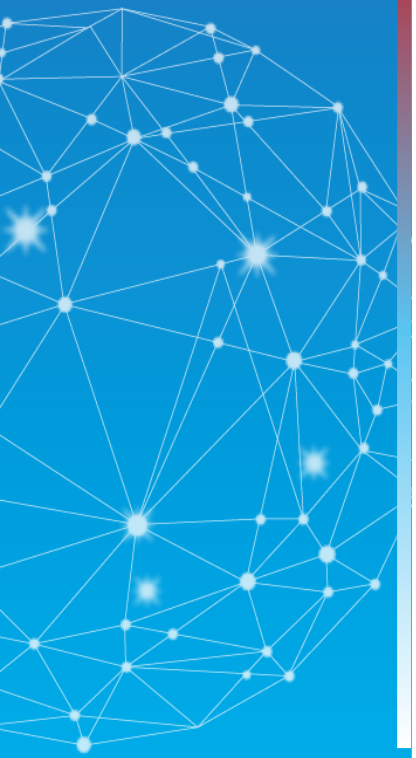
교육부 Ministry of Education

NRF 한국연구재단 National Research Foundation of Korea

6장. 네트워크 모형

- 참고문헌: 최적 의사결정을 위한 경영과학, 권수태외 5인, 청람, 2018
  알기쉬운 알고리즘, 양성봉, 생능, 2021

Contents



- 그래프 개요
- 네트워크 모형 개요
- 최소걸침나무
- 그리디 알고리즘
- 최단경로문제
- 최대흐름문제
- 최소비용 용량제약 네트워크 문제
- CPM/PERT

6. 네트워크 모형

6-1 그래프 개요
6-2 네트워크 모형의 개요(개념, 표현, 탐색)
6-3 최소걸침나무문제
6-4 그리디 알고리즘
6-5 최단경로문제
6-6 최대흐름문제
6-7 최소비용 용량제약 네트워크 문제
6-8 CPM/PERT

6.5 최단경로 문제

❖ 최단 경로 (Shortest Path) 문제

- ➢ 주어진 가중치 그래프에서 어느 한 출발점에서 또 다른 도착점까지의 최단 경로를 찾는 문제
- ➢ 최단 경로를 찾는 가장 대표적인 알고리즘
  - ✓ 다익스트라(Dijkstra) 알고리즘
    - 주어진 네트워크 내에서 하나의 마디(출발 마디)로부터 모든 다른 마디로의 최단경로를 구하기 위해 개발
  - ✓ Floyd-Warshall 알고리즘
    - 주어진 네트워크 내에서 임의의 두 마디 사이의 모든 최단경로들을 구하기 위해 개발

6.5 최단경로 문제

❖ 다익스트라(Dijkstra) 알고리즘

➢ 기호 정의

- ✓ v_i = 출발마디 1로부터 마디 i 까지의 최단거리,
- ✓ d_{ij} = 호 (i,j) 사이의 길이
- ✓ 마디 i 로부터 곧바로 연결되는 마디 j 에 대한 라벨을 다음과 같이 정의:

$$[v_j, \mathrm{i}] = [v_i + d_{ij},\ \ i],\ d_{ij} \geq 0$$

- ✓ 출발 마디에 대한 라벨은 [0, -] 로 표시 : 출발마디의 선행마디는 없음을 나타냄

➢ 라벨

- ✓ 임시라벨: 어떤 마디까지의 더 짧은 경로가 발견될 수 있다면 수정될 수 있음
- ✓ 영구라벨: 더 좋은 경로가 발견되지 않을 때, 임시라벨은 영구라벨로 변경

6.5 최단경로 문제

❖ 다익스트라(Dijkstra) 알고리즘

➢ 절차

✓ 단계 0 : 출발마디 (마디 1)에 영구라벨 [0, -] 을 붙이고 $i = 1$ 로 놓는다.

✓ 단계 k :

(1) 마디 j 가 영구라벨이 아닐 경우, 마디 i 로부터 직접 연결되는 마디 j 에 대해 , 임시라벨 $[v_i + d_{ij},\ \ i]$ 을 계산

마디 j 가 다른 마디 k 를 통해 라벨 $[v_j + \mathrm{k}]$ 를 가지고 있는 경우, $v_i + d_{ij} < v_j$ 이면 $[v_j + \mathrm{k}]$ 를 $[v_i + d_{ij},\ \ i]$ 로 대체

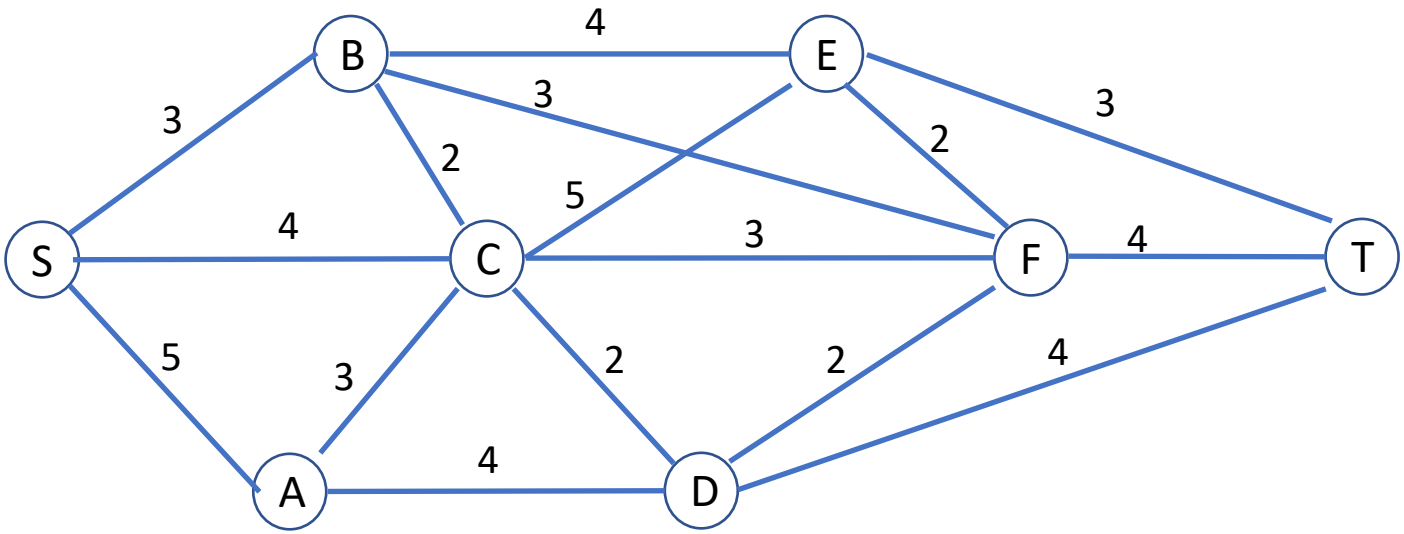
(2) 모든 마디들이 영구라벨을 가진다면 이 절차를 멈춤

아니면, 모든 임시라벨들 중에 최단거리(= v_m)를 갖는 라벨 $[v_m, l]$ 을 선택(동일한 값이 존재할 경우 임의로 선택). i=m 으로 놓고 단계 k 반복

6.5 최단경로 문제

❖ 예제

S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 예제

➢ 단계 0: 마디 s 에 영구라벨 [0, -] 을 붙이고 $i = 1$ 로 놓는다.

➢ 단계 1:

✓ 마지막 영구라벨인 마디 s로부터 마디 A, B, C 가 직접 연결되므로 각 마디에 대한 라벨을 다음과 같이 계산

마디	라벨	상태
S	[0, -]	영구라벨
A	[0 + 5, S] = [5, S]	임시라벨
B	[0 + 3, S] = [3, S]	임시라벨
C	[0 + 4, S] = [4, S]	임시라벨

✓ 3개의 임시라벨 중 마디B 가 최소거리를 제공함으로 마디 B를 영구마디로 변경

6.5 최단경로 문제

❖ 예제

➢ 단계 2:

✓ 마지막 영구라벨인 마디 B로부터 마디 C, E, F 가 직접 연결되므로 각 마디에 대한 라벨을 다음과 같이 계산

마디	라벨	상태
S	[0, -]	영구라벨
A	[0 + 5, S] = [5, S]	임시라벨
B	[3, S]	영구라벨
C	[0 + 4, S] = [4, S] [3 + 2, B] = [5, B] X	임시라벨
E	[3 + 4, B] = [7, B]	임시라벨
F	[3 + 3, B] = [6, B]	임시라벨

✓ 4개의 임시라벨 중 마디 C 가 최소거리를 제공함으로 마디 C를 영구마디로 변경

6.5 최단경로 문제

❖ 예제

➢ 단계 3:

✓ 마지막 영구라벨인 마디 C로부터 마디 E, F 가 직접 연결되므로 각 마디에 대한 라벨을 다음과 같이 계산

마디	라벨	상태
S	[0, -]	영구라벨
A	[0 + 5, S] = [5, S]	임시라벨
B	[3, S]	영구라벨
C	[0 + 4, S] = [4, S]	영구라벨
D	[4 + 2, C] = [6, C]	임시라벨
E	[3 + 4, B] = [7, B] [4 + 5, C] = [9, C] X	임시라벨
F	[3 + 3, B] = [6, B] [4 + 3, C] = [7, C] X	임시라벨

✓ 4개의 임시라벨 중 마디 A 가 최소거리를 제공함으로 마디 A를 영구마디로 변경

6.5 최단경로 문제

❖ 예제

➢ 단계 4:

✓ 마지막 영구라벨인 마디 A로부터 마디 D 가 직접 연결되므로 각 마디에 대한 라벨을 다음과 같이 계산

마디	라벨	상태
S	[0, -]	영구라벨
A	[0 + 5, S] = [5, S]	영구라벨
B	[3, S]	영구라벨
C	[0 + 4, S] = [4, S]	영구라벨
D	[4 + 2, C] = [6, C] [5 + 4, A] = [9, A] X	임시라벨
E	[3 + 4, B] = [7, B]	임시라벨
F	[3 + 3, B] = [6, B]	임시라벨

✓ 3개의 임시라벨 중 마디 D 가 최소거리를 제공함으로 마디 D를 영구마디로 변경

6.5 최단경로 문제

❖ 예제

➢ 단계 5:

✓ 마지막 영구라벨인 마디 D로부터 마디 F, T 가 직접 연결되므로 각 마디에 대한 라벨을 계산

마디	라벨	상태
S	[0, -]	영구라벨
A	[0 + 5, S] = [5, S]	영구라벨
B	[3, S]	영구라벨
C	[0 + 4, S] = [4, S]	영구라벨
D	[4 + 2, C] = [6, C]	영구라벨
E	[3 + 4, B] = [7, B]	임시라벨
F	[3 + 3, B] = [6, B] [4 + 3, C] = [7, C] X [6 + 2, D] =[8, D] X	임시라벨
T	[6 + 4, D] = [10, D]	임시라벨

✓ 3개의 임시라벨 중 마디 F 가 최소거리를 제공함으로 마디 F를 영구마디로 변경

6.5 최단경로 문제

❖ 예제

➢ 단계 6:

✓ 마지막 영구라벨인 마디 F로부터 마디 T 가 직접 연결되므로 각 마디에 대한 라벨을 계산

마디	라벨	상태
S	[0, -]	영구라벨
A	[0 + 5, S] = [5, S]	영구라벨
B	[3, S]	영구라벨
C	[0 + 4, S] = [4, S]	영구라벨
D	[4 + 2, C] = [6, C]	영구라벨
E	[3 + 4, B] = [7, B]	임시라벨
F	[3 + 3, B] = [6, B]	영구라벨
T	[6 + 4, D] = [10, D] [6 + 4, F] = [10, F]	임시라벨

✓ 2개의 임시라벨 중 마디 E 가 최소거리를 제공함으로 마디 E를 영구마디로 변경

6.5 최단경로 문제

❖ 예제

➢ 단계 7:

✓ 마지막 영구라벨인 마디 F로부터 마디 T 가 직접 연결되므로 각 마디에 대한 라벨을 계산

마디	라벨	상태
S	[0, -]	영구라벨
A	[0 + 5, S] = [5, S]	영구라벨
B	[3, S]	영구라벨
C	[0 + 4, S] = [4, S]	영구라벨
D	[4 + 2, C] = [6, C]	영구라벨
E	[3 + 4, B] = [7, B]	영구라벨
F	[3 + 3, B] = [6, B]	영구라벨
T	[6 + 4, D] = [10, D] [6 + 4, F] = [10, F] [7 + 3, E] = [10, E]	임시라벨

✓ 1개의 임시라벨 중 마디 T 가 최소거리를 제공함으로 마디 T를 영구마디로 변경

6.5 최단경로 문제

❖ 예제

➢ 단계 8:

✓ 모든 마디가 영구라벨을 갖게 되었으므로 절차를 멈춘다

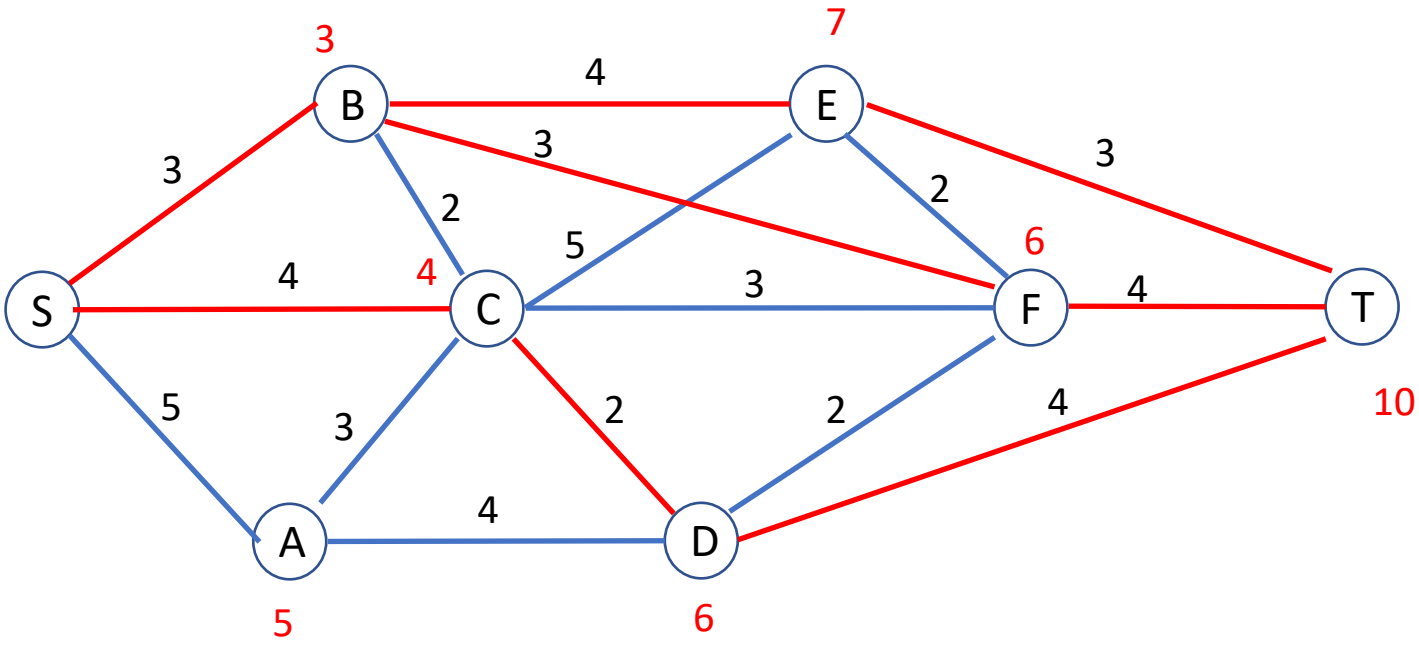
마디	라벨	상태
S	[0, -]	영구라벨
A	[0 + 5, S] = [5, S]	영구라벨
B	[3, S]	영구라벨
C	[0 + 4, S] = [4, S]	영구라벨
D	[4 + 2, C] = [6, C]	영구라벨
E	[3 + 4, B] = [7, B]	영구라벨
F	[3 + 3, B] = [6, B]	영구라벨
T	[6 + 4, D] = [10, D] [6 + 4, F] = [10, F] [7 + 3, E] = [10, E]	영구라벨

✓ T-D-C-S, T-F-B-S, T-E-B-S 가 최단경로

6.5 최단경로 문제

❖ 예제

S 부터 T 까지의 최단경로



6.5 최단경로 문제

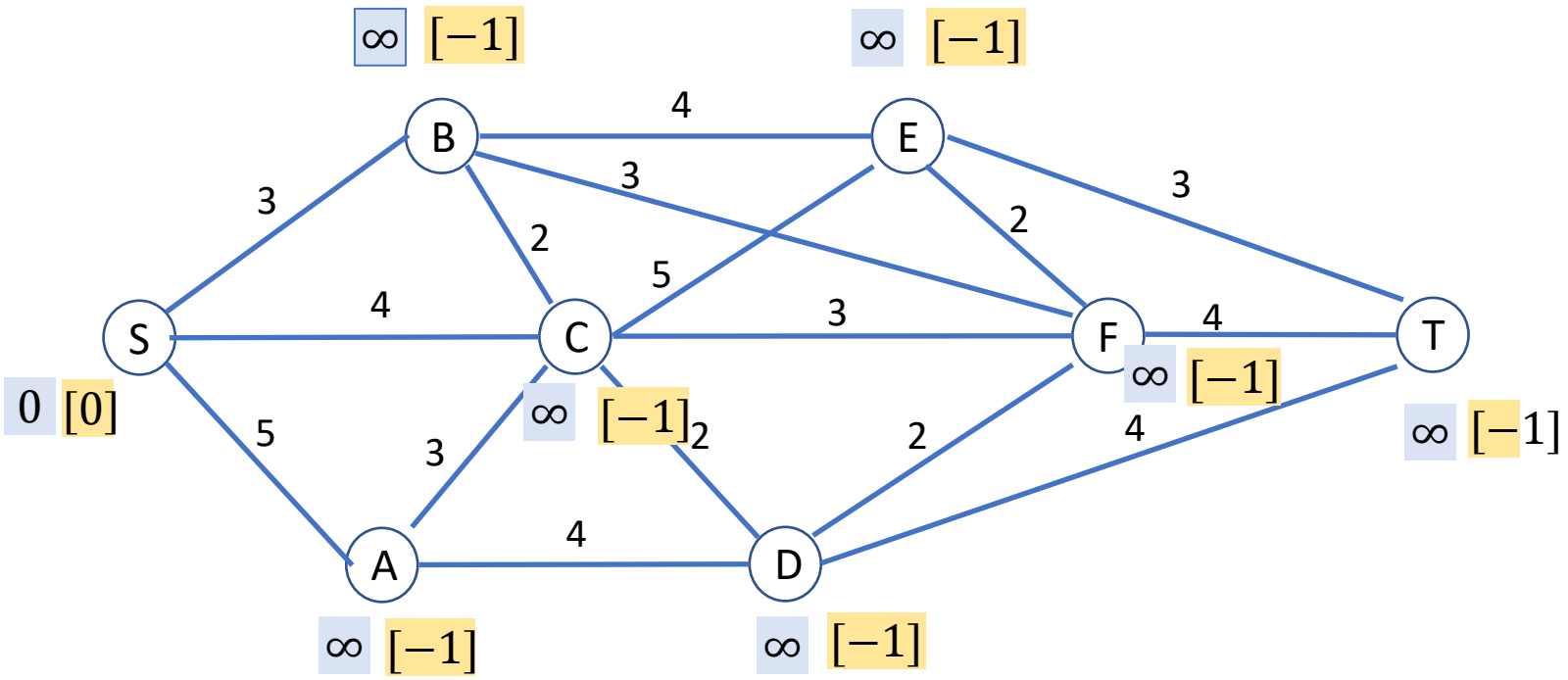
❖ 다익스트라(Dijkstra) 알고리즘

- ➢ Dijkstra 알고리즘은Prim의 MST 알고리즘과 매우 유사
- ➢ 차이점
  - ✓ Dijkstra 알고리즘은 출발점이 주어지지지만 Prim알고리즘에서는 출발점이 주어지지 않는다는 것
  - ✓ Prim 알고리즘에서는 D의 원소에 간선의 가중치가 저장되지만, Dijkstra 알고리즘에서는 D의 원소에 출발점으로부터 각 정점까지의 경로 길이가 저장됨

6.5 최단경로 문제

❖ 예제

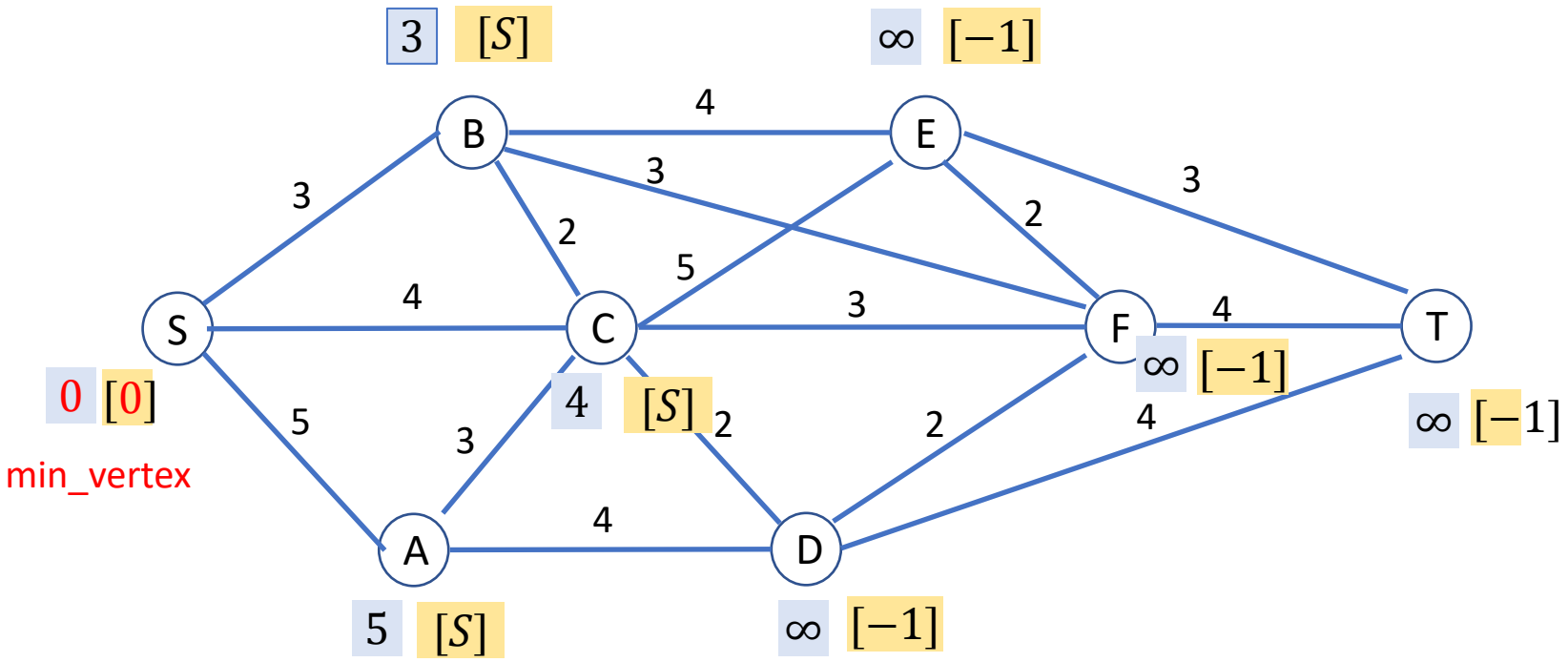
S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 예제

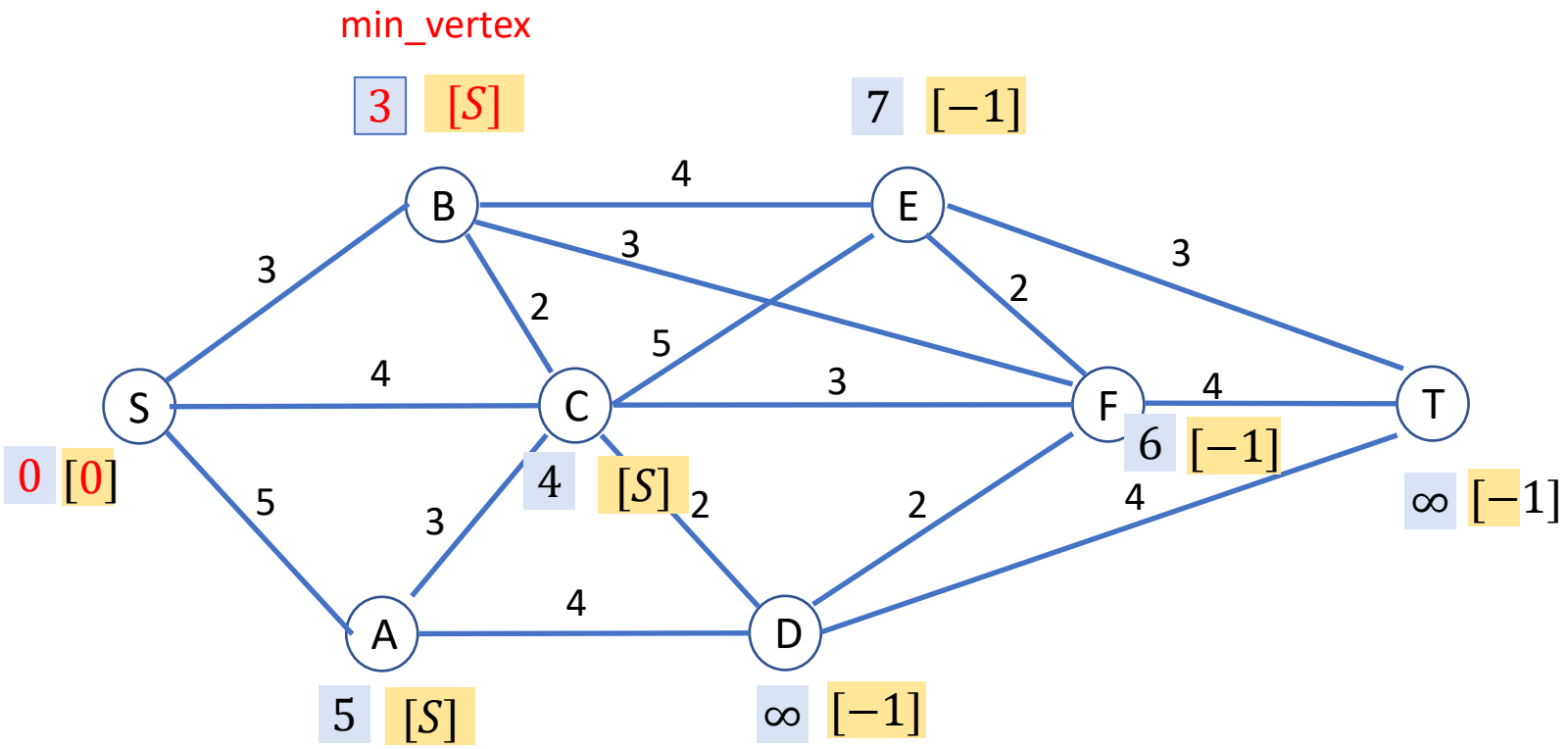
S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 예제

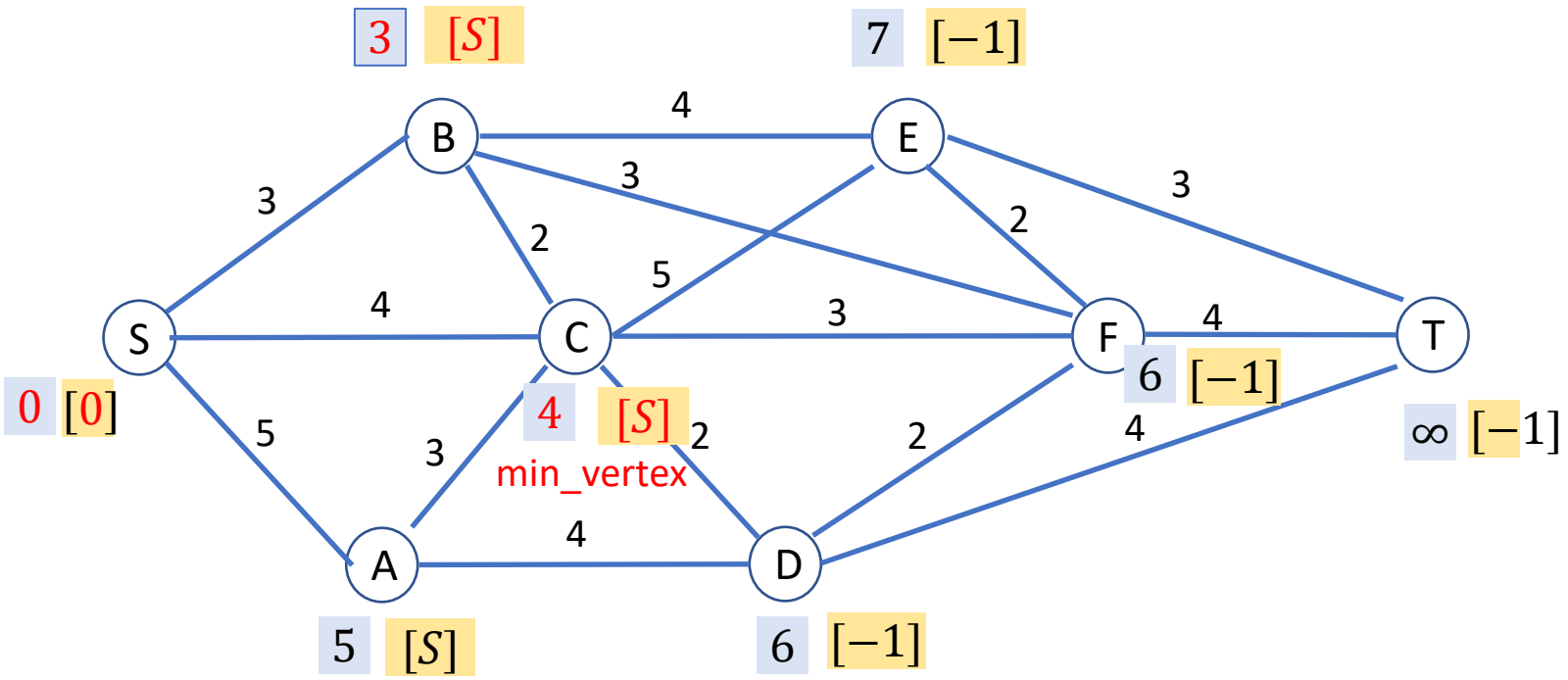
S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 예제

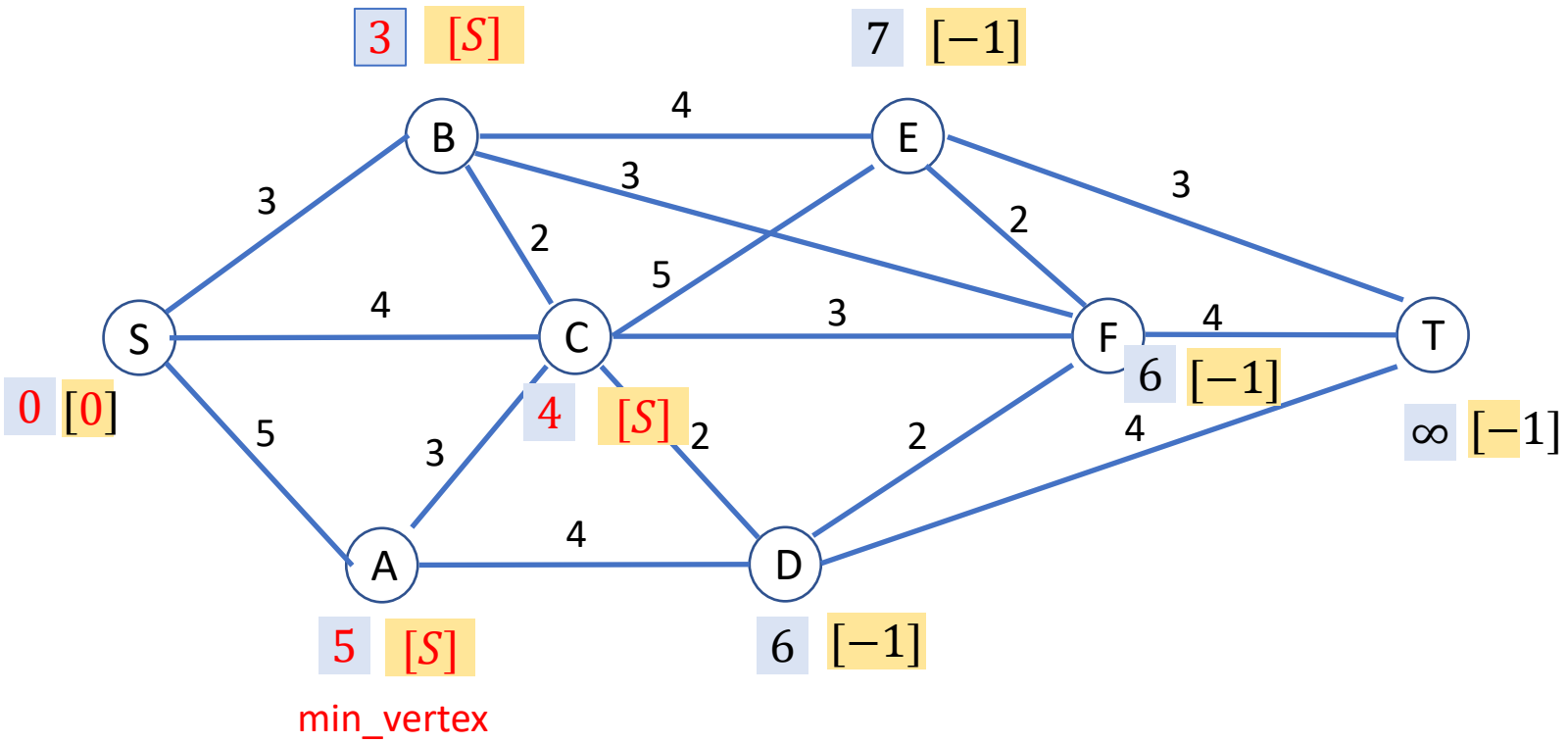
S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 예제

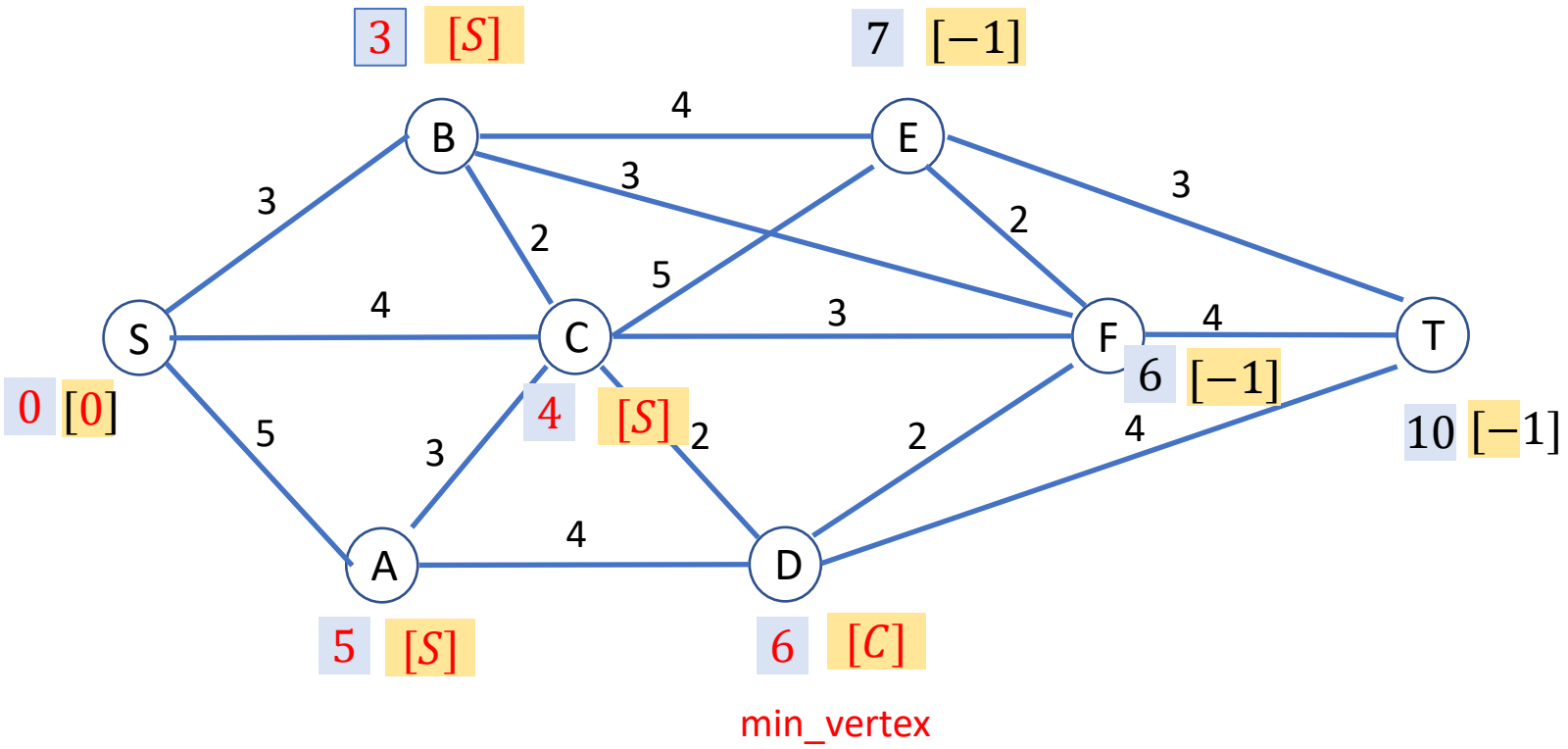
S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 예제

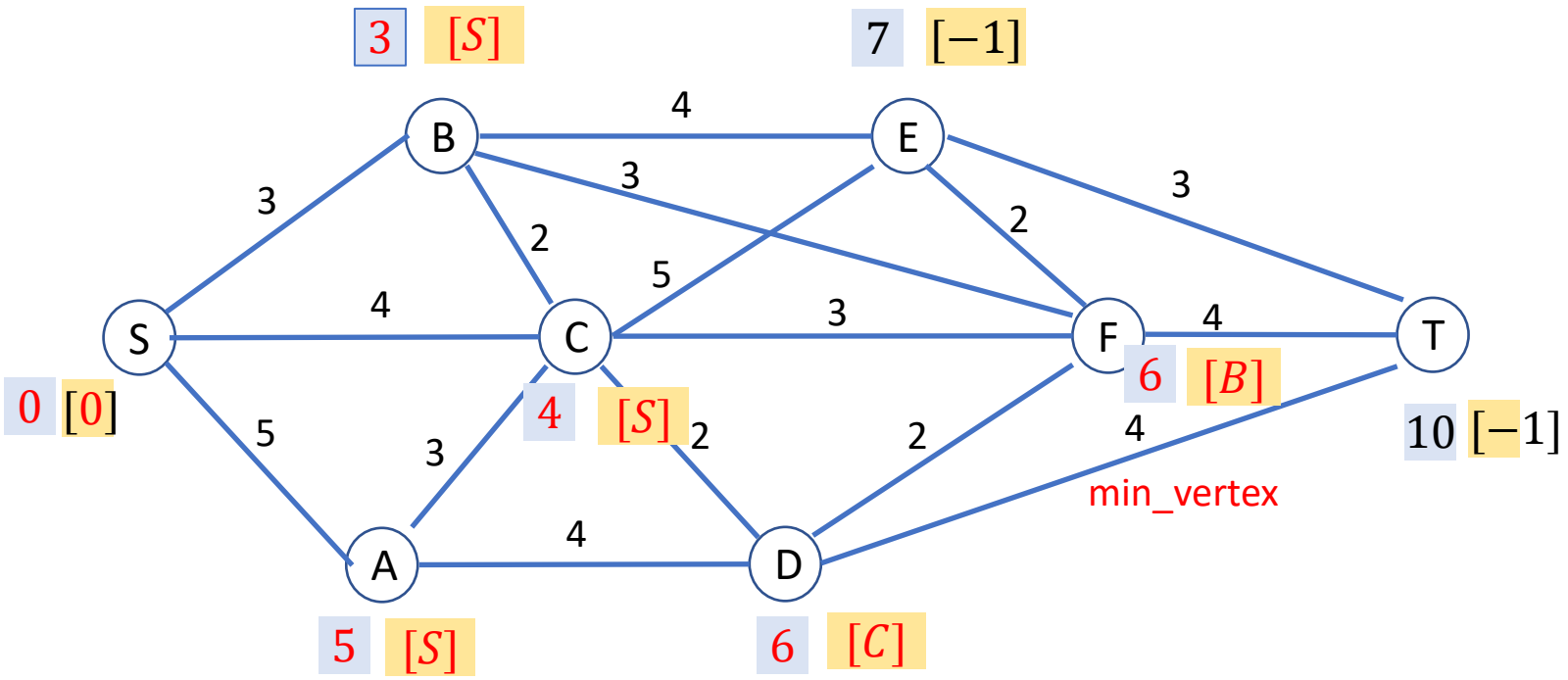
S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 예제

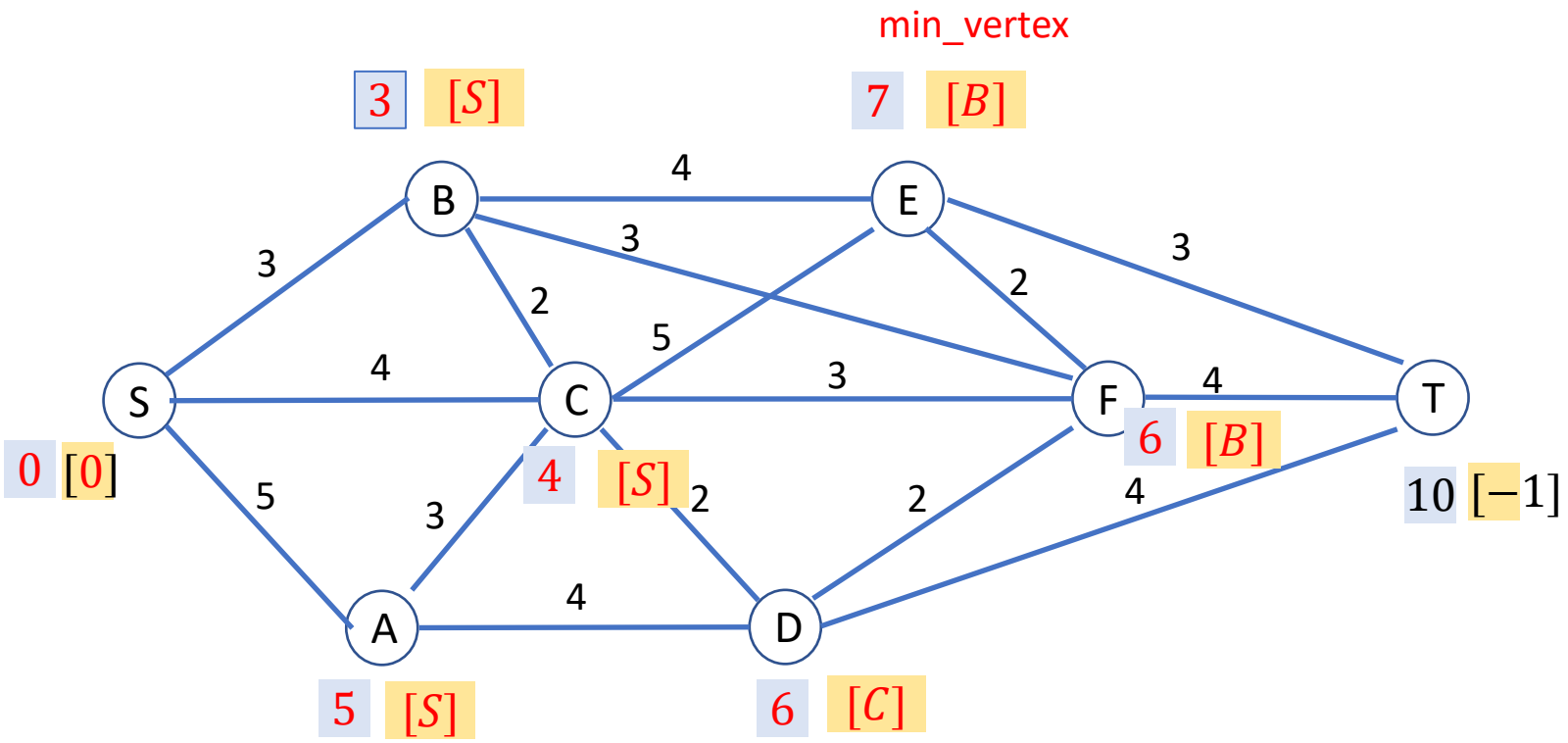
S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 예제

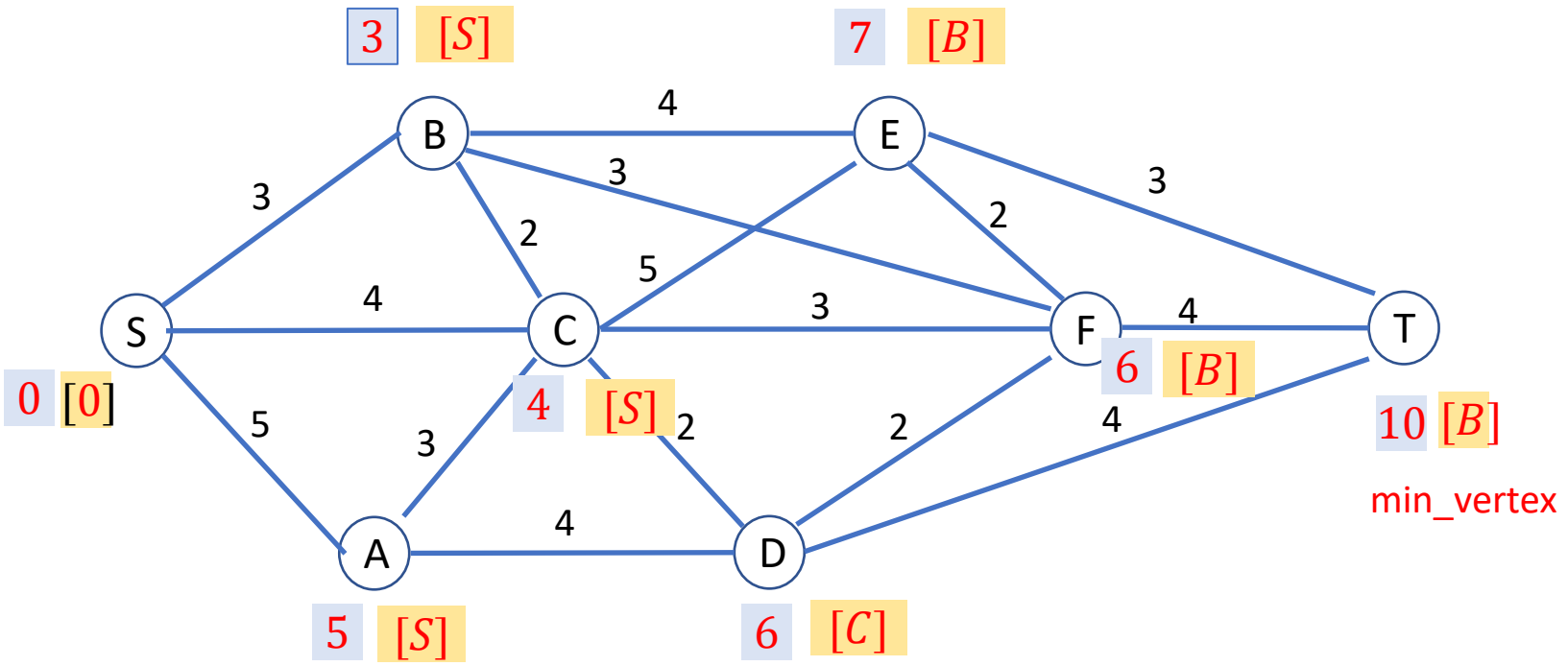
S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 예제

S 부터 T 까지의 최단경로



6.5 최단경로 문제

❖ 수행시간

- Dijkstra 알고리즘은 N번의 반복을 거쳐 min_vertex를 찾고 min_vertex에 인접하면서 방문되지 않은 정점들에 대한 간선완화를 시도
- 이후 D에서 min_vertex를 탐색하는데 O(N) 시간이 소요되고, min_vertex에 인접한 정점들을 검사하여 D의 원소들을 갱신하므로 추가로 O(N) 시간이 소요
- 따라서 총 수행 시간은 N * (O(N) +O(N)) = O(N2)

6.5 최단경로 문제

❖ 최단 경로 (Shortest Path) 문제

➢ 선형계획모형

- ✓ x_{ij} = 호(i, j) 사이의 흐름량
- ✓ c_{ij} = 호(i, j) 의 길이
- ✓ 목적함수: 총 흐름량의 최소화

$$Mininmize \quad \sum_{\{(i,j),i=1,\cdots,n,j=1,\cdots,n\}} c_{ij}x_{ij}$$

- ✓ 제약식: 총 진입흐름량 = 총 진출흐름량

$$\sum_{\{(i,k),i=1,\cdots,n,\}} x_{ik} - \sum_{\{(k,j),i=1,\cdots,n,\}} x_{kj} = \text{순수요량(-1, 0, 1)}$$

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 주어진 네트워크 내의 모든 서로 다른 두 마디 사이의 최단경로

	서울 Seoul	인천 Incheon	수원 Suwon	대전 Daejeon	전주 Jeonju	광주 Gwangju	대구 Daegu	울산 Ulsan	부산 Busan
서 울		40	40	155	230	320	300	410	430
인 천			55	175	250	350	320	450	450
수 원				130	190	300	270	355	390
대 전					95	185	150	260	280
전 주						105	220	330	320
광 주							220	330	270
대 구								110	135
울 산									50
부 산									

➢ 연산방법은 3개의 마디 사이에 비교를 근간으로 산출

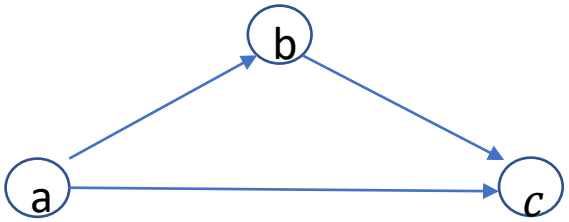
➢ 만일 $d_{kt} + d_{tl} < d_{kl}$ 이 성립된다면 마디 k 로부터 마디 l 까지의 최단거리는 마디 t 를 거쳐 가는 것이다. 이 경우, 두 마디 k 와 l 사이의 최단경로는 $k \rightarrow t \rightarrow l$ 로 대체하는 것이 최적이다

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘 (아이디어)

➢ 작은 그래프에서 부분 문제들을 찾아보자.

✓ 3개의 점이 있는 경우, a에서 c까지의 최단 경로를 찾으려면 2가지 경로, 즉, a에서 c로 직접 가는 경로와 점 b를 경유하는 경로 중에서 짧은 것을 선택



➢ 경유 가능한 점이

✓ 점 1인 경우

✓ 점 1, 2인 경우,

✓ 점 1, 2, 3인 경우

✓ …

✓ 점 1, 2, …, n, 즉 모든 점을 경유 가능한 점들로 고려하면서, 모든 쌍의 최단 경로의 거리를 계산

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘 (동적계획법의 부분문제)

➢ 그래프의 점이 1, 2, 3, …, n일 때

D_{ij}^{k} = 점 {1, 2, …, k}를 경유 가능한 점으로 고려하여, 점 i에서 점 j까지의 모든 경로 중에서 가장 짧은 경로의 거리

✓ [주의] 점 1에서 점 k까지의 모든 점을 반드시 경유하는 경로를 의미하는 것이 아니다.

✓ D_{ij}^{k} 는 {1, 2, …, k}을 하나도 경유하지 않으면서 점 i에서 직접 점 j에 도달하는 간선 (i, j)가 가장 짧은 거리일수도 있음(단, k≠i, k≠j)

✓ k=0인 경우, 점 0은 그래프에 없으므로 어떤 점도 경유하지 않는다는 것을 의미. 즉, D_{ij}^{0} = 간선 (i, j)의 가중치

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 기호정의

- ✓ d_{kt} : 호 (k , l) 의 길이
- ✓ D_t : 반복 t 에서의 각 마디 간 최단거리를 나타내는 거리 행렬
- ✓ S_t : 반복 t 에서의 각 마디 간 최단경로의 직전 마디를 나타내는 순서행렬

➢ 초기 거리행렬과 순서행렬

$$\begin{bmatrix} - & d_{12} & \cdots & d_{1n} \\ d_{21} & - & & d_{2n} \\ \vdots & \ddots & & \vdots \\ d_{n1} & \cdots & & - \end{bmatrix} \qquad \begin{bmatrix} - & 2 & \cdots & n \\ 1 & - & & n \\ \vdots & \ddots & & \vdots \\ 1 & \cdots & & - \end{bmatrix}$$

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

- 단계 0: 초기 거리행렬 D_0 과 순서행렬 S_0 를 정의한다. $t = 1$ 로 놓는다.
- 단계 t : $t > n$ 이면, 이 절차를 멈춘다.

  아니면, 피봇행과 피봇열로써 t 번째 행과 열을 선택

  거리행렬 D_t 내의 모든 k 와 l 에 대해 각 요소 에 대해 삼각연산을 적용

  만일 $d_{kt} + d_{tl} < d_{kl}\ (k \neq t \neq l)$이 성립된다면, 다음을 수행

  (1) 반복 $t-1$ 로부터의 거리행렬 D_{t-1} 에서 d_{kl} 을 $d_{kt} + d_{tl}$ 로 대체하는 것에 의해 새로운 거리행렬 D_t 를 생성

  (2) 반복 $t-1$ 로부터의 순서행렬 S_{t-1} 에서 s_{kl} 을 t 로 대체하는 것에 의해 새로운 순서행렬 S_t 를 생성. $t = t + 1$ 로 놓고 단계 t 를 반복

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

- ➢ 입력: 2차원 배열 D, 단, D[i, j] = 간선 (i, j)의 가중치, 만일 간선 (i, j)가 없으면

  D[i, j] = ∞, 모든 i에 대하여 D[i, i] = 0

- ➢ 출력: 모든 쌍 최단 경로의 거리를 저장한 2차원 배열 D

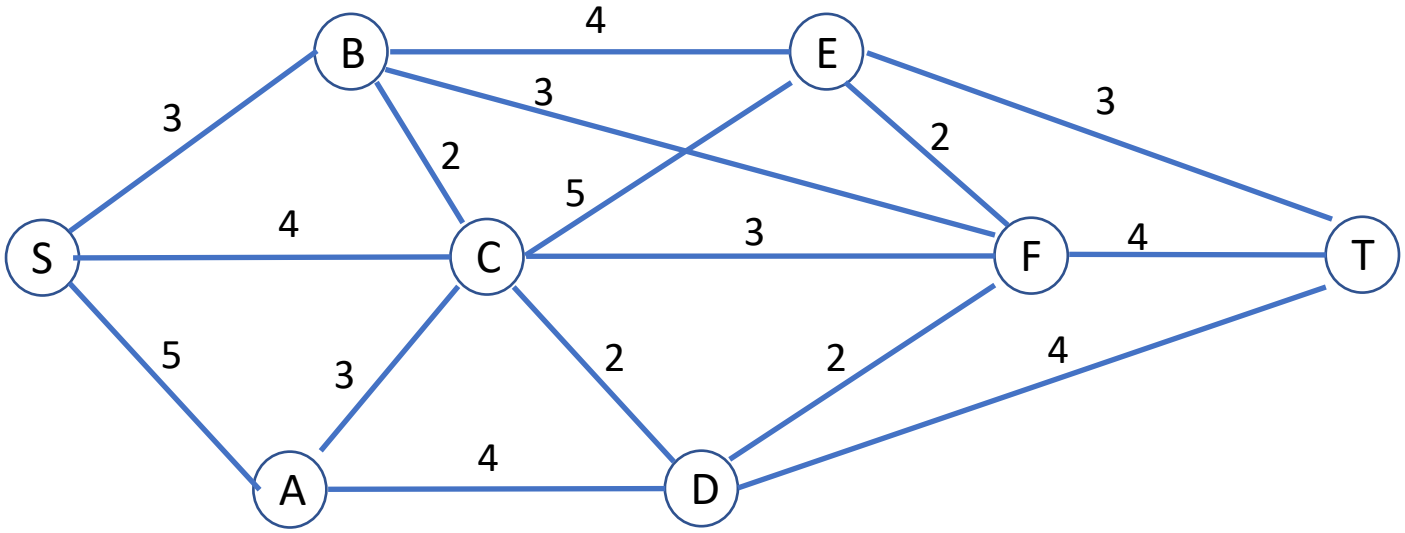
```
1. for k = 1 to n
2.     for i = 1 to n (i≠k)
3.         for j = 1 to n (j≠k, j≠i)
4.             D[i, j] = min{ D[i, k]+D[k, j], D[i, j] }
```

- ➢ 시간복잡도
  - ✓ 각 k에 대해서 모든 i, j 쌍에 대해 계산되므로, 총 n x n x n = n3회 계산이 이루어지고, 각 계산은 O(1) 시간 소요
  - ✓ 시간 복잡도는 $O(n^3)$

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 예제



6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 단계 0 :

$D_0 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞	∞	∞	∞
2	5	-	∞	3	4	∞	∞	∞
3	3	∞	-	2	∞	4	3	∞
4	4	3	2	-	2	5	3	∞
5	∞	4	∞	2	-	∞	2	4
6	∞	∞	4	5	∞	-	2	3
7	∞	∞	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

$S_0 =$

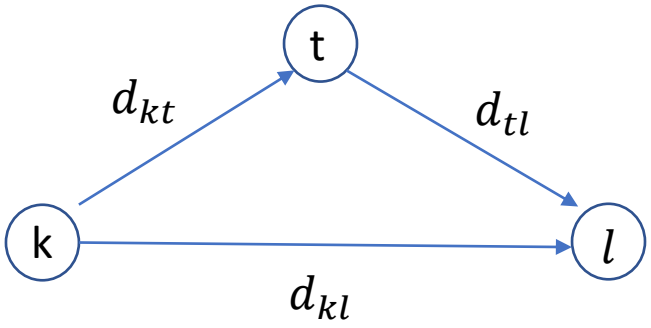
	1	2	3	4	5	6	7	8
1	-	A	B	C	D	E	F	T
2	S	-	B	C	D	E	F	T
3	S	A	-	C	D	E	F	T
4	S	A	B	-	D	E	F	T
5	S	A	B	C	-	E	F	T
6	S	A	B	C	D	-	F	T
7	S	A	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

❖ Floyd-Warshall 알고리즘

➢ 단계 1 :

✓ 거리행렬 D_0 로부터 피봇행과 피봇열로써 첫 번째 행과 열을 선택하고, 거리행렬 내의 모든 k 와 l 에 대해 삼각연산을 적용

✓ $t = 1$ 이므로 $d_{k1} + d_{1l}$ (단, $k \neq t \neq l$) 을 계산한 후 각 요소 d_{kl} 과 비교. 이 계산 결과를 이용하여 아래와 같은 조치를 취하고, 새로운 거리행렬 D_1 과 순서행렬 S_1 을 생성



6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 단계 1 :

$D_1 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞	∞	∞	∞
2	5	-	∞ 8	3	4	∞	∞	∞
3	3	∞8	-	2	∞	4	3	∞
4	4	3	2	-	2	5	3	∞
5	∞	4	∞	2	-	∞	2	4
6	∞	∞	4	5	∞	-	2	3
7	∞	∞	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

$S_1 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	D	E	F	T
2	S	-	S	C	D	E	F	T
3	S	S	-	C	D	E	F	T
4	S	A	B	-	D	E	F	T
5	S	A	B	C	-	E	F	T
6	S	A	B	C	D	-	F	T
7	S	A	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 단계 2 :

$D_2 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9	∞	∞	∞
2	5	-	∞ 8	3	4	∞	∞	∞
3	3	∞8	-	2	∞12	4	3	∞
4	4	3	2	-	2	5	3	∞
5	∞9	4	∞12	2	-	∞	2	4
6	∞	∞	4	5	∞	-	2	3
7	∞	∞	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

$S_2 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	A	E	F	T
2	S	-	S	C	D	E	F	T
3	S	S	-	C	A	E	F	T
4	S	A	B	-	D	E	F	T
5	A	A	A	C	-	E	F	T
6	S	A	B	C	D	-	F	T
7	S	A	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 단계 3 :

$D_3 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9	∞7	∞6	∞
2	5	-	∞ 8	3	4	∞12	∞11	∞
3	3	∞8	-	2	∞12	4	3	∞
4	4	3	2	-	2	5	3	∞
5	∞9	4	∞12	2	-	∞	2	4
6	∞7	∞12	4	5	∞	-	2	3
7	∞6	∞11	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

$S_3 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	A	B	B	T
2	S	-	S	C	D	B	B	T
3	S	S	-	B	A	E	F	T
4	S	A	B	-	D	E	F	T
5	A	A	A	C	-	E	F	T
6	B	B	B	C	D	-	F	T
7	B	B	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 단계 4 :

$D_4 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞96	∞7	∞6	∞
2	5	-	∞ 8	3	4	∞128	∞116	∞
3	3	∞8	-	2	∞124	4	3	∞
4	4	3	2	-	2	5	3	∞
5	∞96	4	∞124	2	-	∞7	2	4
6	∞7	∞128	4	5	∞7	-	2	3
7	∞6	∞116	3	3	2	2	-	4
8	∞	∞	∞	∞	4	3	4	-

$S_4 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	T
2	S	-	S	C	D	C	C	T
3	S	S	-	C	C	E	F	T
4	S	A	B	-	D	E	F	T
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	S	A	B	C	D	E	F	-

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 단계 5 :

$D_5 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9 6	∞7	∞6	∞10
2	5	-	∞ 8	3	4	∞12 8	∞11 6	∞8
3	3	∞8	-	2	∞12 4	4	3	∞8
4	4	3	2	-	2	5	3	∞6
5	∞9 6	4	∞12 4	2	-	∞7	2	4
6	∞7	∞12 8	4	5	∞7	-	2	3
7	∞6	∞11 6	3	3	2	2	-	4
8	∞10	∞8	∞8	∞ 6	4	3	4	-

$S_5 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	D
2	S	-	S	C	D	C	C	D
3	S	S	-	C	C	E	F	D
4	S	A	B	-	D	E	F	D
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	D	D	D	D	D	E	F	-

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 단계 6 :

$D_6 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞9 6	∞7	∞6	∞10
2	5	-	∞ 8	3	4	∞12 8	∞11 6	∞8
3	3	∞8	-	2	∞12 4	4	3	∞8 7
4	4	3	2	-	2	5	3	∞6
5	∞9 6	4	∞12 4	2	-	∞7	2	4
6	∞7	∞12 8	4	5	∞7	-	2	3
7	∞6	∞11 6	3	3	2	2	-	4
8	∞10	∞8	∞8 7	∞ 6	4	3	4	-

$S_6 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	D
2	S	-	S	C	D	C	C	D
3	S	S	-	C	C	E	F	E
4	S	A	B	-	D	E	F	D
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	D	D	E	D	D	E	F	-

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 단계 7 :

$D_7 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞96	∞7	∞6	∞10
2	5	-	∞ 8	3	4	∞128	∞116	∞8
3	3	∞8	-	2	∞124	4	3	∞87
4	4	3	2	-	2	5	3	∞6
5	∞96	4	∞124	2	-	∞7	2	4
6	∞7	∞128	4	5	∞7	-	2	3
7	∞6	∞116	3	3	2	2	-	4
8	∞10	∞8	∞87	∞6	4	3	4	-

$S_7 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	D
2	S	-	S	C	D	C	C	D
3	S	S	-	C	C	E	F	E
4	S	A	B	-	D	E	F	D
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	D	D	E	D	D	E	F	-

6.5 최단경로 문제

❖ Floyd-Warshall 알고리즘

➢ 단계 8 :

$D_8 =$

	1	2	3	4	5	6	7	8
1	-	5	3	4	∞96	∞7	∞6	∞10
2	5	-	∞ 8	3	4	∞128	∞116	∞8
3	3	∞8	-	2	∞124	4	3	∞87
4	4	3	2	-	2	5	3	∞6
5	∞96	4	∞124	2	-	∞7	2	4
6	∞7	∞128	4	5	∞7	-	2	3
7	∞6	∞116	3	3	2	2	-	4
8	∞10	∞8	∞87	∞6	4	3	4	-

$S_8 =$

	1	2	3	4	5	6	7	8
1	-	A	B	C	C	B	B	D
2	S	-	S	C	D	C	C	D
3	S	S	-	C	C	E	F	E
4	S	A	B	-	D	E	F	D
5	C	A	C	C	-	C	F	T
6	B	C	B	C	C	-	F	T
7	B	C	B	C	D	E	-	T
8	D	D	E	D	D	E	F	-

6.6 최대흐름문제

❖ 최대흐름문제

- ➢ 시작마디에서 종료마디까지 최대한 보낼 수 있는 용량을 찾는 문제
  - ✓ 상수원을 제공하는 여러 댐으로부터 여러 도시 내의 상수도 처리장까지를 연결하는 파이프라인 네트워크
  - ✓ 댐에서 상수도 처리장까지 물을 옮기기 위한 펌프장 들이 중간에 위치
  - ✓ 댐과 펌프장 간, 펌프장과 상수도 처리장 간을 연결하는 파이프는 파이프 용량의 제약으로 인해 물의 흐름에 제약을 받게 될
  - ✓ 댐과 상수도 처리장 사이에 최대한 보낼 수 있는 물의 양에 관심

6.6 최대흐름문제

❖ 최대흐름문제

- 시작마디부터 종료마디 사이에서 양의 흐름을 갖는 경로를 우선적으로 찾는 것에서 출발(이하 '기준경로'라 함)
- 기준경로를 구성하는 각 호들의 잔여흐름용량들 중 최소 흐름량이 그 경로의 최대 흐름용량이 됨
- 이러한 흐름용량들을 더함으로써 주어진 네트워크의 총흐름량을 계산

❖ 기호 정의

- C_{ij}^R = 마디 i 에서 마디 j 로의$(i \rightarrow j)$ 사용가능한 잔여흐름용량
- C_{ji}^R = 마디 j 에서 마디 i 로의$(j \rightarrow i)$ 사용가능한 잔여흐름용량

6.6 최대흐름문제

❖ 알고리즘

➢ $[f_j, i]$: i 마디 로부터 마디 j에 흐름이 발생, f_j 는 마디 i 로부터 마디 j 까지의 흐름량

➢ 단계 1 : 모든 호(i, j)에 대해, 잔여흐름용량을 초기 흐름용량으로 놓는다.

$f_j = \infty$, 시작마디 1에 라벨 $[\infty, -]$ 를 붙이고 [단계 2]로 간다.

➢ 단계 2 : (기준경로의 탐색) 마디 i 로부터 양의 잔여흐름용량을 가지고 있으면서 마디 j 로 직접 연결되는 라벨이 붙지 않은 마디들의 집합을 찾고, 이를 UN_i로 놓는다 (즉, 모든 $j \in UN_i$ 에 대해 $C_{ij}^R > 0$). 만일 $UN_i \neq \emptyset$ 라면 [단계 3]으로 간다. 아니면 [단계 4]로 간다.

6.6 최대흐름문제

❖ 알고리즘

➢ 단계 3 : 다음을 만족하는 마디 $k \in UN_i$ 를 찾는다.

$$C_{ik}^R = \max\{ C_{ij}^R, j \in UN_i \}$$

$f_k = C_{ik}^R$ 로 놓고 마디에 라벨 $[f_k, i]$ 를 붙인다. 만일 $k = n$ 이면 종료마디까지 라벨이 붙여졌으므로 기준경로가 탐색되었다. [단계 5]로 간다.

아니면, $i = k$ 로 놓고 [단계 2]로 간다.

➢ 단계 4 : (역추적) 만일 $i = 1$ 이면 더 이상의 기준경로는 존재하지 않는다.

[단계 6]으로 간다. 아니면, 현행 마디 i 직전에 라벨이 붙여진 마디 b 를 선택하고 현 반복작업에서는 마디 i 를 고려대상에서 제거된다.

$i = b$ 로 놓고 [단계 2]로 간다.

6.6 최대흐름문제

❖ 알고리즘

➢ 단계 5 : (수정네트워크의 결정)

$N_m = (1, n_1, n_2, \cdots, n)$ 을 시작마디 1부터 종료마디 n 까지의 m 번째 기준경로를 구성하는 마디들의 순서를 나타내는 집합이라 정의한다.

그 기준경로에 대응되는 최대 흐름량은 다음과 같이 계산된다

$$F_m = \min(f_1, f_{n_1}, f_{n_2}, \cdots, f_n)$$

기준경로를 따르는 각 호들의 잔여흐름용량은 다음과 같이 수정된다

(1) 흐름방향 $i \rightarrow j$ 에 대해, $(C_{ij}^R - F_m, C_{ji}^R + F_m)$

(2) 흐름방향 $j \rightarrow i$ 에 대해, $(C_{ij}^R + F_m, C_{ji}^R - F_m)$

[단계 4]에서 제거되었던 마디들은 다시 복원시킨다.

[단계 1]로 되돌아가 새로운 기준경로를 찾는다.

6.6 최대흐름문제

❖ 알고리즘

➢ 단계 6 : (해의 탐색)

(1) l 개의 기준경로가 탐색되었다면 다음과 같이 최대 흐름량을 결정한다.

$$F_m = F_1 + \cdots + F_l$$

(2) 호 (i, j) 에 대한 초기 및 최종 잔여흐름용량을 각각 (C_{ij}^0, C_{ji}^0)과 (C_{ij}^R, C_{ji}^R) 로 정의하자. 호 (i, j) 에 대한 최적 흐름량은 다음과 같이 계산된다.

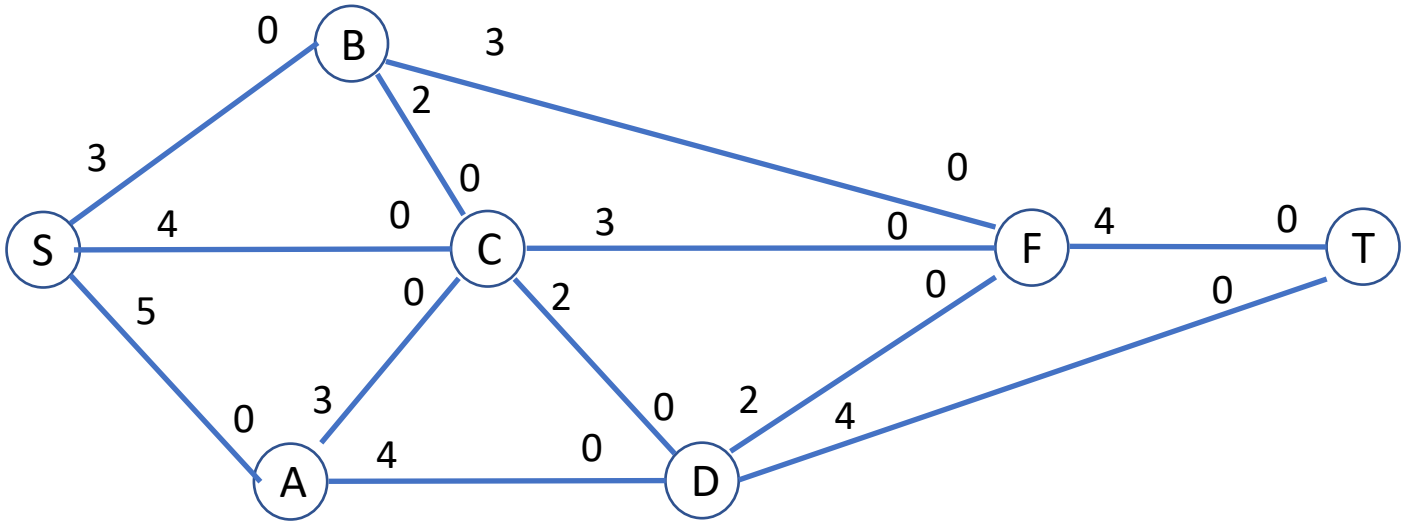
$$\left(x_{ij}^*, x_{ji}^*\right) = (C_{ij}^0 - C_{ij}^R, C_{ji}^0 - C_{ji}^R)$$

여기서, $\left(x_{ij}^*, x_{ji}^*\right)$ 는 각각 흐름방향에 대한 최적 흐름량을 나타낸다

6.6 최대흐름문제

❖ 예제

➢ 가지위의 숫자는 도시간 정보전달량일때, s 에서 t 까지의 최대 전달 정보량?



6.6 최대흐름문제

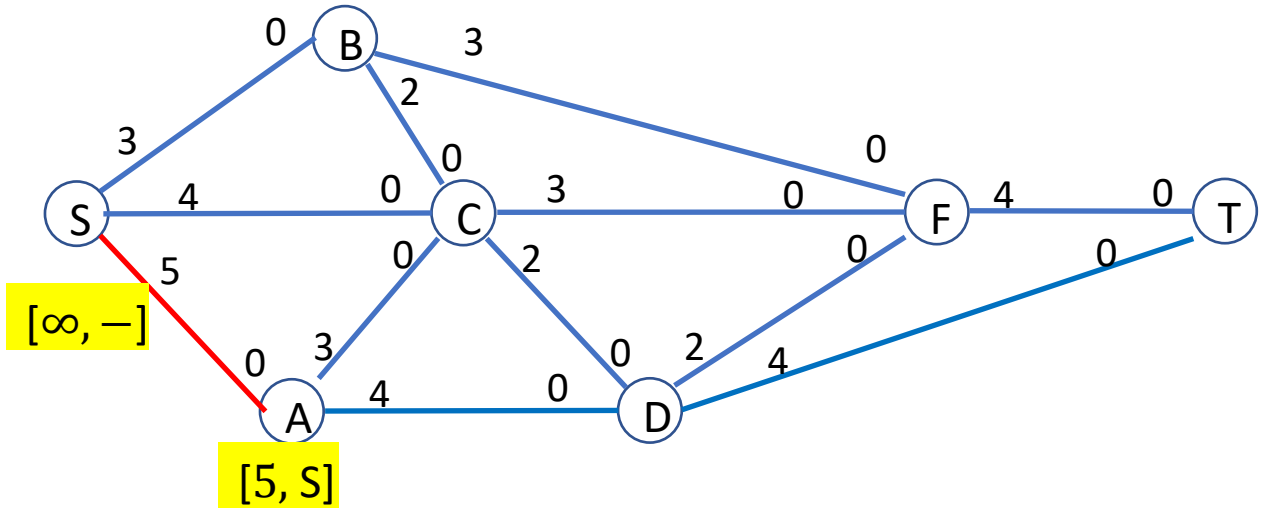
❖ 예제

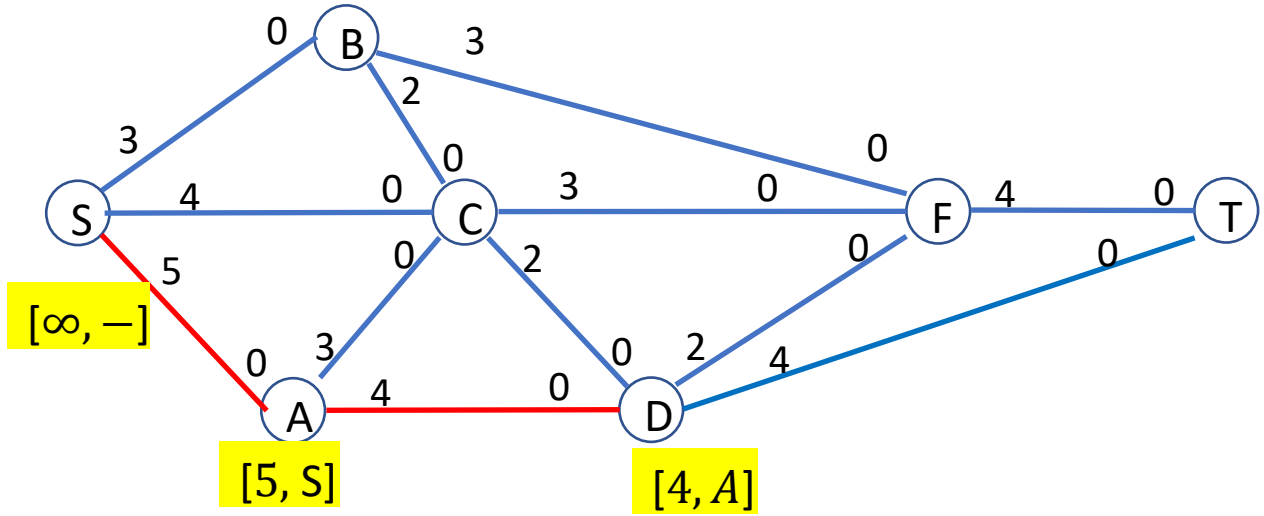
- 단계 1: $f_S = \infty$, 시작마디 S에 라벨 $[\infty, -], i = S,$ [단계 2]로 간다.
- 단계 2: 마디 S 로부터 직접 연결되는 마디들 중 양의 잔여용량흐름을 갖는 마디들의 집합 $UN_S \in$ {A, B, C}를 결정. $UN_S \neq \emptyset$ 이므로 [단계 3]으로 간다.
- 단계 3: $\max\{ C_{SA}^R, C_{SB}^R, C_{SC}^R\}$ = max{5, 3, 4|=5 를 제공하는 집합내의 마디는 k= A. $f_A = 5$ 로 놓고 마디 A에 라벨 [5, S] 을 붙인다. $i = A$ 로 놓고 [단계 2]로 간다.
- 단계 2: 마디 A 로부터 직접 연결되는 마디들 중 양의 잔여용량흐름을 갖는 마디들의 집합 $UN_A \in$ {C, D} 를 결정. $UN_A \neq \emptyset$ 이므로 [단계 3]으로 간다.
- 단계 3: : $\max\{ C_{AC}^R, C_{AD}^R\}$ = max{3, 4|=4 를 제공하는 집합내의 마디는 k = D. $f_D = 4$ 로 놓고 마디 D에 라벨 [5, A] 을 붙인다. $i = D$ 로 놓고 [단계 2]로 간다.
- 단계 2: 마디 D 로부터 직접 연결되는 마디들 중 양의 잔여용량흐름을 갖는 마디들의 집합 $UN_D \in$ {F, T} 를 결정. $UN_D \neq \emptyset$ 이므로 [단계 3]으로 간다.
- 단계 3: $\max\{ C_{DF}^R, C_{DT}^R\}$ = max{2, 4|=4 를 제공하는 집합내의 마디는 $k = T$. $f_T = 4$ 로 놓고 마디 T에 라벨 [4, D] 을 붙인다. $k = T$이므로 [단계 5]로 간다.
- 단계 5: $N_1 = \{S, A, D, T\}, F_1 = min\{\infty, 5, 4, 4\} = 4,$ 잔여용량 수정

6.6 최대흐름문제

❖ 예제

➢반복1

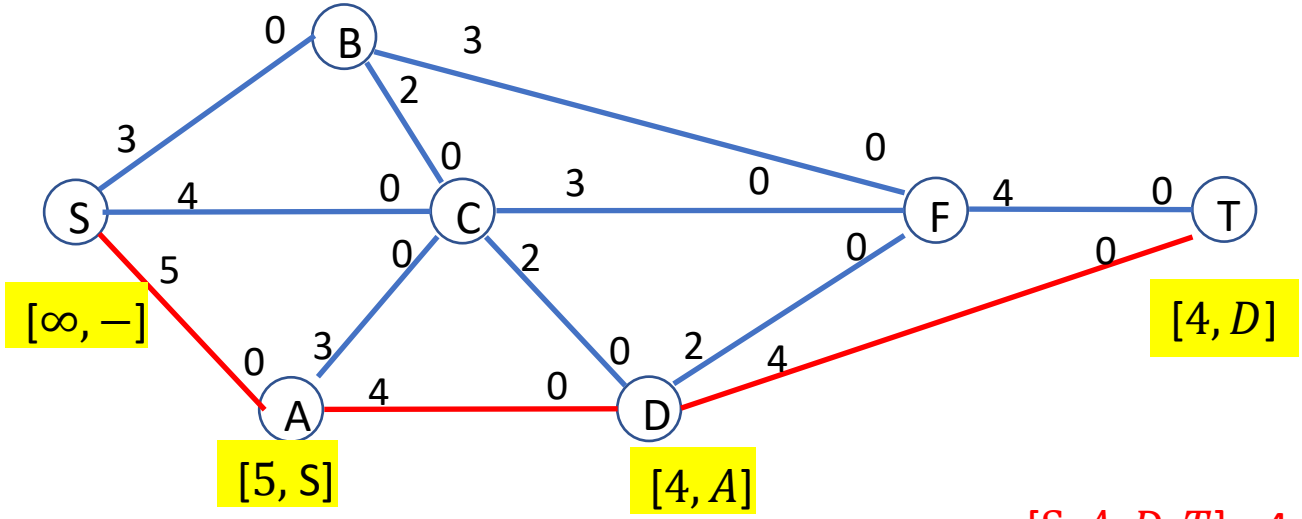




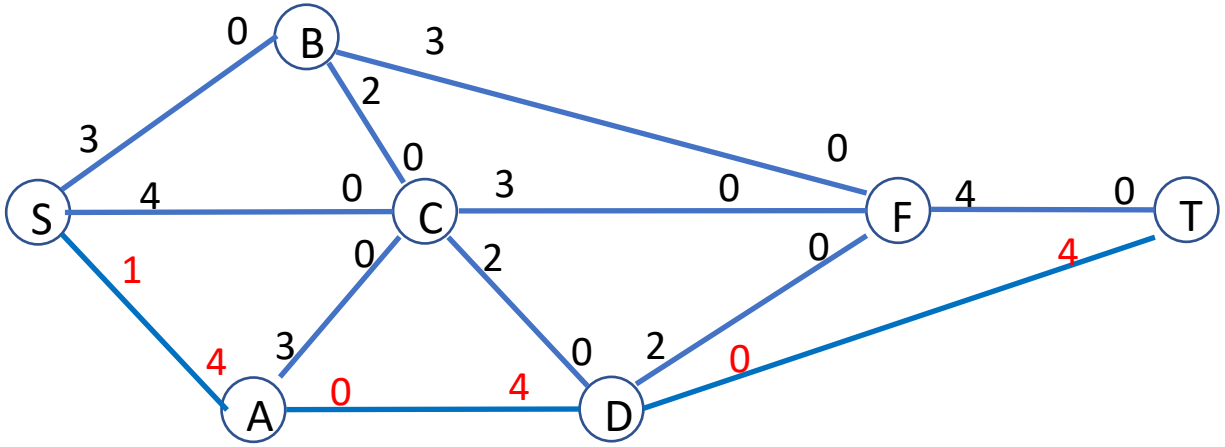
6.6 최대흐름문제

❖ 예제

➢반복1



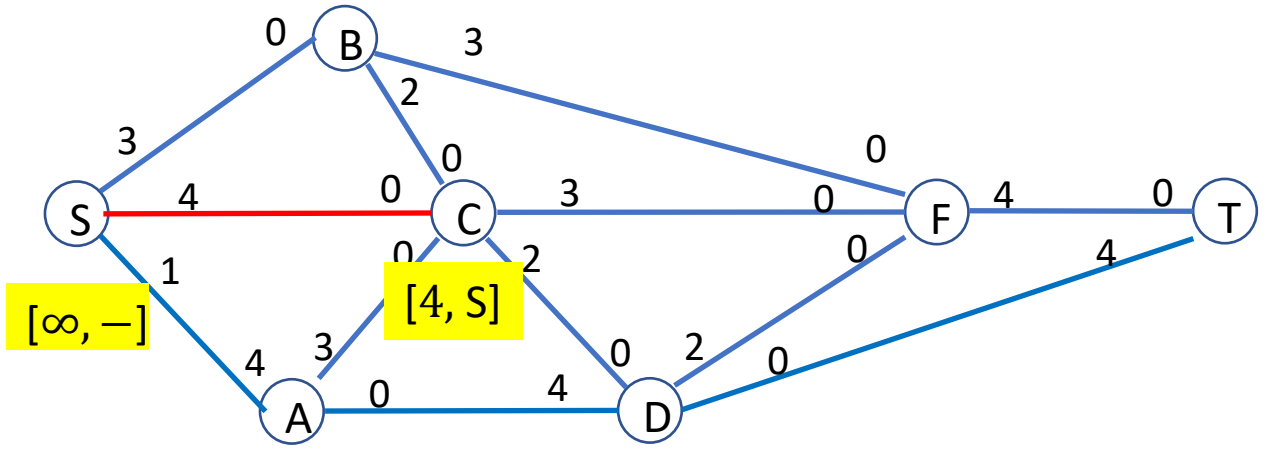
[S, A, D, T] - 4

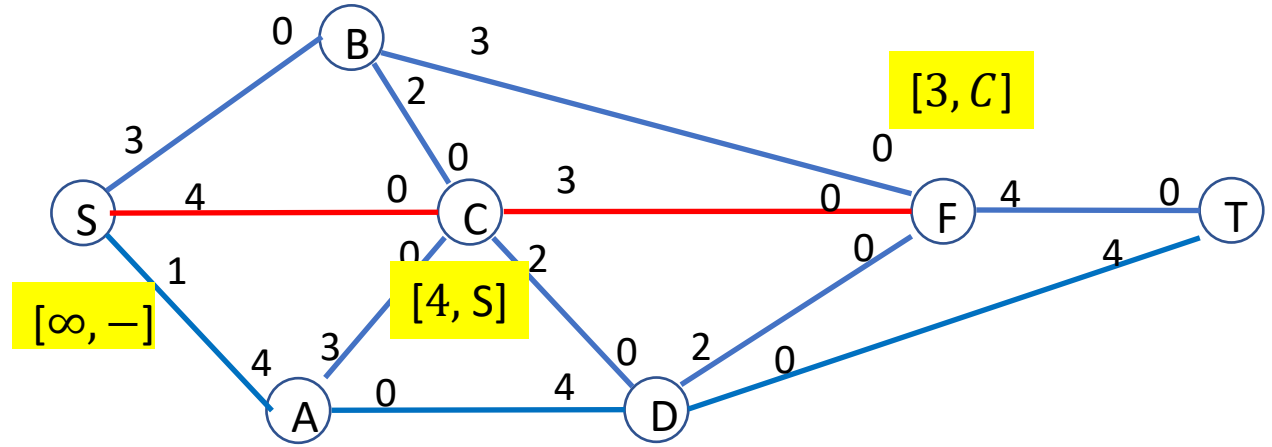


6.6 최대흐름문제

❖ 예제

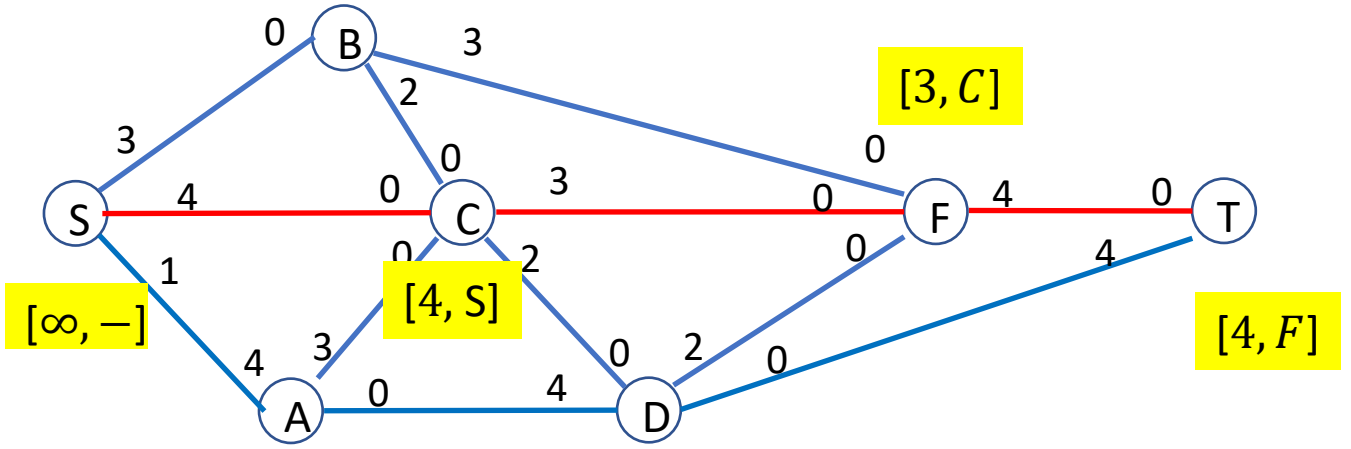
➢반복2





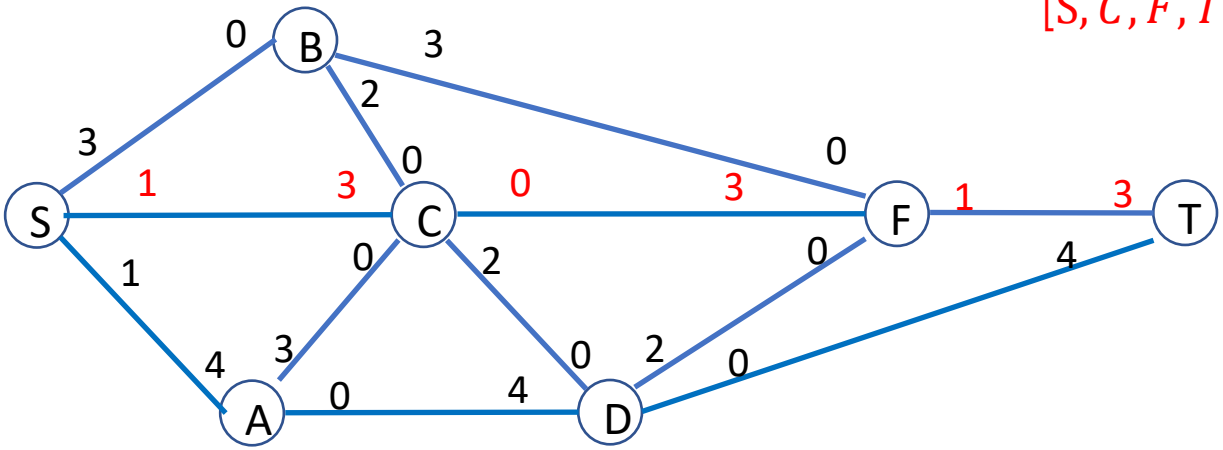
❖ 예제

➢반복2



$[S, A, D, T] - 4$

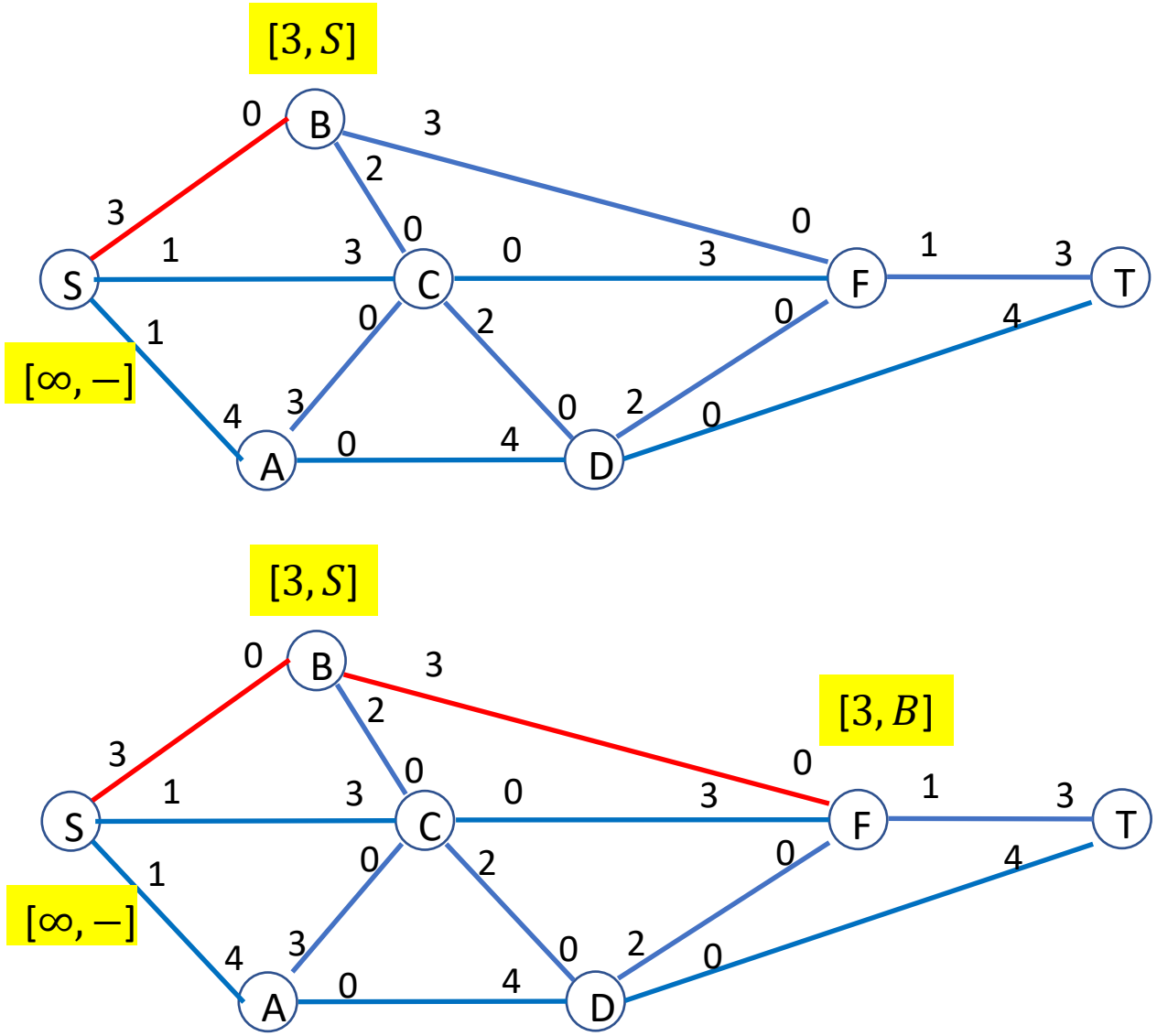
$[S, C, F, T] - 3$



6.6 최대흐름문제

❖ 예제

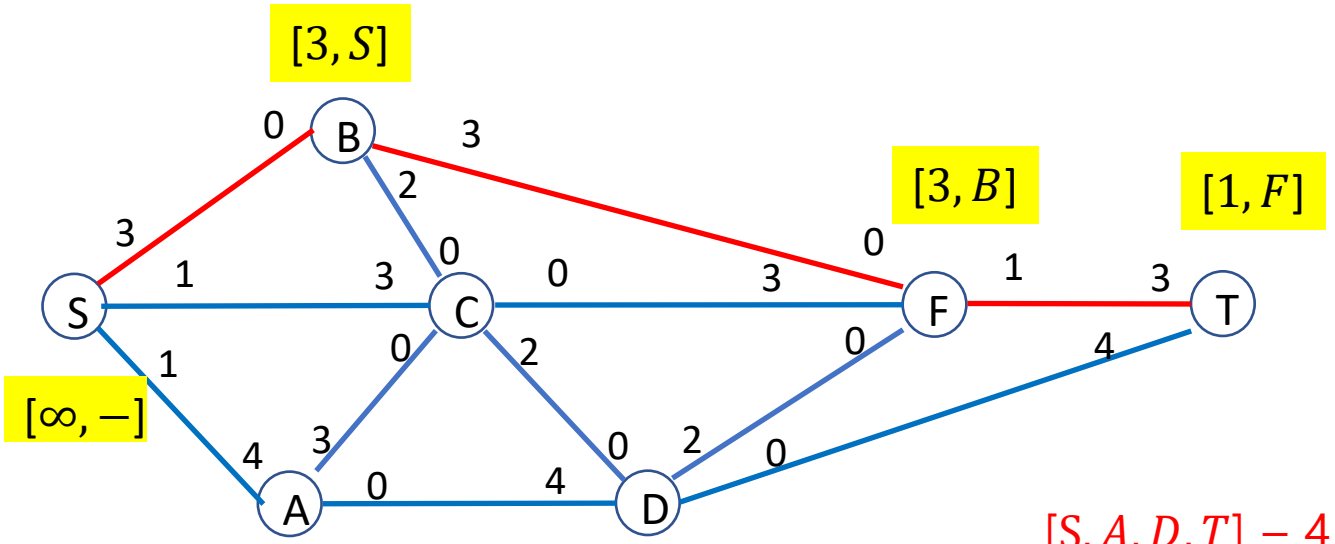
➢반복3



6.6 최대흐름문제

❖ 예제

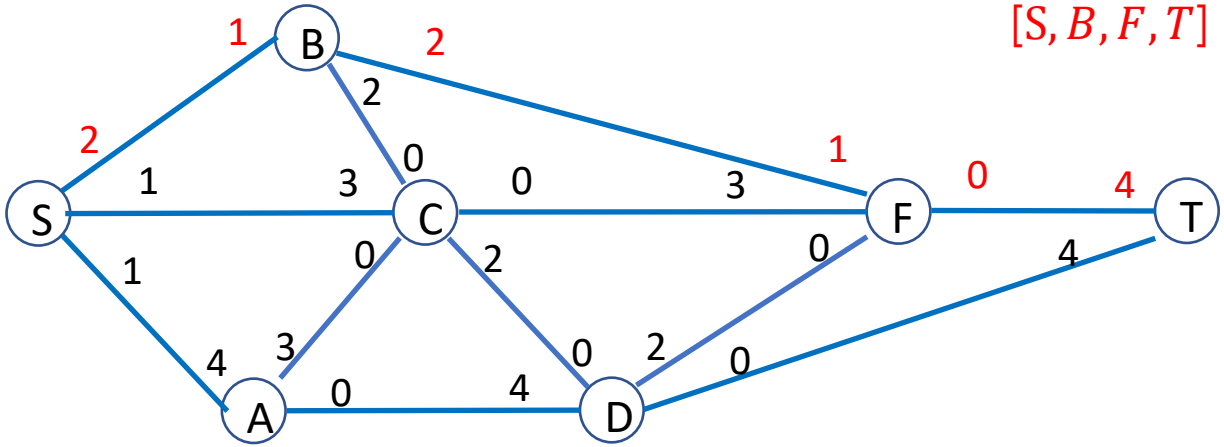
➢반복3



$[S, A, D, T] - 4$

$[S, C, F, T] - 3$

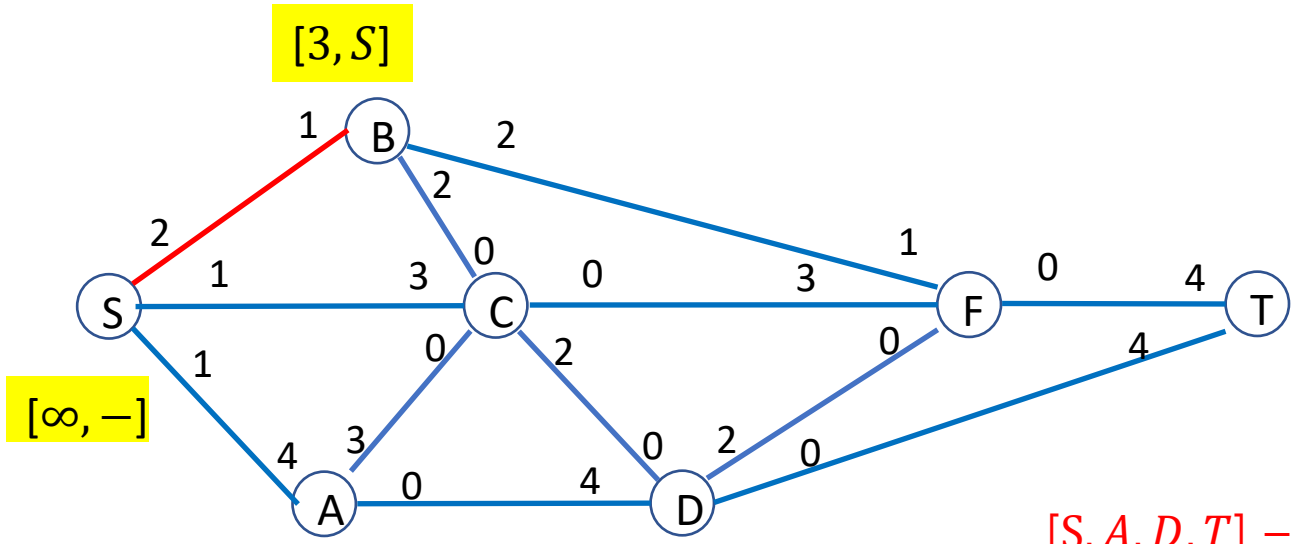
$[S, B, F, T] - 1$



6.6 최대흐름문제

❖ 예제

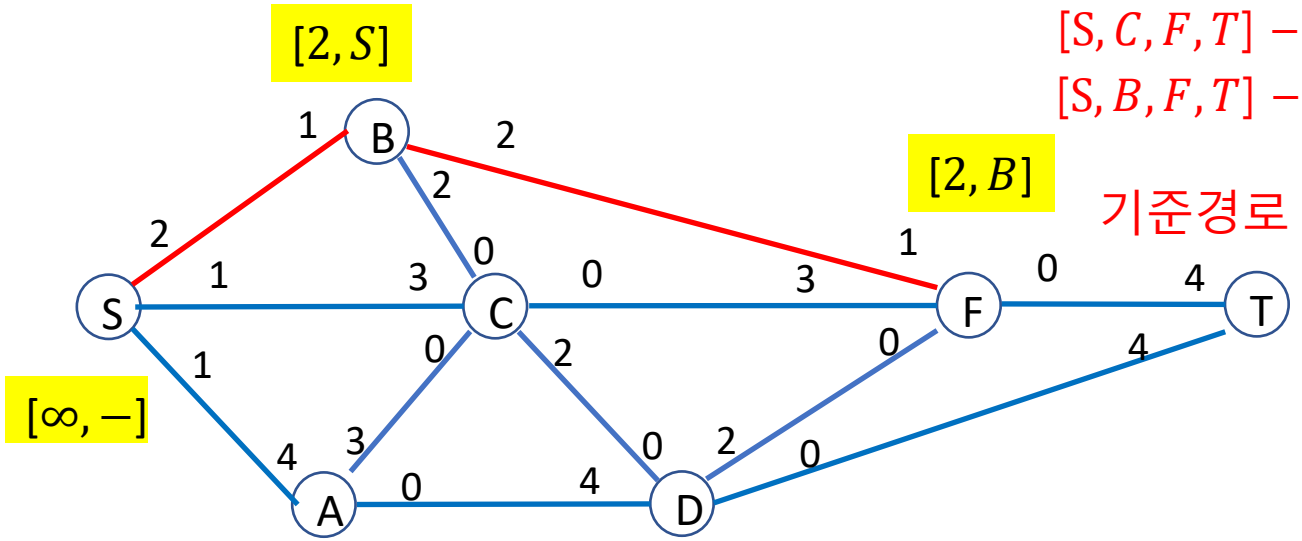
➢반복4



[S, A, D, T] − 4

[S, C, F, T] − 3

[S, B, F, T] − 1



6.6 최대흐름문제

❖ Cut을 이용한 계산

➢ Cut의 정의

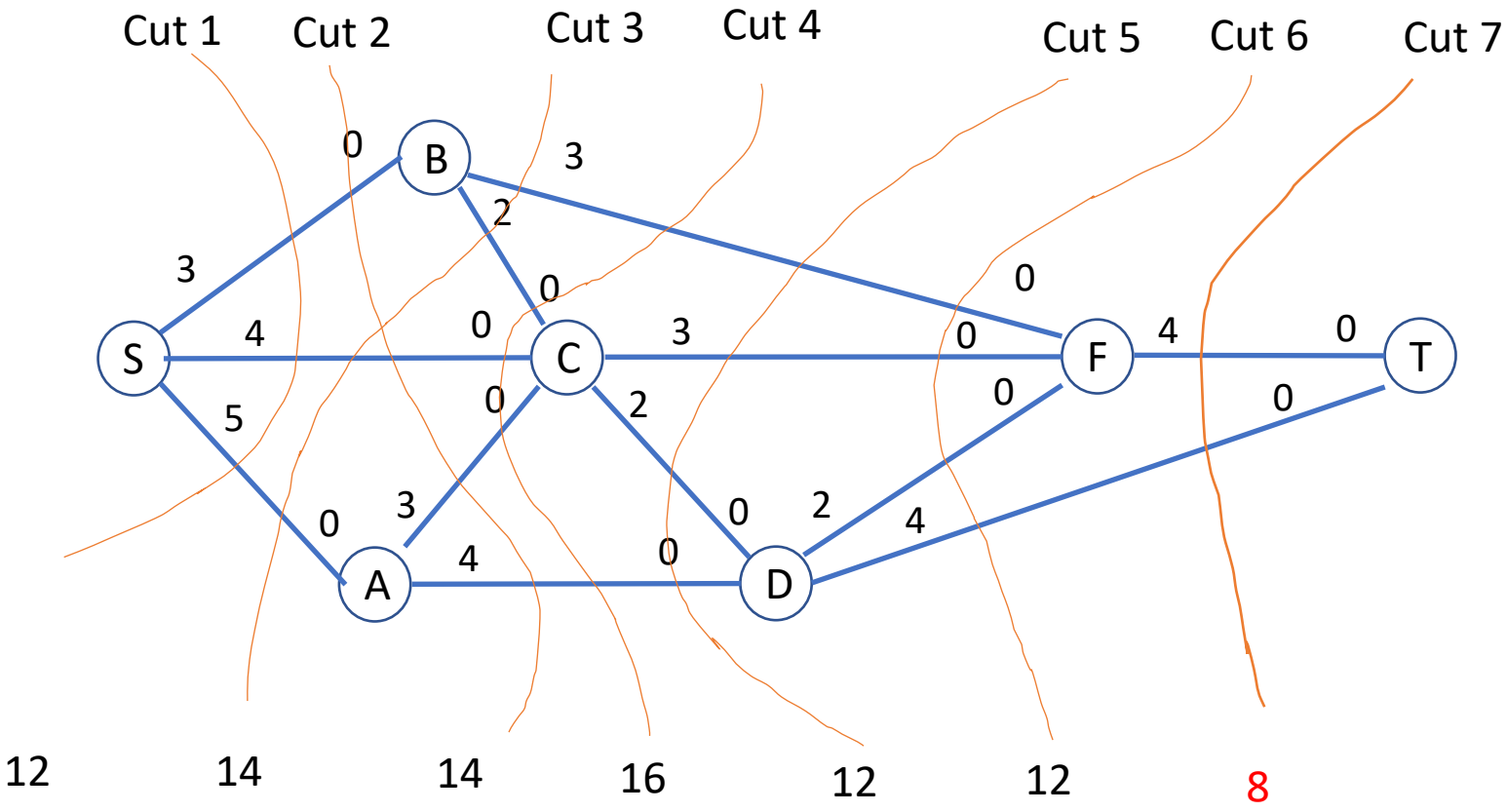
- ✓ 네트워크로부터 제거될 때 시작마디부터 종료마디까지의 흐름을 완전하게 차단할 수 있는 호들의 집합
- ✓ 최소의 Cut용량을 갖는 Cut이 최대 흐름량을 제공
- ✓ 네트워크 상에 모든 가능한 Cut들을 열거할 수 없다면 최대 흐름량의 계산을 보장받을 수 없다. 다만, 열거된 Cut들 중에서 얻을 수 있는 최소 흐름용량을 이용하여 최대흐름문제에 대한 상한값(upper bound value)을 보장받을 수 있다.

➢ [예제] Cut을 이용한 최대 흐름량 계산

- ➢ 네트워크를 이용하여 가능한 Cut들을 열거하고 각 Cut들에 대한 Cut 용량을 계산

6.6 최대흐름문제

❖ Cut을 이용한 계산



6.6 최대흐름문제

❖ 최대흐름 문제

➢ 선형계획모형

- ✓ x_{ij} = 호(i, j) 사이의 흐름량
- ✓ $u_{ij}(l_{ij})$ = 호(i, j) 의 용량 상한(하한)값
- ✓ f = 최대 흐름량
- ✓ 목적함수: $Maximize \quad f$
- ✓ 제약식: 총 진입흐름량 = 총 진출흐름량

$$\sum_{\{(i,k)\in A\}} x_{ik} - \sum_{\{(k,j),\in A\}} x_{kj} = \text{순수요량 (-f, o, f)}$$

$$l_{ij} \le x_{ij} \le u_{ij}$$

6.7 최소비용 용량제약 네트워크 문제

❖ 최소비용 용량제약 네트워크문제

- 각 호에 대한 흐름용량제약과 각 마디에서의 공급량 및 수요량을 만족시키면서 총비용을 최소화하는 문제
- 각 호에서의 최적 흐름량을 결정

- 수송문제, 할당문제, 최단경로문제 및 최대흐름문제 등은 최소비용 용량제약 네트워크문제의 특수한 경우
- 최소비용 용량제약 네트워크문제는 다음과 같은 가정들을 기반으로 한다
  - ✓ 각 호에 대해 단위흐름비용이 주어진다.
  - ✓ 각 호는 양의 하한값을 가질 수 있다.
  - ✓ 각 마디에는 공급량 혹은 수요량이 주어진다

6.7 최소비용 용량제약 네트워크 문제

❖ 선형계획모형

➢ 용량제약을 갖는 네트워크 $G(N, A)$를 고려

➢ 기호 정의

✓ x_{ij} = 마디 i 로 부터 마디 j 까지의 흐름량

✓ $u_{ij}(l_{ij})$ = 호(i, j) 의 용량 상한(하한)값

✓ C_{ij} = 마디 i 로 부터 마디 j 까지의 단위당 흐름 비용

✓ f_i = 마디 i 에서의 순 흐름량

✓ 목적함수: 각 호의 흐름량과 관련된 총비용

$$Minimize \quad \sum_{(i,j)} \sum_{A} C_{ij} x_{ij}$$

✓ 제약식: 총 진출흐름량 - 총 진입흐름량 = 순흐름량

$$\sum_{\{(k,j) \in A\}} x_{kj} - \sum_{\{(i,k), \in A\}} x_{ik} = f_k$$

$$l_{ij} \leq x_{ij} \leq u_{ij}$$

6.7 최소비용 용량제약 네트워크 문제

❖ 선형계획모형

- ➢ 용량제약을 갖는 네트워크 $G(N, A)$를 고려
- ➢ 하한값의 대체
  - ✓ $x_{ij} = x'_{ij} + l_{ij}$

$$Minimize \quad \sum_{(i,j)} \sum_{A} C_{ij} x_{ij}$$

$$s.t. \quad \sum_{\{(k,j)\in A\}} x_{kj} - \sum_{\{(i,k),\in A\}} x_{ik} = f_k$$

$$0 \le x_{ij} \le u_{ij}$$

6.8 CPM/PERT

❖ CPM(Critical Path Method) and PERT(Program Evaluation and Review Technique)

- 프로젝트: 시간과 자원을 소모하며 상호연관성을 갖는 활동(activity)들의 집합
- 각 활동은 작업기간(duration)과 선행활동 리스트를 가짐
- 이러한 프로젝트의 계획,일정관리 및 통제를 위해 개발된 네트워크 기반의 기법

❖ 기본 용어

- 활동 (activity)
- 작업기간 (duration)
- 선행활동 (predecessor)
- 최조 시작기간 (earliest start time)
- 최지 시작시간 (latest start time)
- 여유시간 (slack time)

6.8 CPM/PERT

❖ CPM(Critical Path Method) 프로젝트

- CPM의 적용을 위해서는 작업 활동들 간의 상호 선행관련성을 고려하여 프로젝트 네트워크를 만들어야 함
- 프로젝트의 각 활동은 프로젝트 내에서의 진행방향을 나타내는 호에 의해 표현
- 네트워크의 마디들은 프로젝트의 다른 활동들 사이에 선행관련성을 나타냄
- 네트워크 작성을 위한 3가지 기준
  - ✓ 기준 1: 각 활동은 반드시 하나의 호로만 표현된다.
  - ✓ 기준 2: 각 활동은 2개의 서로 다른 마디들로 표현되어야 한다.
  - ✓ 기준 3: 하나의 활동이 네트워크에 추가될 때 마다 정확한 상호 선행관련성을 유지하기 위해 다음과 같은 사항을 반드시 확인하여야 한다:

    (1) 어떤 활동들이 현 활동 바로 전에 작업되어야만 하는가?

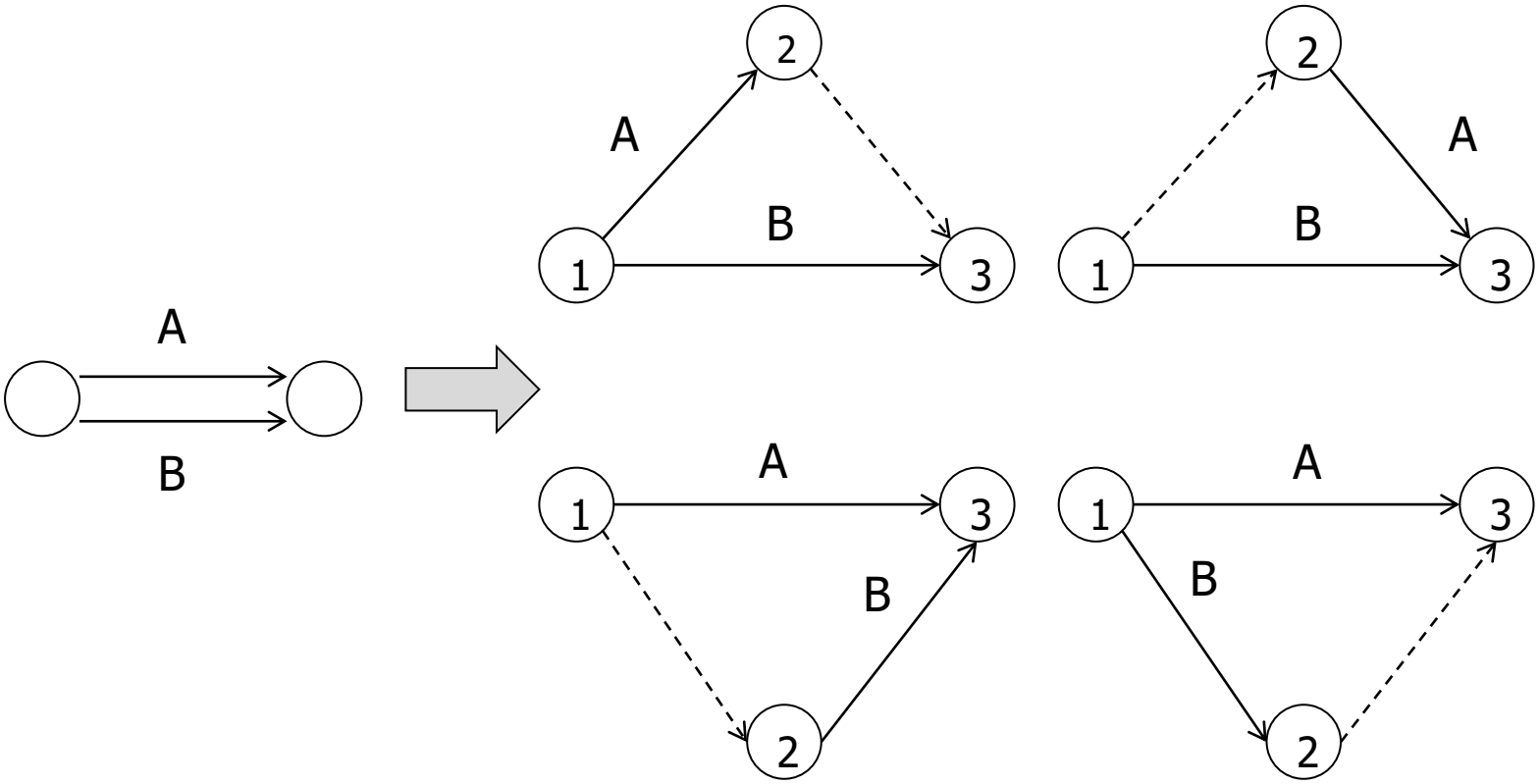
    (2) 어떤 활동들이 현 활동 이후에 작업되어야만 하는가?

    (3) 어떤 활동들이 현 활동과 동시에 발생되어야만 하는가?

❖ CPM(Critical Path Method) 프로젝트

➢ 가상 활동의 사용 예

✓ 동시 발생 활동의 표현을 위한 가상활동의 사용

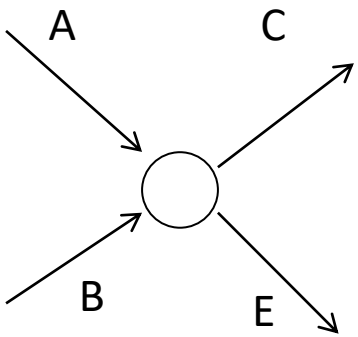


❖ CPM(Critical Path Method) 프로젝트

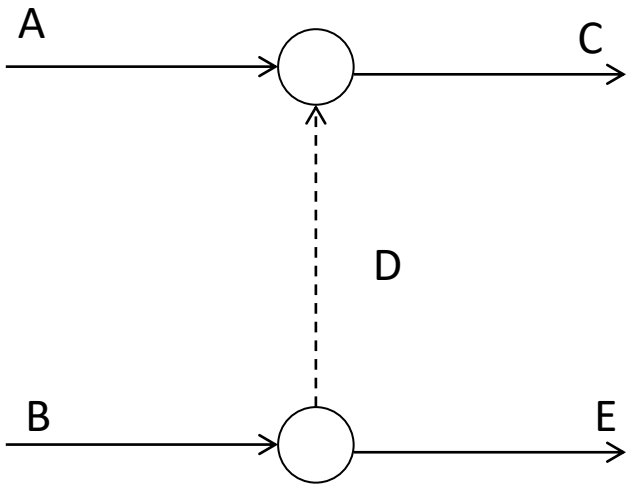
➢ 가상 활동의 사용 예

✓ 정확한 선행 관계 표현을 위한 가상 활동의 사용

- 활동 C는 활동 A와 B가 완료된 이후에 즉시 시작한다.
- 활동 E는 활동 B 만 완료되면 시작한다.



(가) 잘못된 표현



(나) 올바른 표현

❖ CPM(Critical Path Method) 프로젝트

➢ 예제

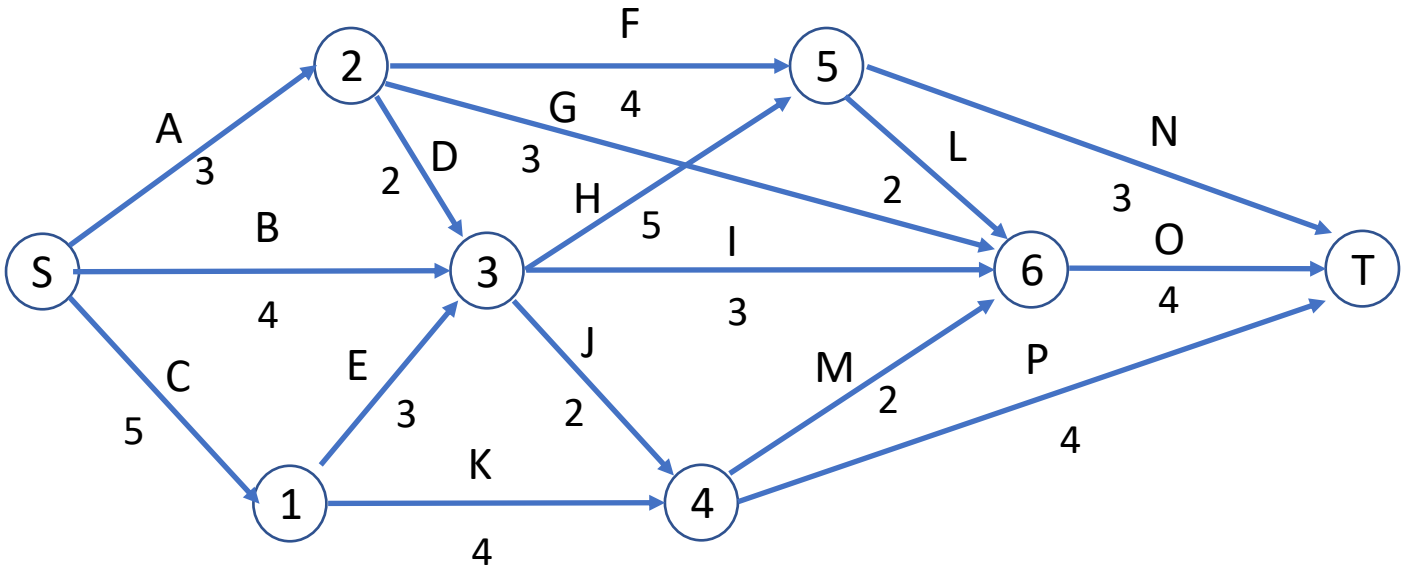
✓ 다음과 같은 프로젝트가 주어졌을 때, 주경로와 최단완성시간?

활동	시간	선행작업		활동	시간	선행작업
A	3	-		I	3	B, D, E
B	4	-		J	2	B, D, E
C	5	-		K	4	C
D	2	A		L	2	F, H
E	3	C		M	2	J, K
F	4	A		N	3	F, H
G	3	A		O	4	G, I, L, M
H	5	B, D, E		P	4	J, K

❖ CPM(Critical Path Method) 프로젝트

➢ 예제

✓ 다음과 같은 프로젝트가 주어졌을 때, 주경로와 최단완성시간?



6.8 CPM/PERT

❖ CPM(Critical Path Method) 프로젝트

➢ 주경로

✓ 어떤 프로젝트에서 임의의 활동 k 의 최조 시작시간은 대응되는 프로젝트 네트워크에서 시작마디로부터 마디 k 까지의 최장경로의 길이와 같다. 이러한 최장경로를 주경로(critical path)라 한다

➢ 최소 완료시간

✓ 어떤 프로젝트의 최소 완료시간은 대응되는 프로젝트 네트워크에서 시작마디로부터 종료마디 까지의 최장경로 혹은 주경로의 길이와 같다.

6.8 CPM/PERT

❖ CPM 최조 시작일정 알고리즘

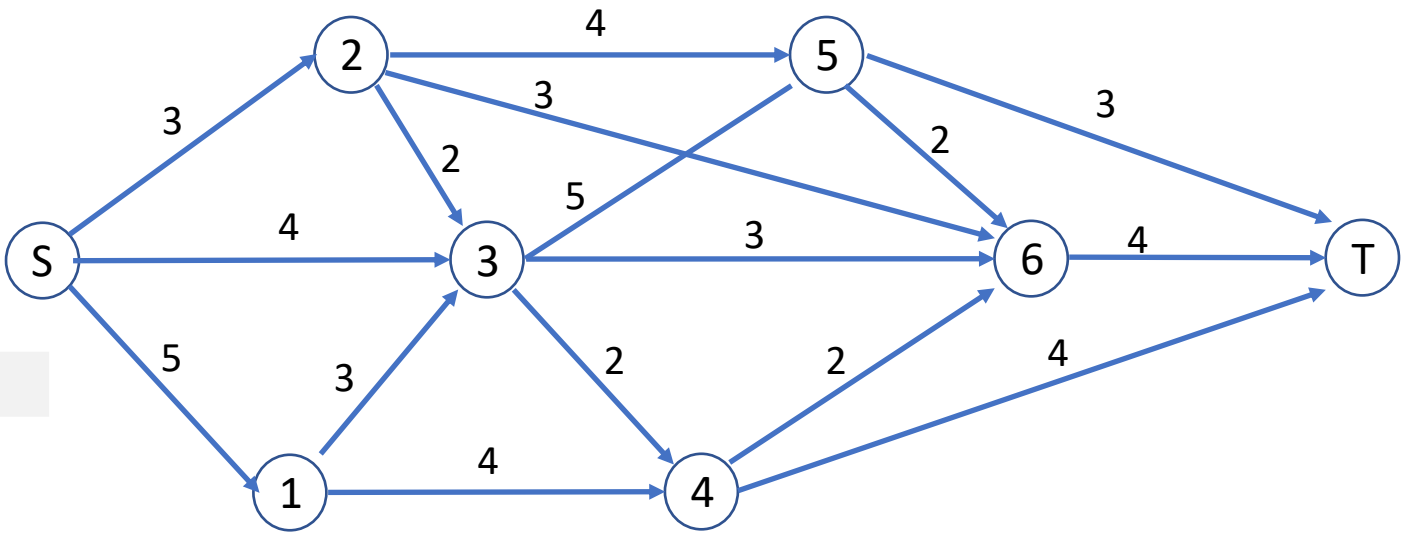
- ➢ 최단경로 알고리즘을 최장경로(Longest Route)를 구하는 형태로 수정
- ➢ 기호 정의
  - ✓ v[i] = 시작마디 1로부터 마디 i 까지의 최장경로의 길이,
  - ✓ s[i] = 최장경로 상에 마디 i 의 직전 선행 마디,
  - ✓ a_i = 활동 i 를 완료하기 위해 요구되는 시간.
- ➢ 단계0 : 네트워크 상의 모든 선행관계를 나타내는 호 (i, j)가 $i < j$ 를 만족하 도록 마디번호를 새롭게 정렬. 시작마디는 v[S]=0
- ➢ 단계1 : 종료마디의 최조 시작시간이 발견되었다면 종료. 아니면, 남아 있는 마디들 중 번호가 가장 작은 마디 p 를 선택
- ➢ 단계2 : 마디 p 에 대한 최조 시작시간을 계산

  $v[p] = \max\{v[i] + a_i\}$, i 는 p의 선행 활동

  s[p] 에 최대값을 제공하는 마디의 번호를 저장. [단계 1]로 간다.

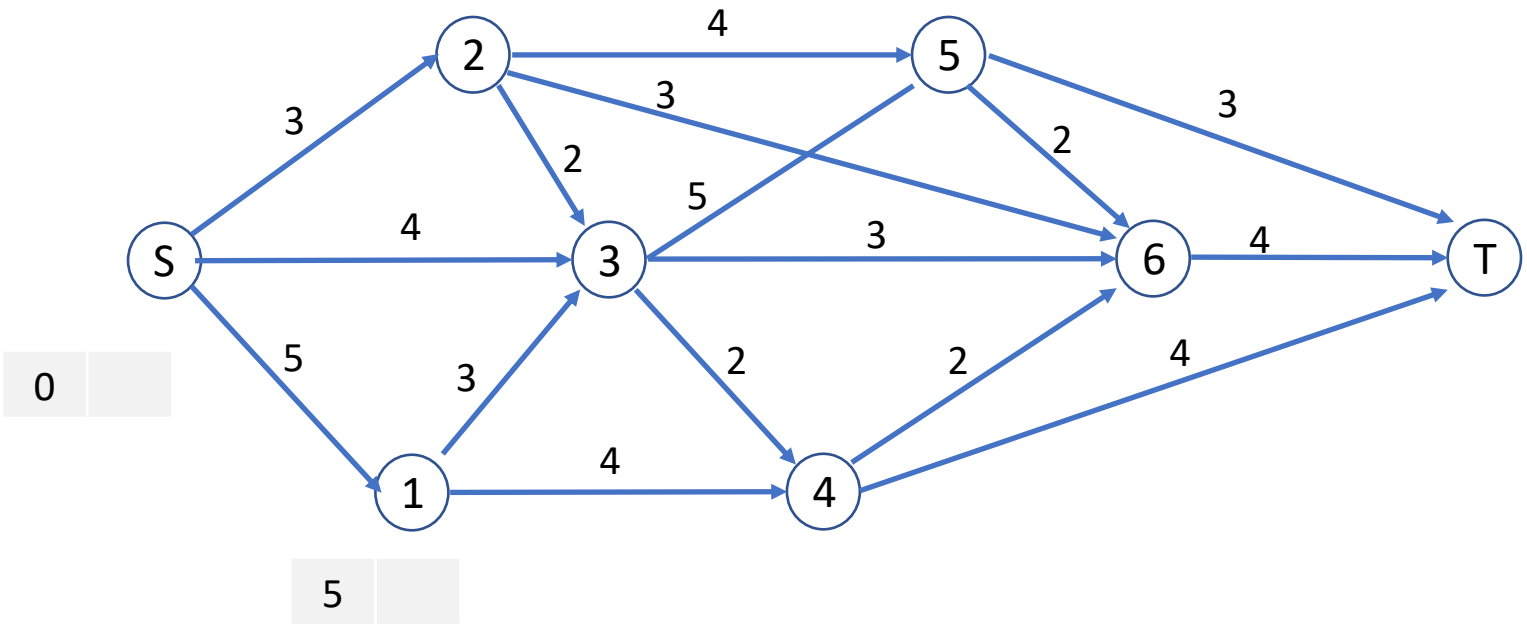
6.8 CPM/PERT

❖ CPM 최조 시작일정 알고리즘



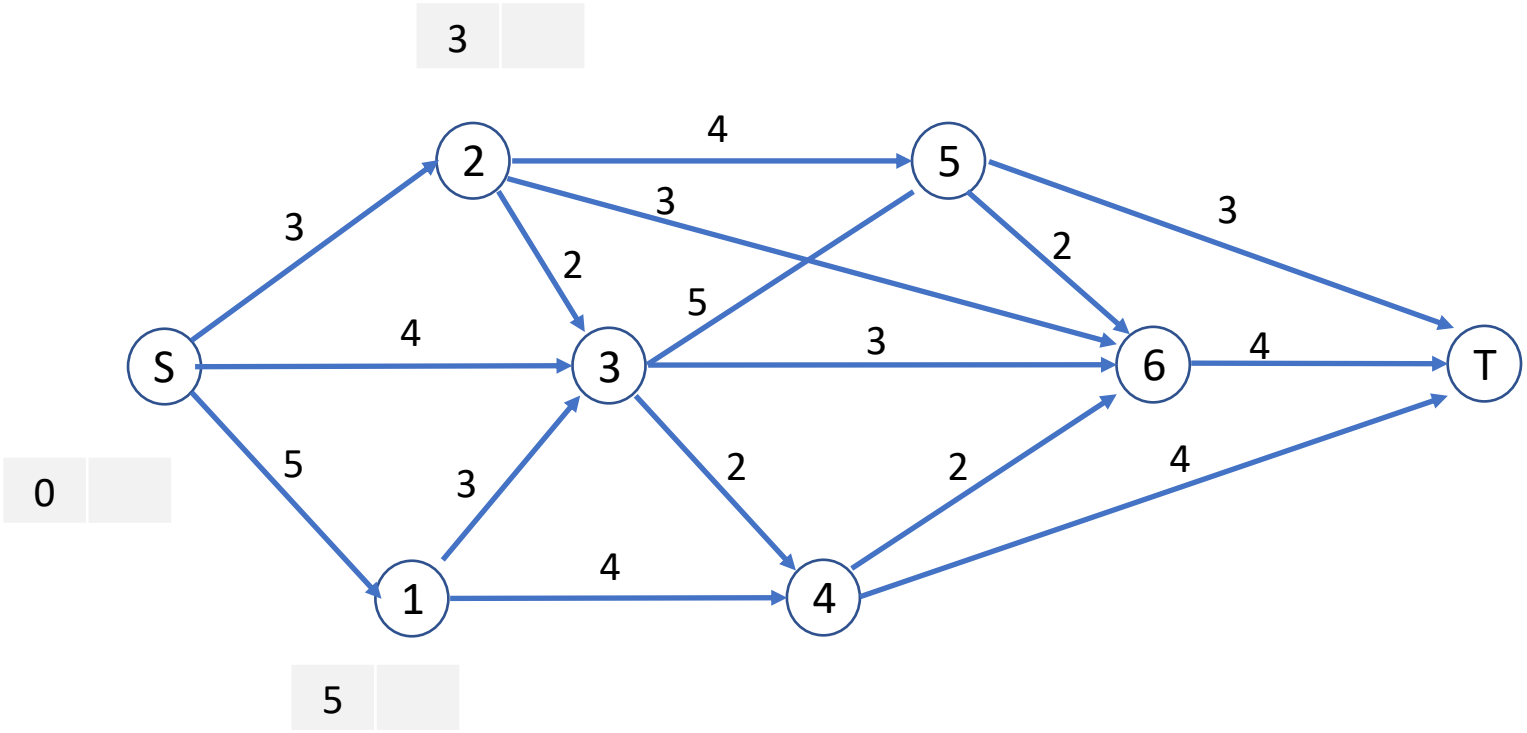
6.8 CPM/PERT

❖ CPM 최조 시작일정 알고리즘



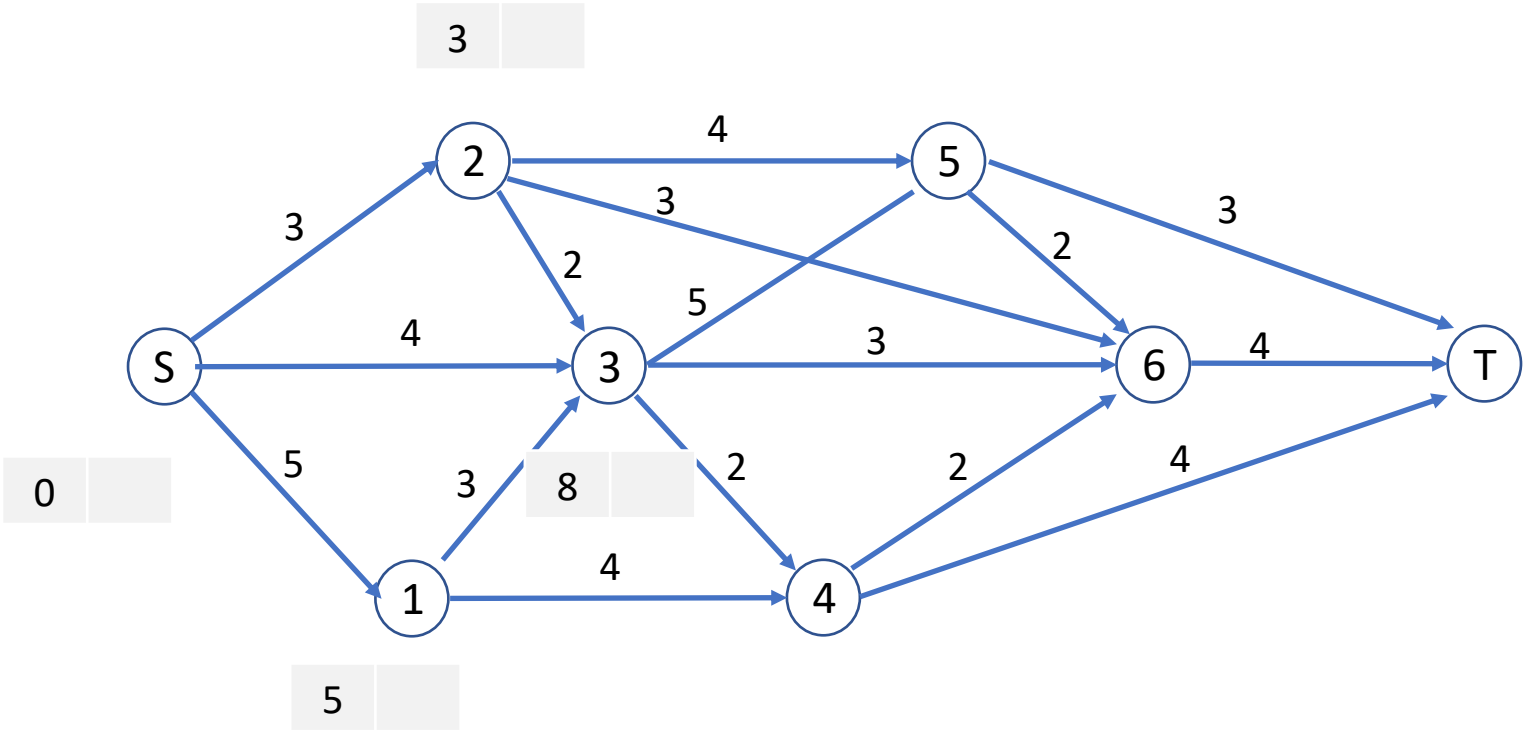
6.8 CPM/PERT

❖ CPM 최조 시작일정 알고리즘

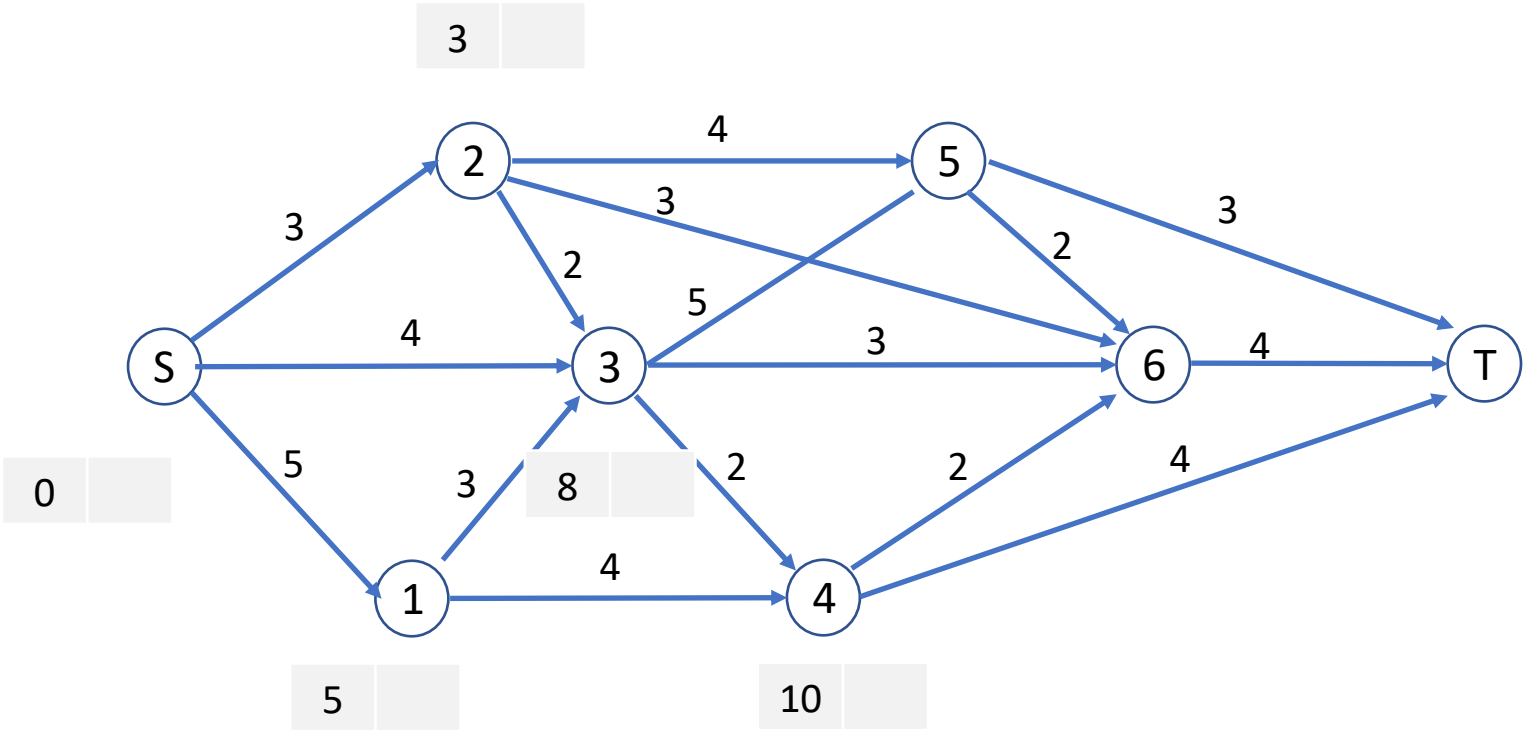


6.8 CPM/PERT

❖ CPM 최조 시작일정 알고리즘

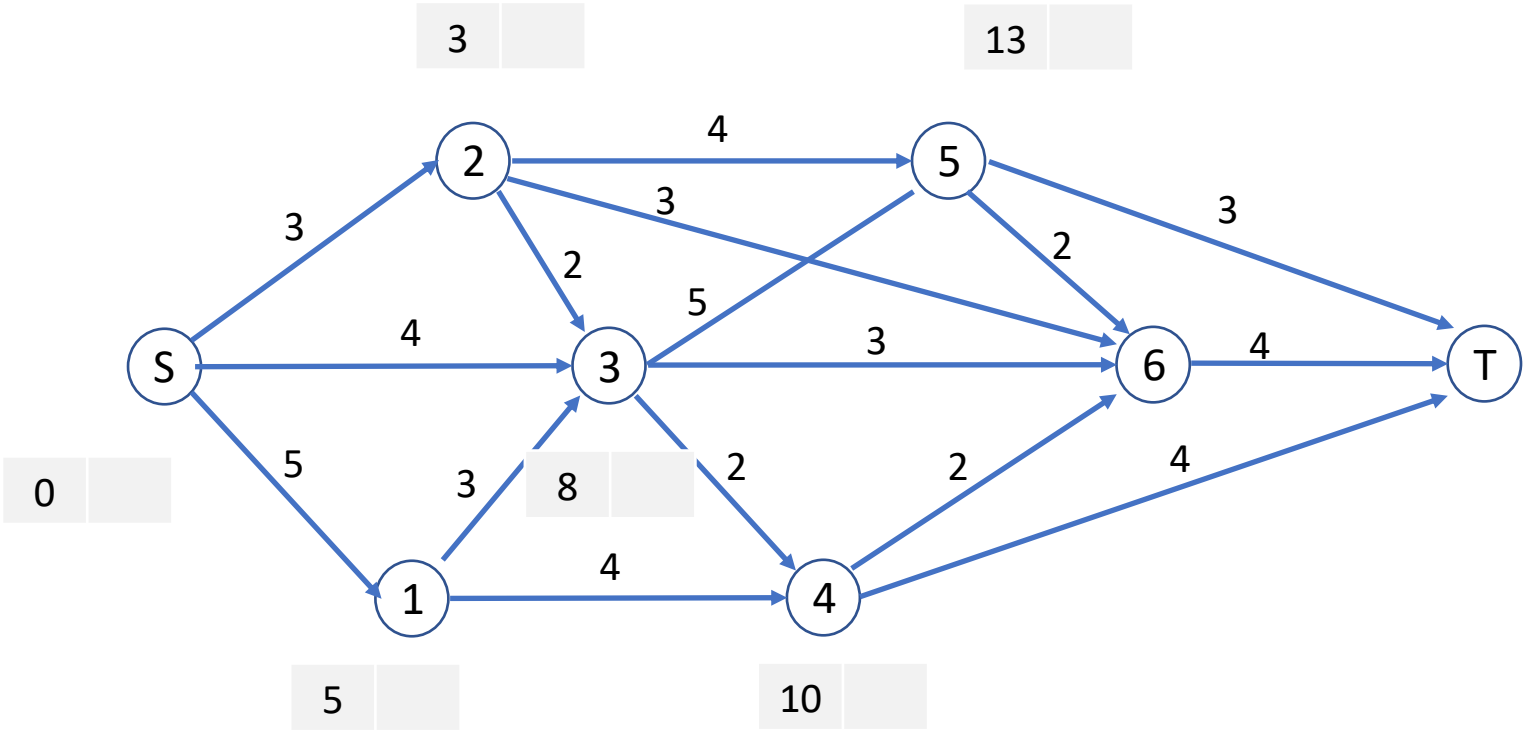


❖ CPM 최조 시작일정 알고리즘



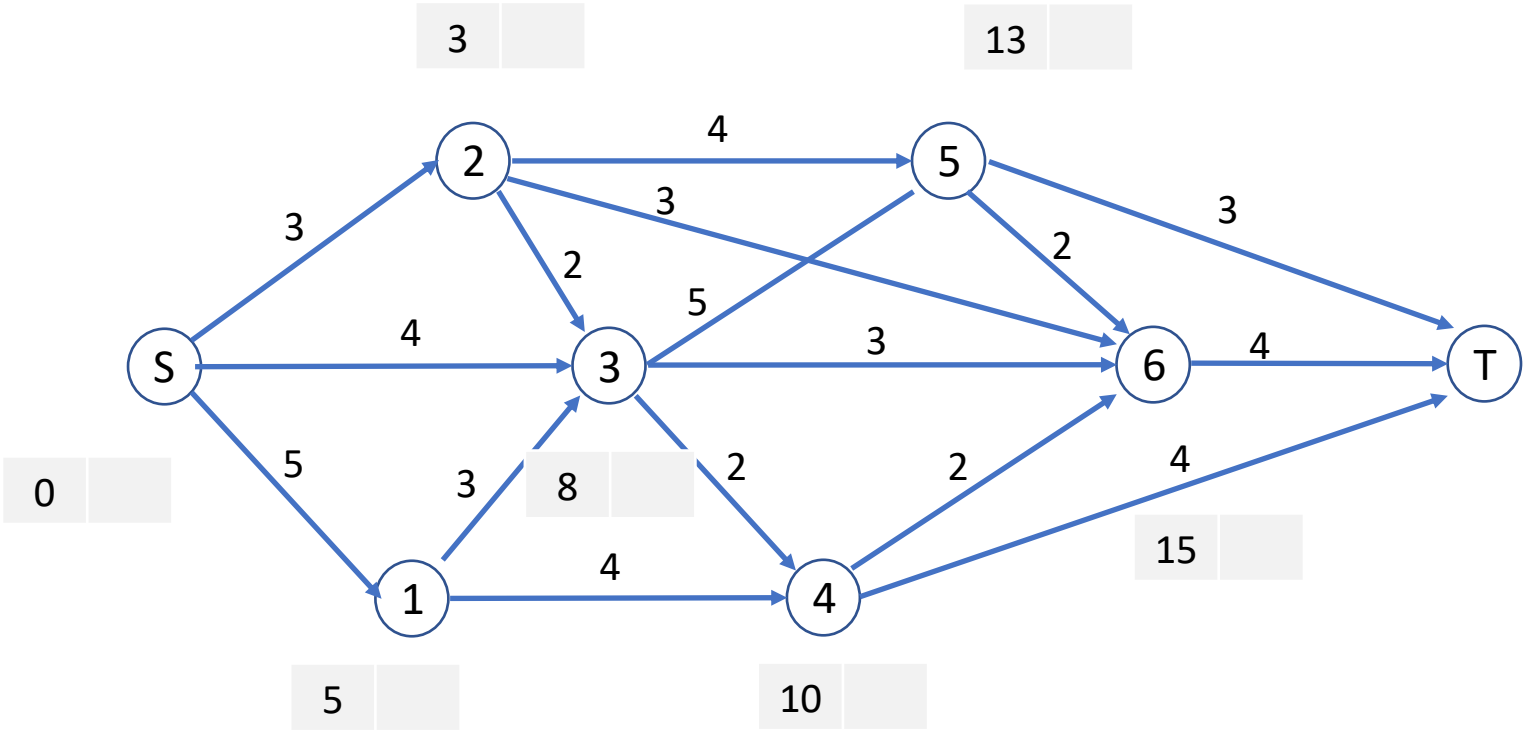
6.8 CPM/PERT

❖ CPM 최조 시작일정 알고리즘



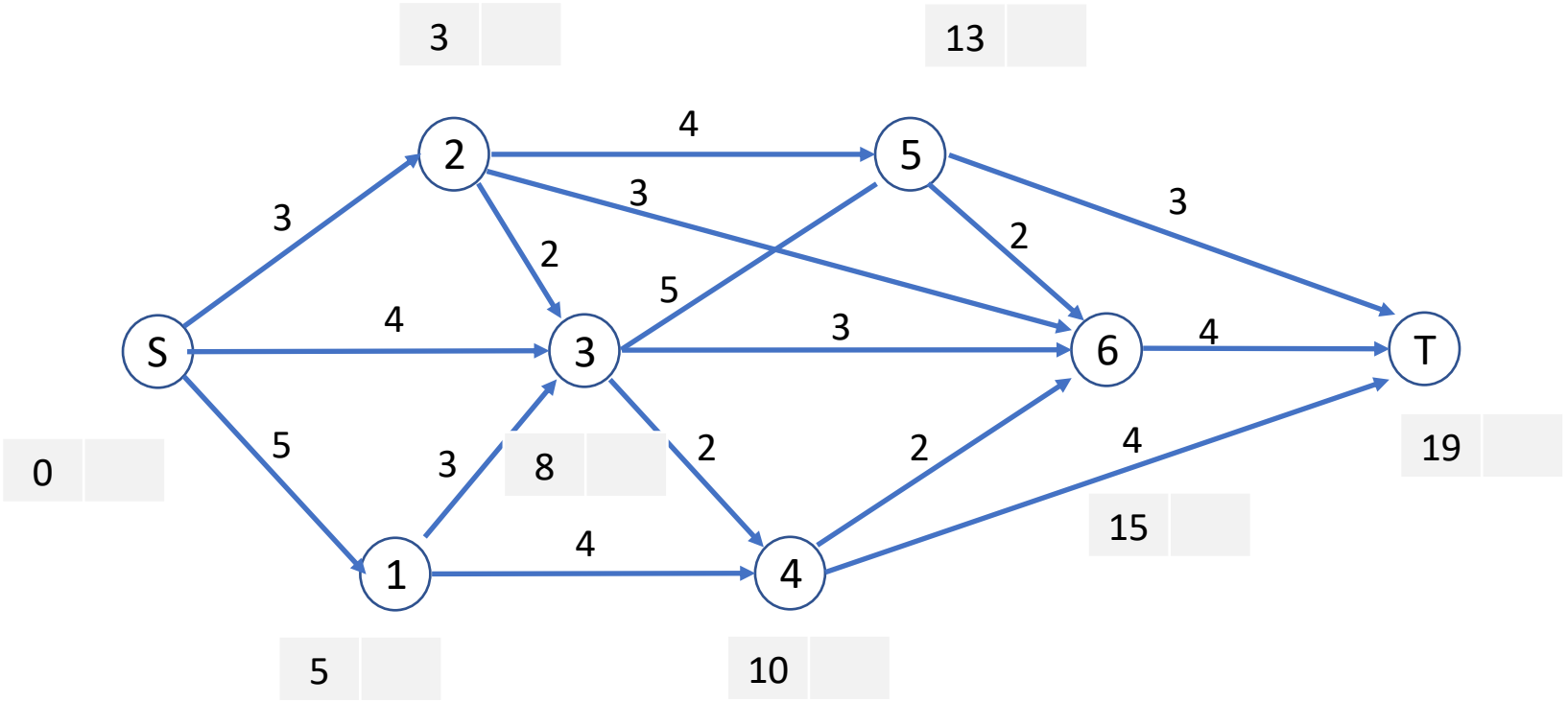
6.8 CPM/PERT

❖ CPM 최조 시작일정 알고리즘



6.8 CPM/PERT

❖ CPM 최조 시작일정 알고리즘



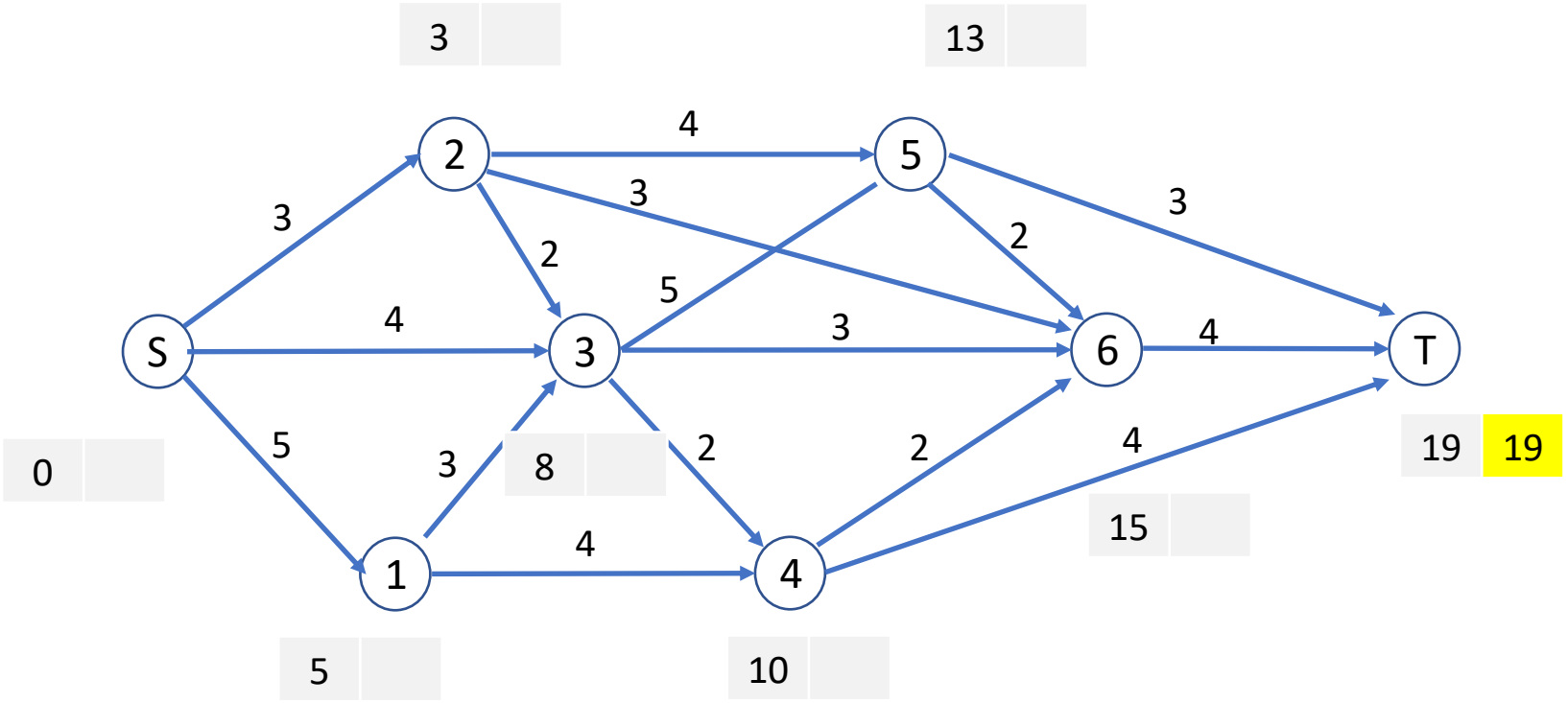
6.8 CPM/PERT

❖ CPM 최지 시작일정 알고리즘

- 어떤 프로젝트에서 임의의 활동 k 의 최지 시작시간은 대응되는 CPM 네트워크에서 마디 k 로부터 종료마디까지의 최장경로의 길이를 납기일에서 빼준 값과 같다.
- 여유시간은 전체 프로젝트 납기일에 맞추어 각 활동을 일정관리함에 있어 얼마나 많은 관리상의 여유를 가질 수 있는 지를 보여준다.

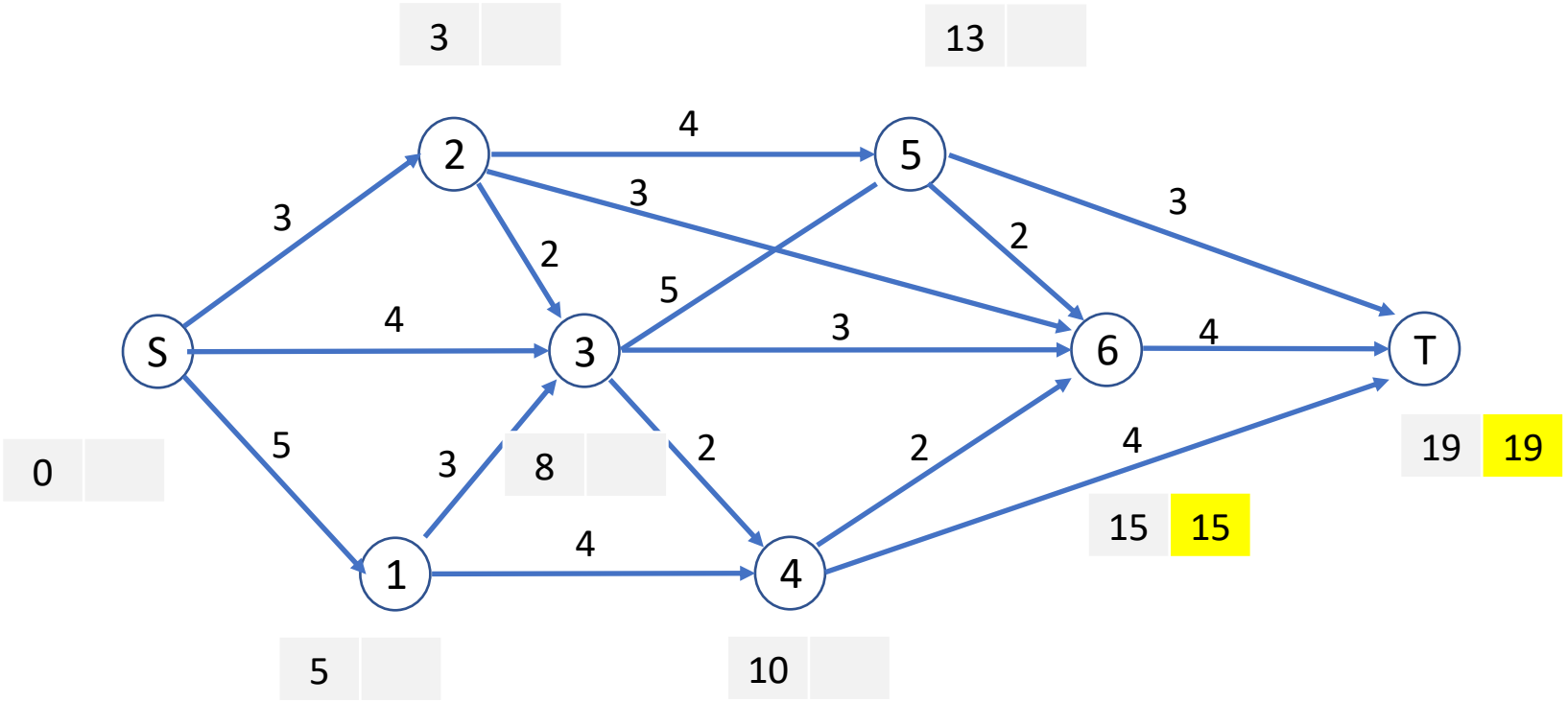
- 최지 시작시간은 CPM 최조 시작일정 알고리즘을 역순으로 적용하면 간단히 계산

❖ CPM 최지 시작일정 알고리즘

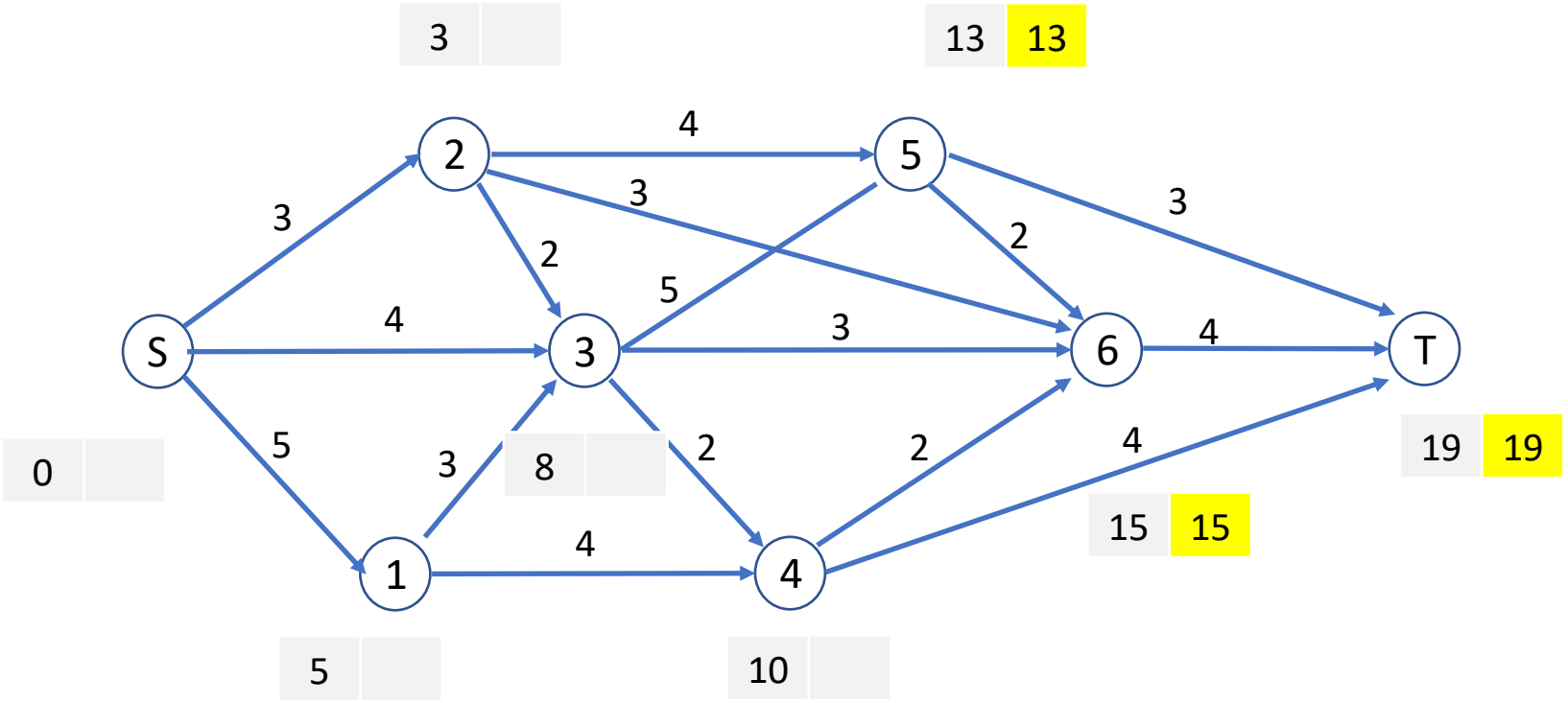


6.8 CPM/PERT

❖ CPM 최지 시작일정 알고리즘

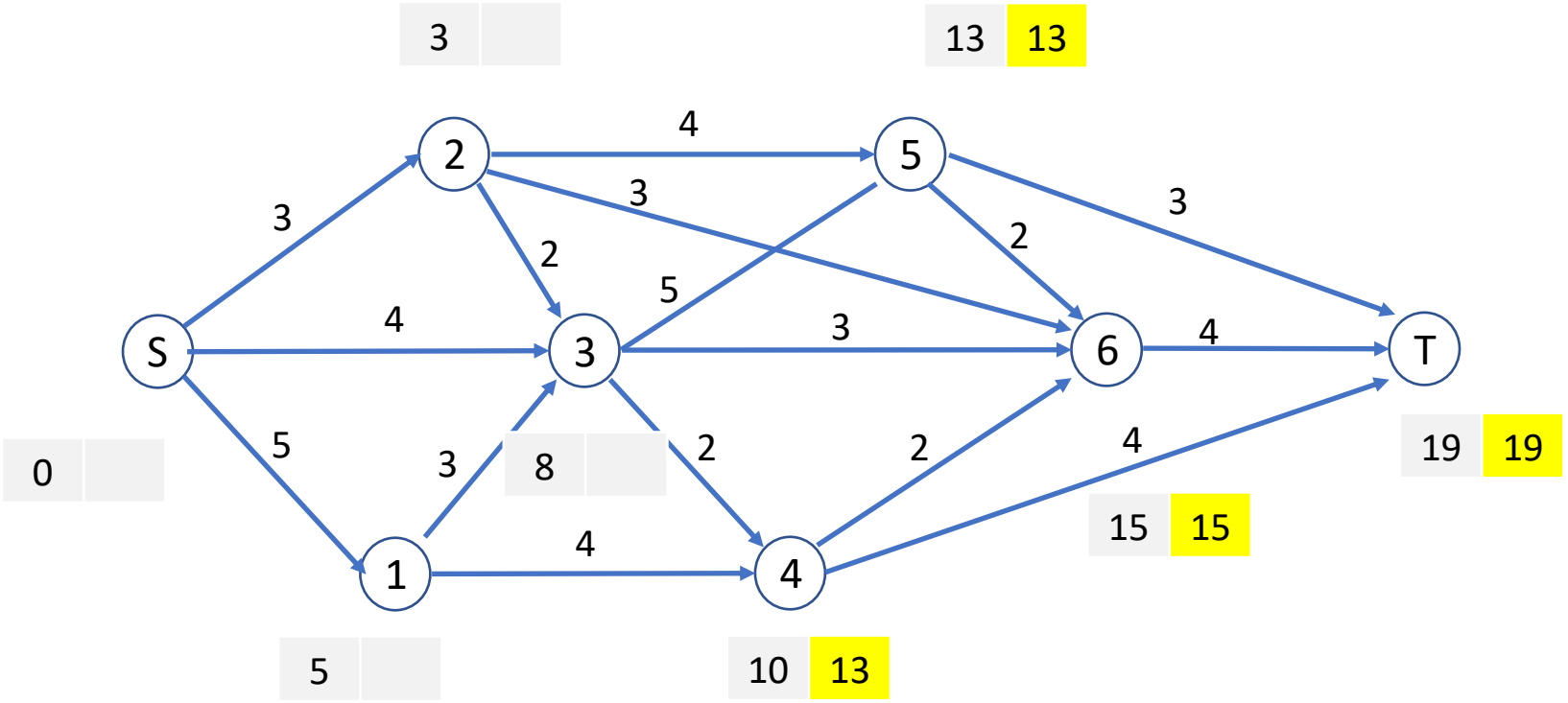


❖ CPM 최지 시작일정 알고리즘



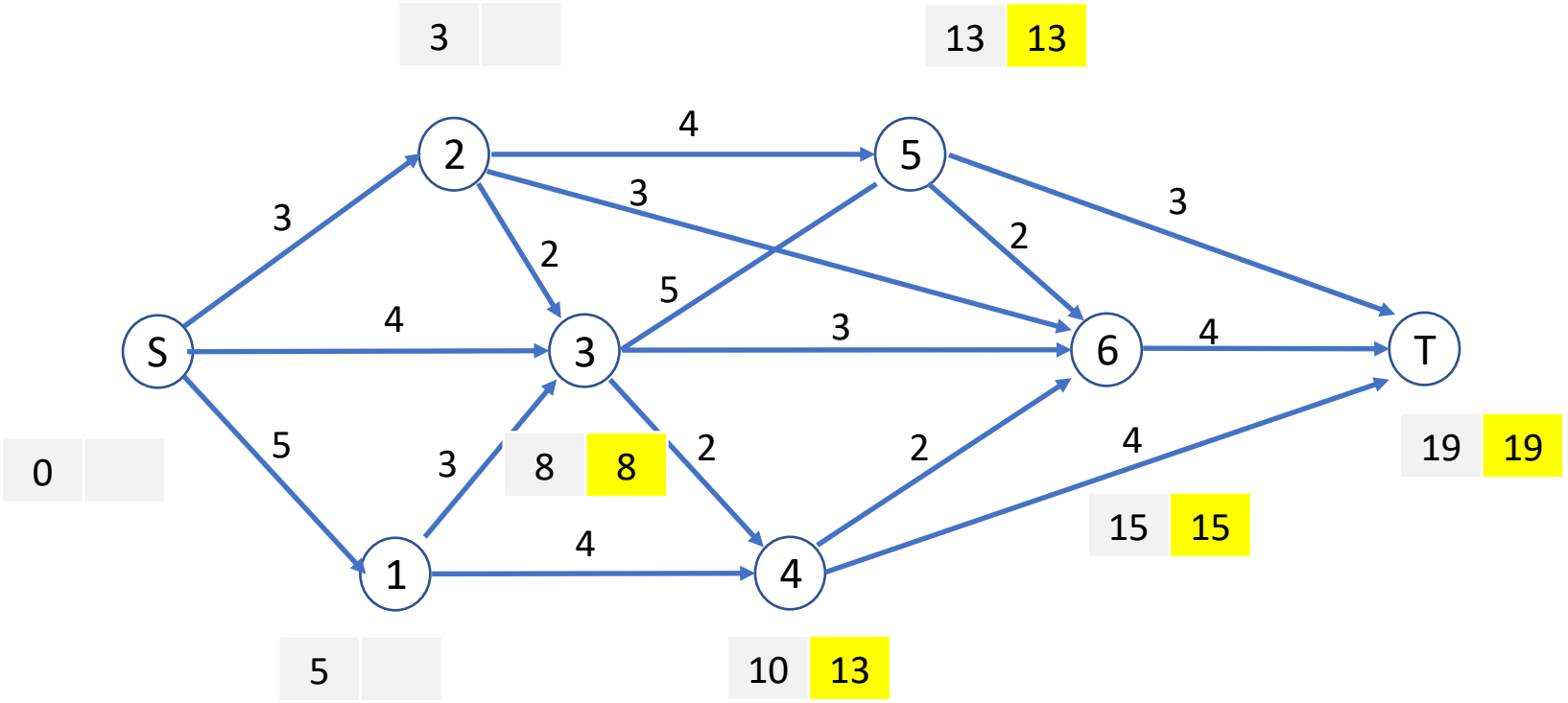
6.8 CPM/PERT

❖ CPM 최지 시작일정 알고리즘

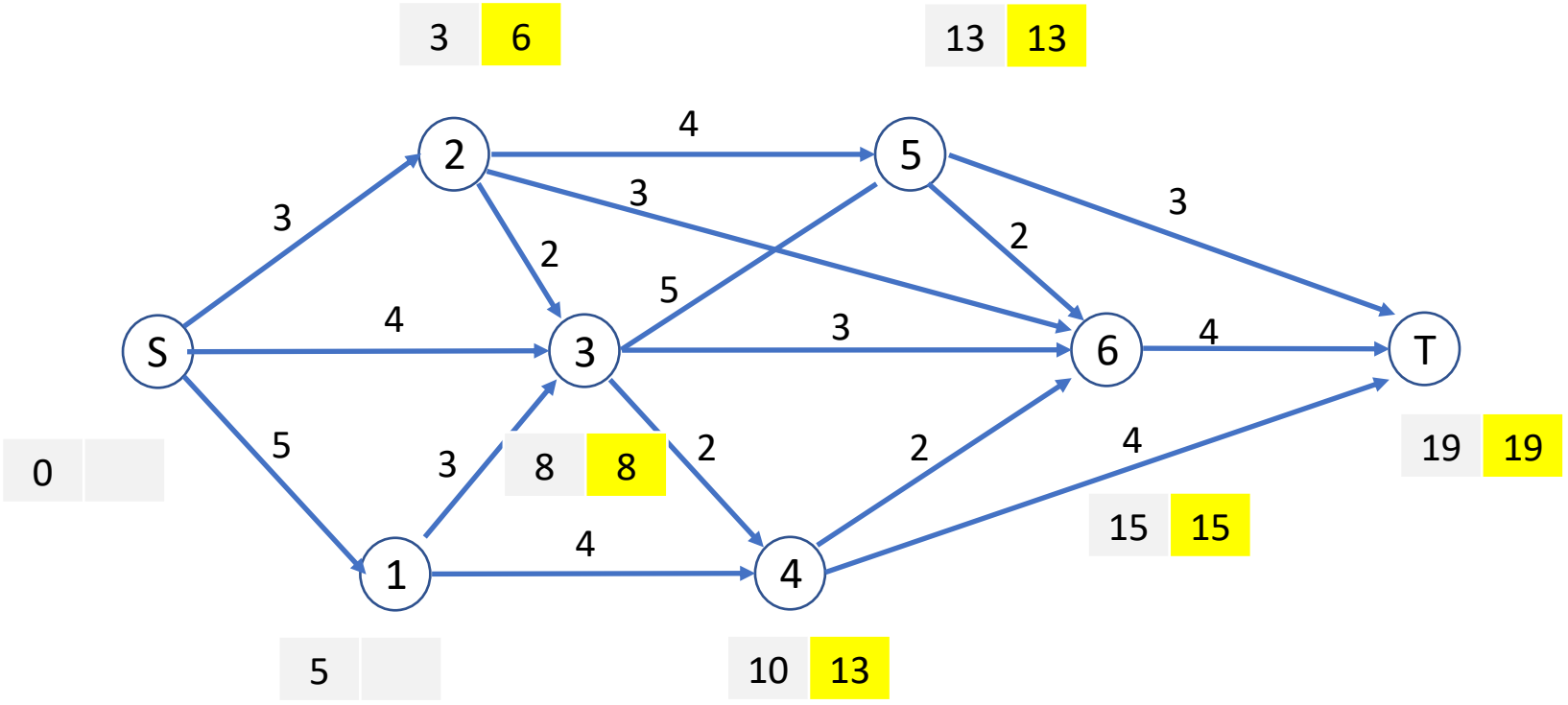


6.8 CPM/PERT

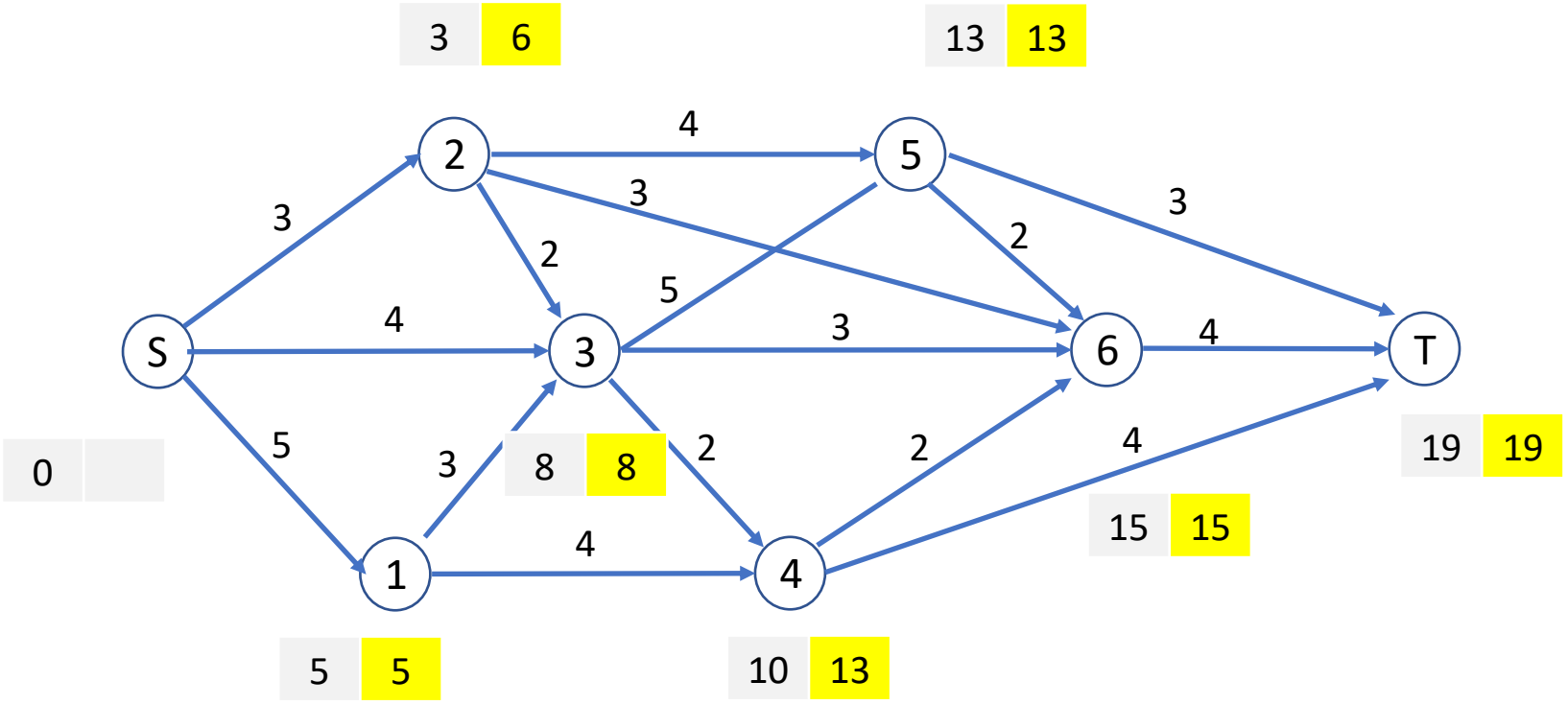
❖ CPM 최지 시작일정 알고리즘



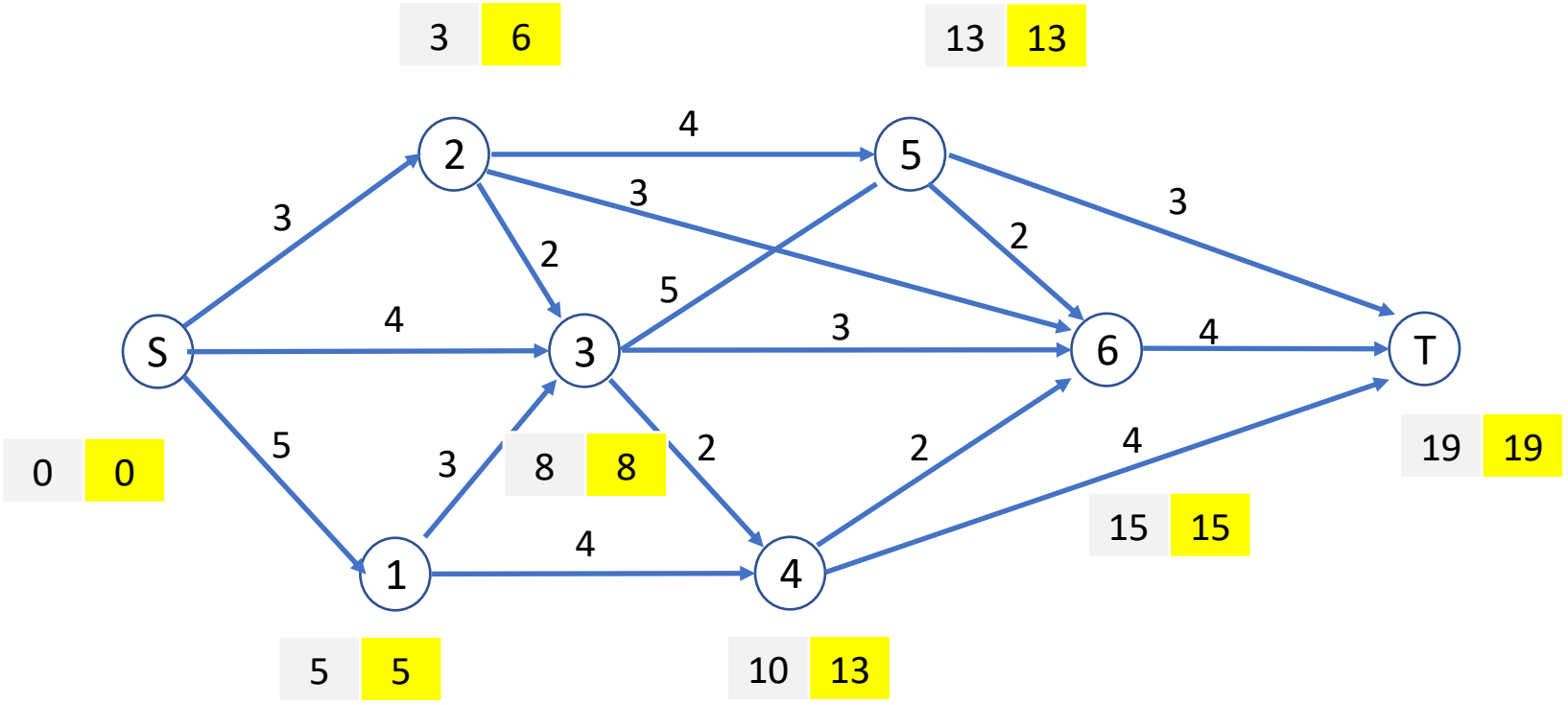
❖ CPM 최지 시작일정 알고리즘



❖ CPM 최지 시작일정 알고리즘

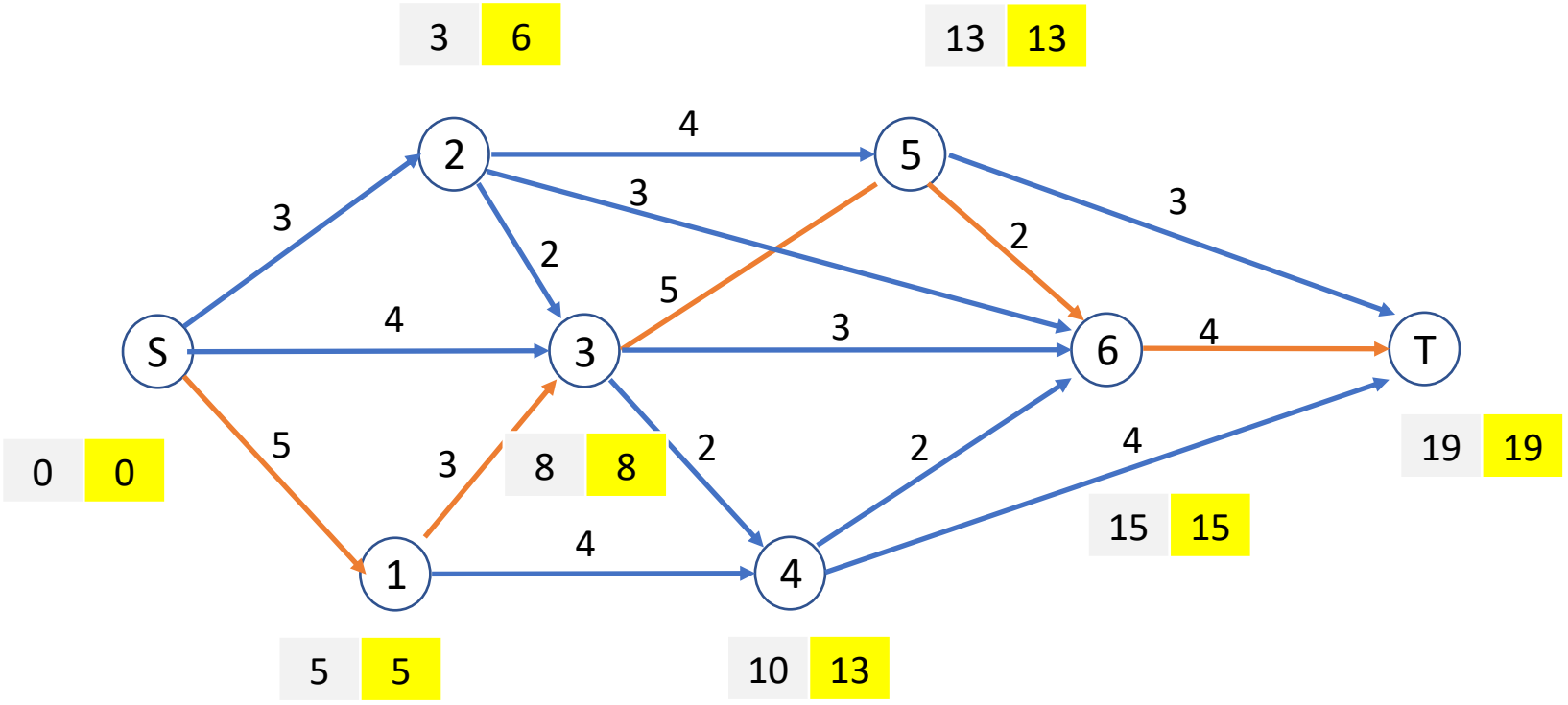


❖ CPM 최지 시작일정 알고리즘



❖ 주경로, 최조 및 초지 시작시간

주경로 단축방법?



6.8 CPM/PERT

❖ 선형계획모형

- ➢ 입력변수
  - ✓ d_{ij} = 마디 i 로 부터 마디 j 까지의 작업시간
- ➢ 의사결정변수
  - ✓ x_{ij} = 마디 i 로 부터 마디 j 까지의 흐름량
- ➢ 목적함수: 총 흐름량의 최대화

$$Maximize \quad \sum_{(i,j)} d_{ij} x_{ij}$$

- ➢ 제약식: 총 진출흐름량 = 총 진입흐름량

$$\sum_{\{(k,j)\in A\}} x_{kj} - \sum_{\{(i,k),\in A\}} x_{ik} = \text{순수요량(-1, 0, 1)}$$

6.8 CPM/PERT

❖ PERT

- 활동의 작업기간에 대한 추정치를 하나만 사용하는 것은 비합리적일 수 있음
- 이것은 모든 활동들이 정확히 작업기간을 지키며 수행된다는 확실성의 전제 하에서만 가능
- 따라서, 작업기간의 불확실성을 고려하기 위해서는 작업기간에 대한 확률이나 확률분포를 가정
- 프로젝트의 완료날짜를 정하는 대신에 기대 프로젝트 완료날짜 혹은 어떤 기간 이내에 프로젝트가 완료될 확률을 계산할 필요가 있음
- 이와 같은 프로젝트 관리를 위한 확률적 접근방법을 PERT라 함

6.8 CPM/PERT

❖ PERT

➢ PERT 네트워크는 3가지 추정치를 활동의 작업기간으로 사용

(1) 낙관적 시간(a) : 프로젝트가 최선의 경우로 진행될 때 활동의 작업기간

(2) 최빈시간(m) : 프로젝트가 정상적인 상태에서 진행될 때 활동의 작업기간

(3) 비관적 시간(b) : 프로젝트가 최악의 경우로 진행될 때 활동의 작업기간

➢ 활동 k 의 평균작업시간과 분산

$$\mu_k = \frac{a+4M+b}{6} \qquad \sigma_k^2 = \left(\frac{b-a}{6}\right)^2$$

6.8 CPM/PERT

❖ PERT

- CPM 네트워크에서 CPM의 계산은 작업기간 대신에 평균 작업기간 μ_k 를 대체하여 계산
- 활동들의 작업기간이 확률변수들이므로 주경로상의 활동들의 작업기간의 합인 프로젝트 기간도 확률변수
- 중심극한정리(Central Limit Theorem)에 의해, 프로젝트 완료시간은 평균과 분산이 개별 활동들의 평균들과 분산들의 합과 같은 정규분포(Normal Distribution)를 따른다고 가정

6.8 CPM/PERT

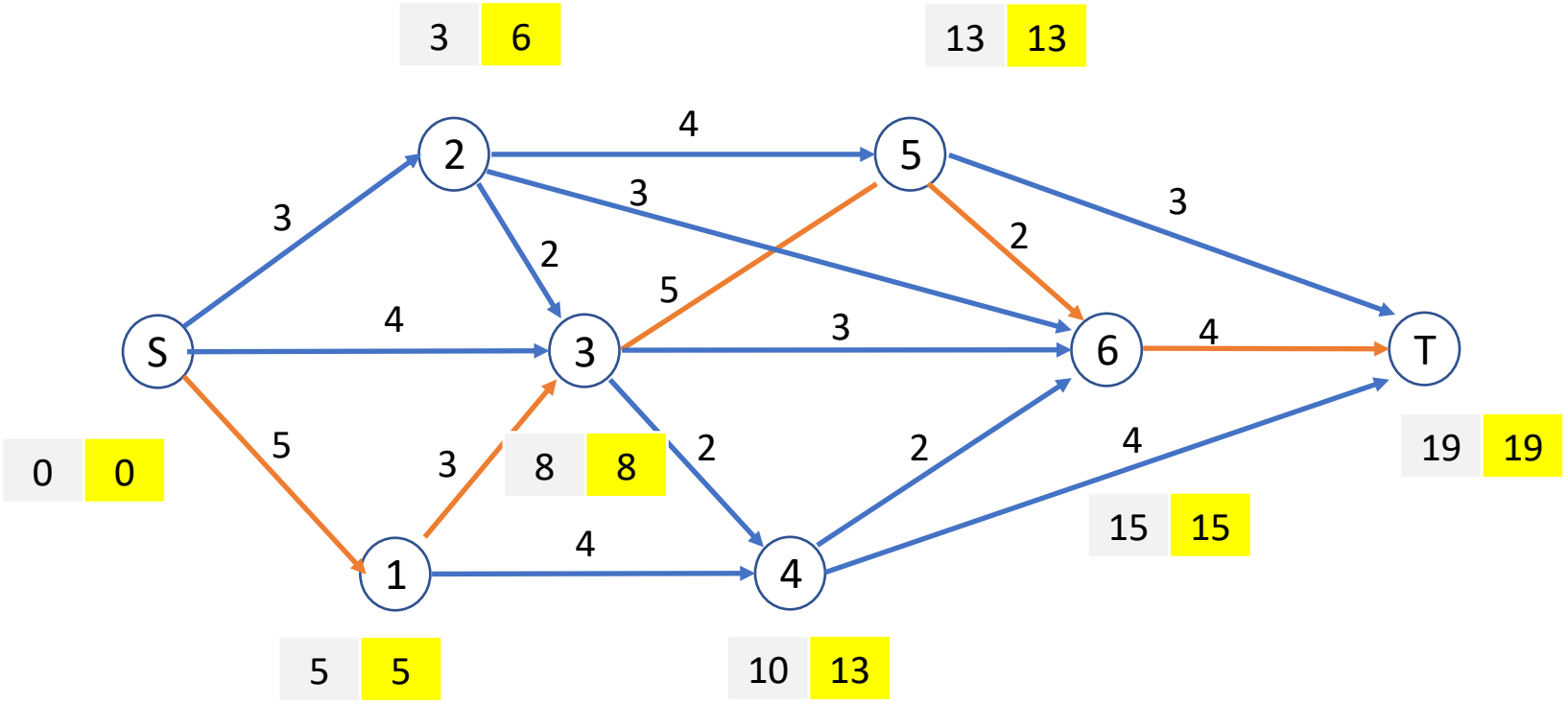
❖ PERT

활동	a	m	b	μ_k	σ_k^2	활동	a	m	b	μ_k	σ_k^2
A	2	3	4	3	0.11	I	1	3	5	3	0.44
B	2	4	6	4	0.44	J	1	2	3	2	0.11
C	4	5	6	5	0.11	K	3	4	5	4	0.11
D	1	2	3	2	0.11	L	1	2	3	2	0.11
E	1	3	5	3	0.44	M	1	2	3	2	0.11
F	3	4	5	4	0.11	N	1	3	5	3	0.44
G	2	3	4	3	0.11	O	3	4	5	4	0.11
H	2	5	8	5	1	P	2	4	6	4	0.44

6.8 CPM/PERT

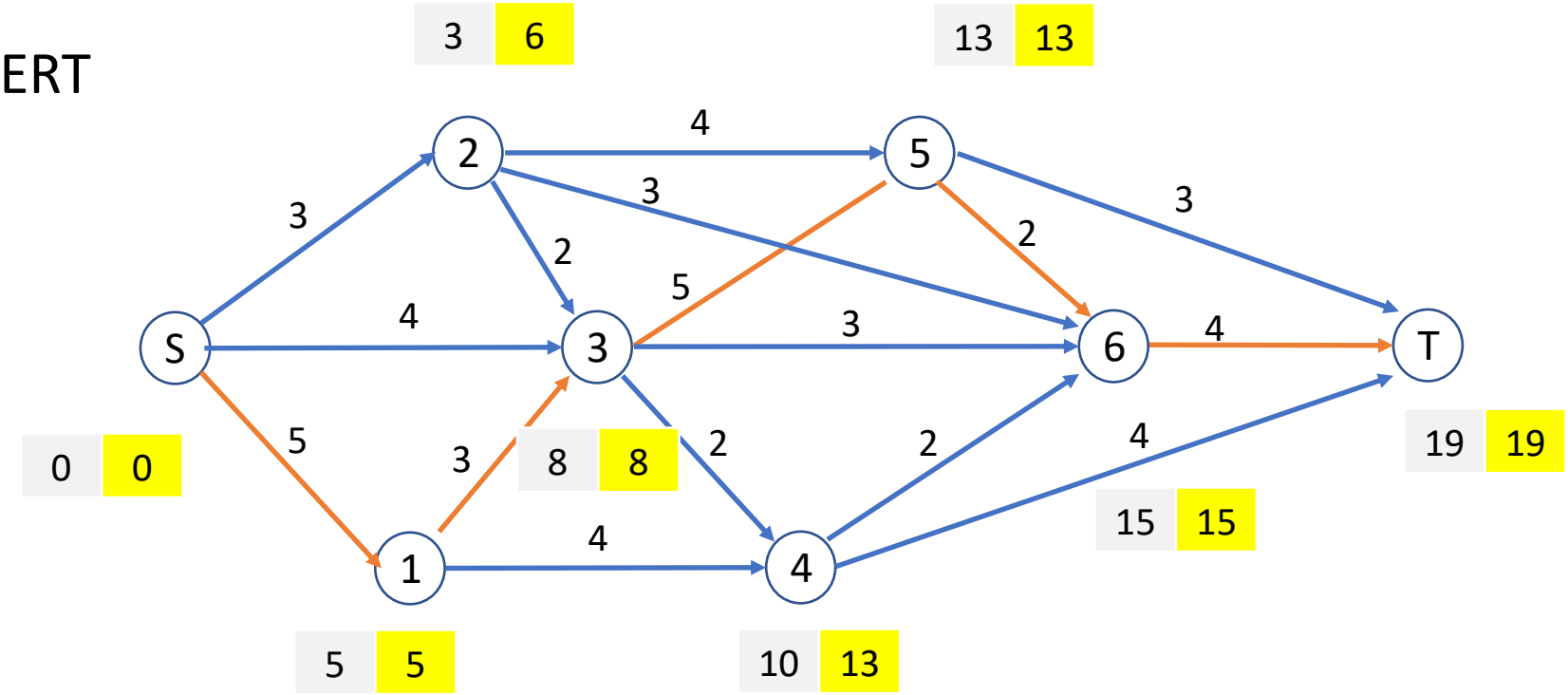
❖ PERT

주경로 기대기간 = 19
주경로 분산 = 1.78



6.8 CPM/PERT

❖ PERT



프로젝트가 21일에 완료될 확률

$$P(X \leq 21) = P\left(\frac{X-\mu}{\sigma} \leq \frac{21-19}{1.33}\right) = P(Z \leq 1.503) \cong 0.9332$$

교육부 Ministry of Education

NRF 한국연구재단 National Research Foundation of Korea

thank you

본 과제(결과물)는 교육부와 한국연구재단의 재원으로 지원을 받아 수행된
디지털신기술인재양성 혁신공유대학사업의 연구결과입니다.

Convergence and Open Sharing System COSS 인공지능

전남대학교 CHONNAM NATIONAL UNIVERSITY

성균관대학교 SUNGKYUNKWAN UNIVERSITY

KNU 경북대학교 KYUNGPOOK NATIONAL UNIVERSITY

전주대학교 JEONJU UNIVERSITY

서울시립대학교 UNIVERSITY OF SEOUL

서울과학기술대학교 SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

영진전문대학교 YEUNGJIN UNIVERSITY