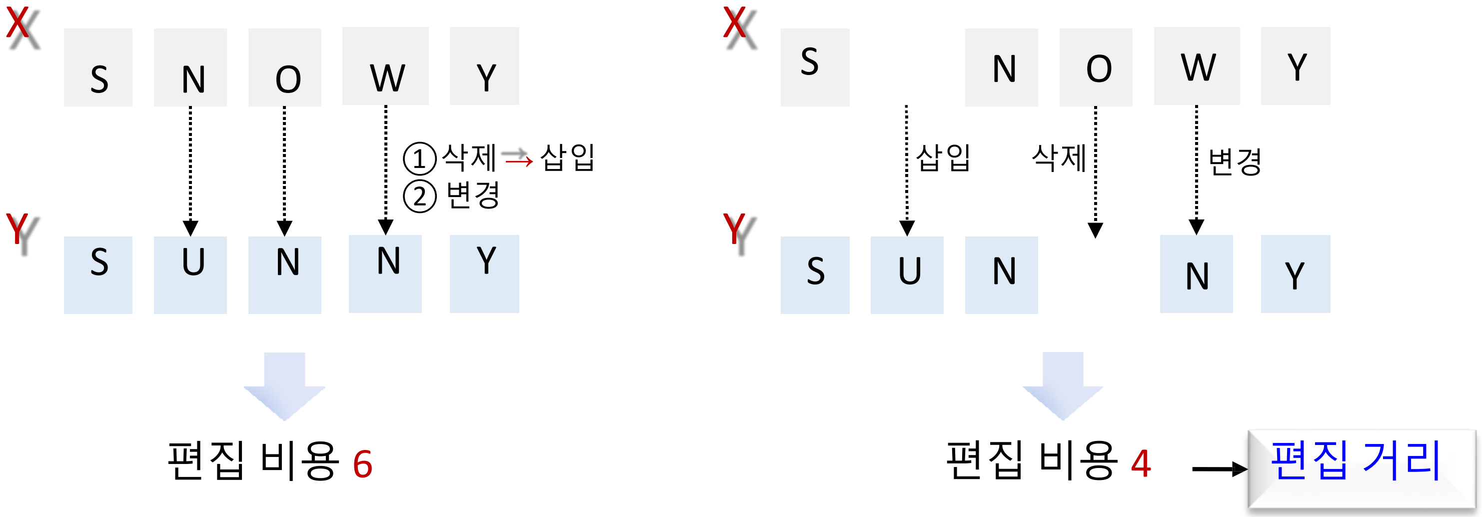
- 두 문자열 X와 Y 사이의 **편집 거리 edit distance**
  - 두 문자열 사이의 근접성 또는 유사성을 판단하는 척도
  - $X = x_1 x_2 \ldots x_n$ $Y = y_1 y_2 \ldots y_m$
  - 문자열 X를 Y로 변환하는 데 필요한 전체 편집 연산에 대한 최소 비용
    - 특정 위치에 새 문자를 삽입하는 연산 → 삽입 비용 δ_I
    - 특정 위치의 문자를 삭제하는 연산 → 삭제 비용 δ_D
    - 특정 위치의 문자를 다른 문자로 변경하는 연산 → 변경 비용 δ_C

스트링 편집 거리 문제의 개념과 원리

$$X = x_1 x_2 x_3 x_4 x_5 = SNOWY$$

$$Y = y_1 y_2 y_3 y_4 y_5 = SUNNY$$

$\delta_D = \delta_I = 1$ $\delta_C = 2$



스트링 편집 거리 문제의 최적성의 원리

- X와 Y 사이의 편집 거리는 이들의 부분 문자열 사이의 편집 거리를 포함
  - X의 마지막 글자가 Y의 마지막 글자와 같거나 같게 변경된 경우
    - $x_1x_2 \ldots x_{n-1}$과 $y_1y_2 \ldots y_{m-1}$ 의 최소 편집 비용
      \+ 마지막 글자의 변경 비용 δ_C (일치한 경우에는 비용 0)
  - X의 마지막 글자가 삭제된 경우
    - $x_1x_2 \ldots x_{n-1}$과 $y_1y_2 \ldots y_m$ 의 최소 편집 비용 + X의 마지막 글자의 삭제 비용 δ_D
  - Y의 마지막 글자가 삽입된 경우
    - $x_1x_2 \ldots x_n$과 $y_1y_2 \ldots y_{m-1}$ 의 최소 편집 비용 + Y의 마지막 글자의 삽입 비용 δ_I

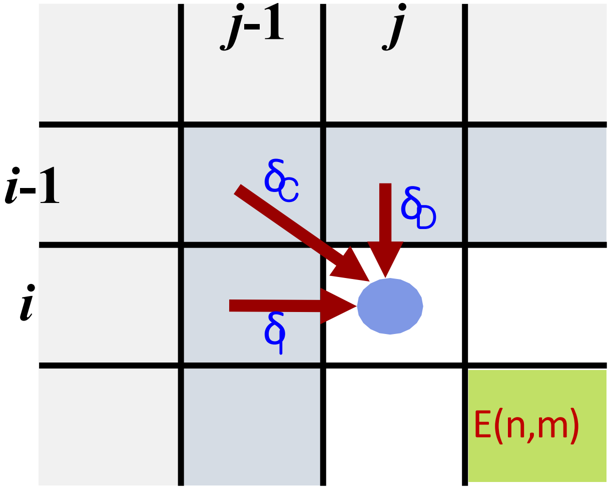
스트링 편집 거리 문제의 점화

- X와 Y 사이의 편집 거리는 이들의 부분 문자열 사이의 편집 거리를 포함
  - X의 마지막 글자가 Y의 마지막 글자와 같거나 같게 변경된 경우
    - $x_1x_2 \ldots x_{n-1}$과 $y_1y_2 \ldots y_{m-1}$ 의 최소 편집 비용
      \+ 마지막 글자의 변경 비용 δ_C (일치한 경우에는 비용 0)
  - X의 마지막 글자가 삭제된 경우
    - $x_1x_2 \ldots x_{n-1}$과 $y_1y_2 \ldots y_m$ 의 최소 편집 비용 + X의 마지막 글자의 삭제 비용 δ_D
  - Y의 마지막 글자가 삽입된 경우
    - $x_1x_2 \ldots x_n$과 $y_1y_2 \ldots y_{m-1}$ 의 최소 편집 비용 + Y의 마지막 글자의 삽입 비용 δ_I

- $X_i = x_1x_2 \ldots x_i$와 $Y_j = y_1y_2 \ldots y_j$ 사이의 편집 거리

$$
\begin{aligned}
E(i, j) = \min[&E(i-1, j) + \delta_D, \\
&E(i, j-1) + \delta_I, \\
&E(i-1, j-1) + (0/\delta_C)]
\end{aligned}
$$

$$
0/\delta_C = \begin{cases} 0, & x_i = y_i \\ \delta_c, & x_i \neq y_i \end{cases}
$$



스트링 편집 거리 문제의 알고리즘

```
ED (n, X[], m, Y[], ins, del, chg)
{
   int E[n+1][m+1], i, j;
   E[0][0] = 0;
   for (i=1; i<n+1; i++)  E[i][0]     // 첫 열의 초기화
      = E[i-1][0] + del;
   for (j=1; j<m+1; j++)              // 첫 행의 초기화
      E[0][j] = E[0][j-1] + ins;
   for (i=1; i<n+1; i++)
      for (j=1; j<m+1; j++) {
         c = (X[i] == Y[j]) ? 0 : chg;
         E[i][j] = min( E[i-1][j]+del, E[i][j-1]+ins, E[i-1][j-1]+c );
      }
   return E[n][m];
}
```

입력 : 문자 배열 X[1..n], Y[1..m]
삽입 비용 ins, 삭제 비용 del, 변경 비용 chg
출력 : E[n][m] : 편집 거리

스트링 편집 거리 문제의 적용 예시_1

$X = bbabb,\ \ Y = abaa$ $\delta_D = \delta_I = 1$ $\delta_C = 2$

E(1,1) = min[E(0,1)+1, E(1,0)+1, E(0,0)+2]
= min[2,2,2] = 2

E(1,2) = min[E(0,2)+1, E(1,1)+1, E(0,1)+0]
= min[3,3,1] = 1

E(1,3) = min[E(0,3)+1, E(1,2)+1, E(0,2)+2]
= min[4,2,3] = 2

⋮

E(5,4) = min[E(4,4)+1, E(5,3)+1, E(4,3)+2]
= min[5,5,5] = 5

E(i, j)		a	b	a	a
	0	1	2	3	4
b	1	2	1	2	3
b	2	3	2	3	4
a	3	2	3	2	3
b	4	3	2	3	4
b	5	4	3	4	5

스트링 편집 거리 문제의 적용 예시_2

$X = SNOWY, \; Y = SUNNY$

$\delta_D = \delta_I = 1 \; \delta_C = 2$

E(i, j)		S	U	N	N	Y
	0	1	2	3	4	5
S	1	0	1	2	3	4
N	2	1	2	1	2	3
O	3	2	3	2	3	4
W	4	3	4	3	4	5
Y	5	4	5	4	5	4

스트링 편집 거리 문제의 성능 분석

```
ED (n, X[], m, Y[], ins, del, chg)
{
   int E[n+1][m+1], i, j;
   E[0][0] = 0;
   for (i=1; i<n+1; i++)  E[i][0]
      = E[i-1][0] + del;                    O(n)
   for (j=1; j<m+1; j++)  E[0][j
      ] = E[0][j-1] + ins;                  O(m)
   for (i=1; i<n+1; i++)
      for (j=1; j<m+1; j++) {               O(nm)
         c = (X[i] == Y[j]) ? 0 : chg;
         E[i][j] = min( E[i-1][j]+del, E[i][j-1]+ins, E[i-1][j-1]+c );
      }
   return E[n][m];
}
```

스트링 편집 거리 문제의 특징

- P (i,j) ← E (i,j)로 선택되는 최솟값이 어떤 연산으로 결정되는 지를 표시

  ⇒ 적용된 편집 연산을 구할 수 있음

E(i, j)		S	U	N	N	Y
	0	1	2	3	4	5
S	1	0	1	2	3	4
N	2	1	2	1	2	3
O	3	2	3	2	3	4
W	4	3	4	3	4	5
Y	5	4	5	4	5	4

P(i, j)		S	U	N	N	Y
	0	1	2	3	4	5
S	1	╲	—	—	—	—
N	2	|	✱	╲	╲	—
O	3	|	✱	|	✱	✱
W	4	|	✱	|	✱	✱
Y	5	|	✱	|	✱	╲

모든 정점 간의 최단 경로

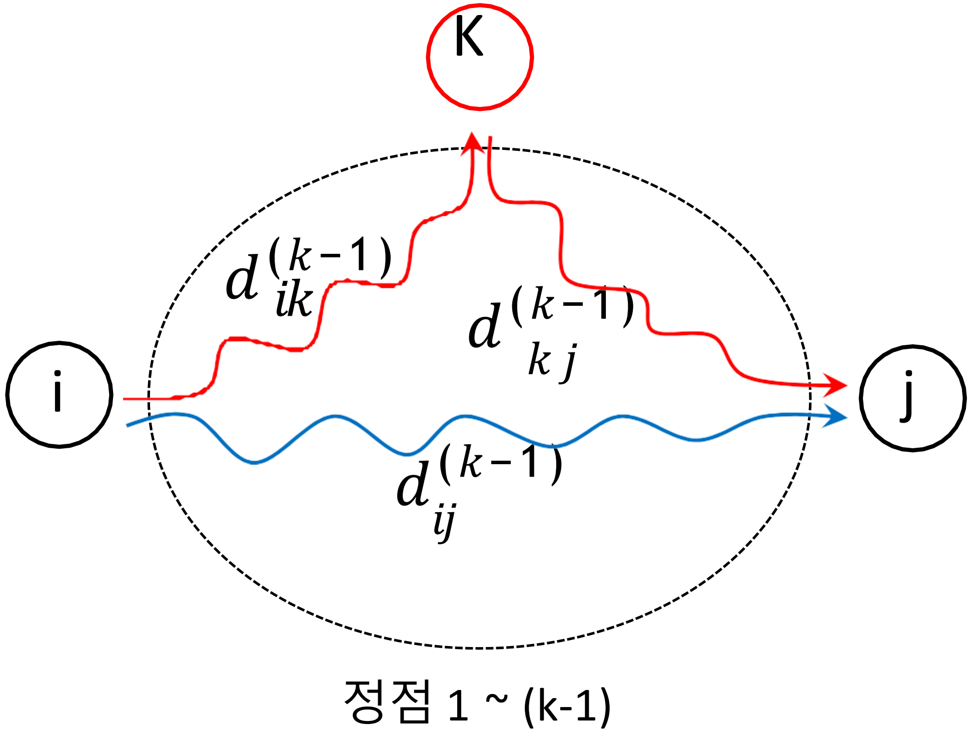
모든 정점 간의 최단 경로의 개념과 원리

- **두 정점 간의 최단 경로**shortest path
  - 가중 방향 그래프에서 두 정점을 연결하는 경로 중에서 간선의 가중치의 합이 가장 작은 경로
  - 유형
    - 하나의 특정 정점에서 다른 모든 정점으로의 최단 경로
      → 데이크스트라 Dijkstra 알고리즘 (욕심쟁이 방법)
    - 모든 정점에서 다른 모든 정점으로의 최단 경로
      → **플로이드 Floyd 알고리즘**

모든 정점 간의 최단 경로의 개념과 원리

- 모든 정점 간의 최단 경로
  - 가정 → 가중치의 합이 음수인 사이클은 존재하지 않음
- 플로이드 알고리즘
  - 동적 프로그래밍 방법 적용
  - 간선의 인접 행렬 표현을 활용하여 경유할 수 있는 정점의 범위가 1인 경로부터 시작해서 |V|까지인 경로까지 단계적으로 범위를 늘려 가면서 모든 정점 간의 최단 경로를 구하는 알고리즘
  - 인접 행렬 $D^{(k)} = (d_{ij}^{(k)})$ k=0,1,…,|V|
    - $d_{ij}^{(k)}$: 정점 i에서 j까지의 경로 중에서 정점 1부터 k까지의 정점만을 경유할 수 있는 최단 경로의 길이

플로이드 알고리즘의 최적성의 원리



플로이드 알고리즘의 점화식

- G=(V, E), | V |=n
  - D (I, j, 0) → 정점 i에서 정점 j까지 경유하는 정점 없이 간선으로 직접 연결된 경로의 길이

$$D(i,j,0) = d_{ij}^{0} = w_{ij}$$

  - D (i, j, k) → 정점 i에서 정점 j까지의 경로 중에서 정점 1부터 k까지 정점만을 경유할 수 있는 최소 경로의 길이

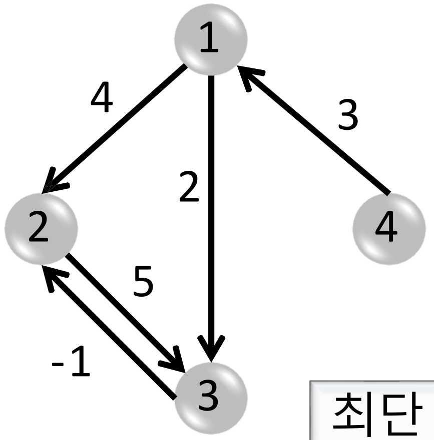
$$D(i,j,k) = d_{ij}^{(k)} = min[\ d_{ij}^{(k-1)},\ d_{ik}^{(k-1)} + d_{kj}^{(k-1)}]$$

플로이드 알고리즘

입력: G=(V,E)
출력: D[][] : 모든 정점에서 모든 정점으로의 최단 경로의 길이

```
Floyd (G)
{
  D[][] ←입력 간선의 인접 행렬로 초기화 f
  or (k=1부터 |V|까지)
      for (i=1부터 |V|까지)       //시작 정점
         for (j=1부터 |V|까지)    //끝 정점
            if ( D[i][j] > D[i][k] + D[k][j] )
               D[i][j] = D[i][k] + D[k][j];
  return D[][];
}
```

플로이드 알고리즘의 적용 예



$$D^{(k)} = \begin{bmatrix} d_{11}^{(k)} & d_{12}^{(k)} & d_{13}^{(k)} & d_{14}^{(k)} \\ d_{21}^{(k)} & d_{22}^{(k)} & d_{23}^{(k)} & d_{24}^{(k)} \\ d_{31}^{(k)} & d_{32}^{(k)} & d_{33}^{(k)} & d_{34}^{(k)} \\ d_{41}^{(k)} & d_{42}^{(k)} & d_{43}^{(k)} & d_{44}^{(k)} \end{bmatrix}$$

최단 경로 계산에 사용되는 점화식

$$D^{(k)} = (d_{ij}^{(k)}) = \min[d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}]$$

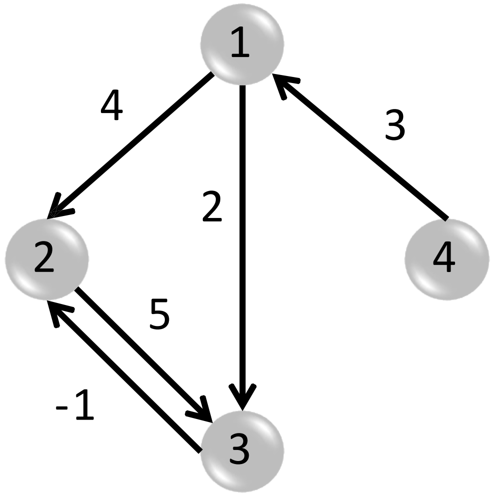
$$D^{(0)} = (d_{ij}^{(0)}) = (w_{ij})$$

$$D^{(0)} \rightarrow D^{(1)} \rightarrow D^{(2)} \rightarrow \cdots \rightarrow D^{(n)}$$

플로이드 알고리즘의 적용 예

$$D^{(k)} = \begin{bmatrix} d_{11}^{(k)} & d_{12}^{(k)} & d_{13}^{(k)} & d_{14}^{(k)} \\ d_{21}^{(k)} & d_{22}^{(k)} & d_{23}^{(k)} & d_{24}^{(k)} \\ d_{31}^{(k)} & d_{32}^{(k)} & d_{33}^{(k)} & d_{34}^{(k)} \\ d_{41}^{(k)} & d_{42}^{(k)} & d_{43}^{(k)} & d_{44}^{(k)} \end{bmatrix}$$

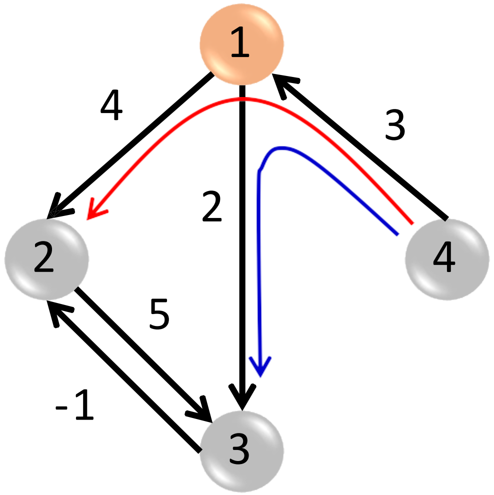
$D^{(0)}$



$$D^{(0)} = \begin{bmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & \infty & \infty & 0 \end{bmatrix}$$

플로이드 알고리즘의 적용 예

D(1)



$$D^{(0)} = \begin{bmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & \infty & \infty & 0 \end{bmatrix}$$

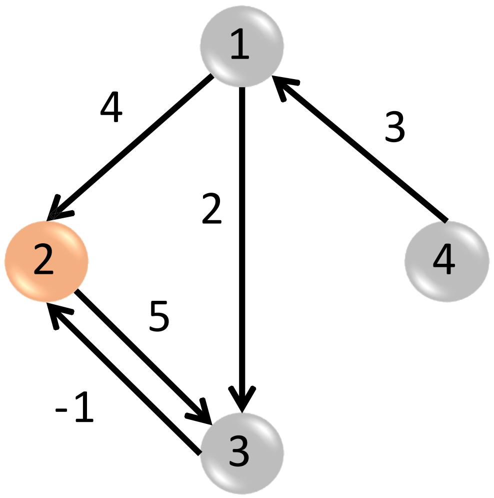
$$d_{42}^{(1)} = \min\{d_{42}^{(0)}, d_{41}^{(0)} + d_{12}^{(0)}\} = \min\{\infty, 3+4\} = 7$$

$$d_{43}^{(1)} = \min\{d_{43}^{(0)}, d_{41}^{(0)} + d_{13}^{(0)}\} = \min\{\infty, 3+2\} = 5$$

$$D^{(1)} = \begin{bmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 7 & 5 & 0 \end{bmatrix}$$

플로이드 알고리즘의 적용 예

$D^{(2)}$



$$D^{(1)} = \begin{bmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 7 & 5 & 0 \end{bmatrix}$$

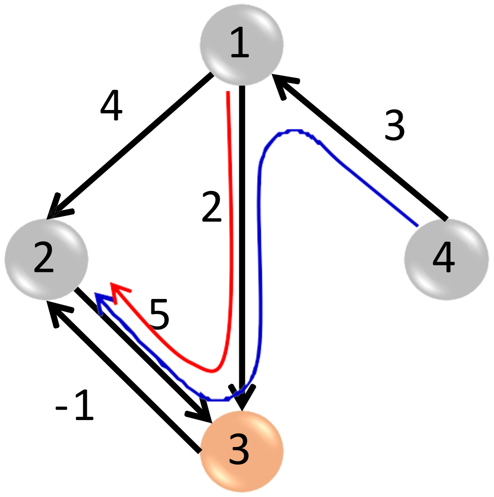
$$d_{13}^{(2)} = \min\{d_{13}^{(1)}, d_{12}^{(1)} + d_{23}^{(1)}\} = \min\{2, 4+5\} = 2$$

$$d_{43}^{(2)} = \min\{d_{43}^{(1)}, d_{42}^{(1)} + d_{23}^{(1)}\} = \min\{5, 7+5\} = 5$$

$$D^{(2)} = \begin{bmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 7 & 5 & 0 \end{bmatrix}$$

플로이드 알고리즘의 적용 예

$D^{(3)}$



$$D^{(2)} = \begin{bmatrix} 0 & 4 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 7 & 5 & 0 \end{bmatrix}$$

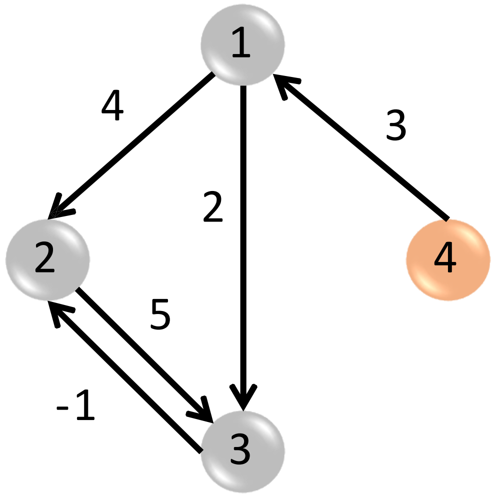
$$d_{12}^{(3)} = \min\{d_{12}^{(2)}, d_{13}^{(2)} + d_{32}^{(2)}\} = \min\{4, 2+(-1)\} = 1$$

$$d_{42}^{(3)} = \min\{d_{42}^{(2)}, d_{43}^{(2)} + d_{32}^{(2)}\} = \min\{7, 5+(-1)\} = 4$$

$$D^{(3)} = \begin{bmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 4 & 5 & 0 \end{bmatrix}$$

플로이드 알고리즘의 적용 예

$D^{(4)}$



$$D^{(3)} = \begin{bmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 4 & 5 & 0 \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 4 & 5 & 0 \end{bmatrix}$$

플로이드 알고리즘의 성능분석

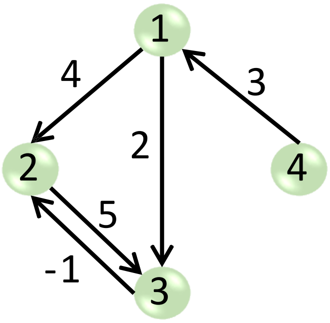
```
Floyd (G)
{
   D[][] ←입력 간선의 인접 행렬로 초기화
   for (k=1부터 |V|까지)
      for (i=1부터 |V|까지)
         for (j=1부터 |V|까지)
            if ( D[i][j] > D[i][k] + D[k][j] )
               D[i][j] = D[i][k] + D[k][j];
   return D[][];
}
```

$O(|V|^2)$

$O(|V|^3)$

플로이드 알고리즘의 특징

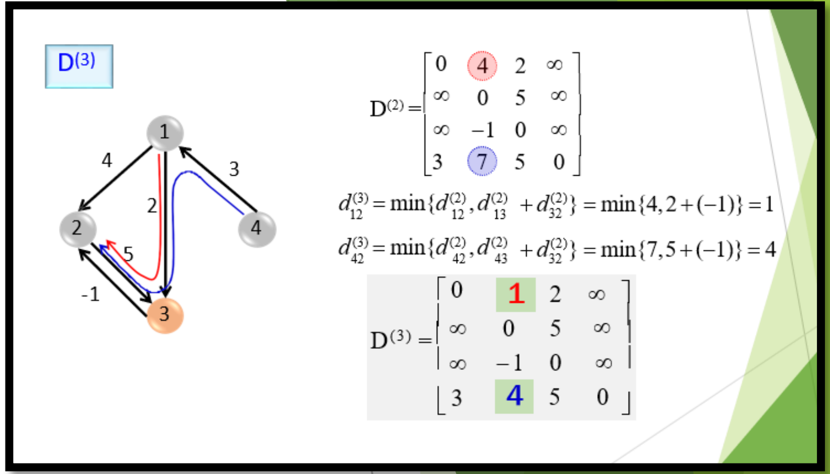
- 최단 경로 자체를 구할 수 있음
  - P(i,j) ← k if $d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$



$$D^{(4)} = \begin{bmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 5 & \infty \\ \infty & -1 & 0 & \infty \\ 3 & 4 & 5 & 0 \end{bmatrix}$$

P(i, j)

0	3	0	0
0	0	0	0
0	0	0	0
0	3	1	0



P(4,2)=3

$\underbrace{\text{정점4} \rightarrow \text{정점3}}_{\text{P(4,3)=1}} \underbrace{\rightarrow \text{정점2}}_{\text{P(3,2)=0}}$

P(4,3)=1 P(3,2)=0

$\underbrace{\text{정점4} \rightarrow \text{정점1}}_{\text{P(4,1)=0}} \underbrace{\rightarrow \text{정점3}}_{\text{P(1,3)=0}}$

P(4,1)=0 P(1,3)=0

정점4 → 정점1 → 정점3 → 정점2

과제 안내

5주차 과제

- 스트링 편집 거리 문제, 플로이드 알고리즘 구현하기
  - e-Class 업로드

- 양식 (한글, 워드,PDF -> 자유)

- 파일명 (이름_학번_전공)
  - 예) 최희석_2014182009_게임공학

- 질의 응답은 e-Class 질의응답 게시판에 남겨 주시길 바랍니다.