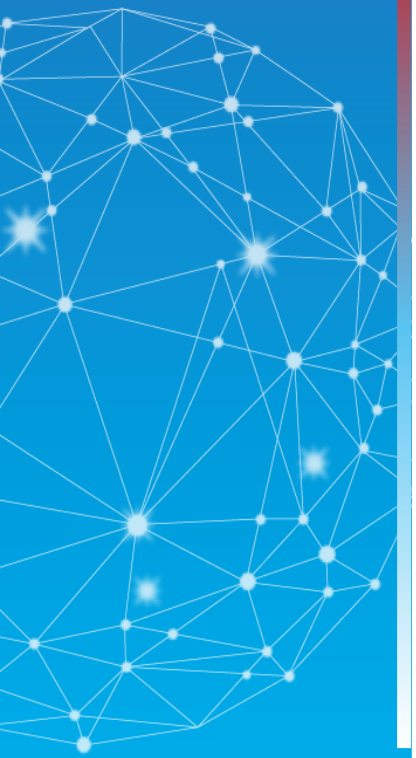
교육부 Ministry of Education

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

6.1 그래프 개요

❖ 그래프

- 그래프(도표)는 직선, 곡선, 도형 등 그래픽의 요소에 의해 시각화된 차트를 말하며 넓은 뜻으로는 다이어그램도 포함
- 함수의 그래프는 주어진 함수가 나타내는 직선이나 곡선

- 그래프는 수학에서 꼭짓점의 집합과 변의 집합을 이루는 순서쌍
- 그래프 (자료 구조)
  - Vertex(정점)와 edge(정점과 정점을 연결하는 간선)로 구성된 한정된 자료구조를 의미
  - 추상적 자료구조(리스트, 스택, 큐, 트리, 그래프)

6.1 그래프 개요

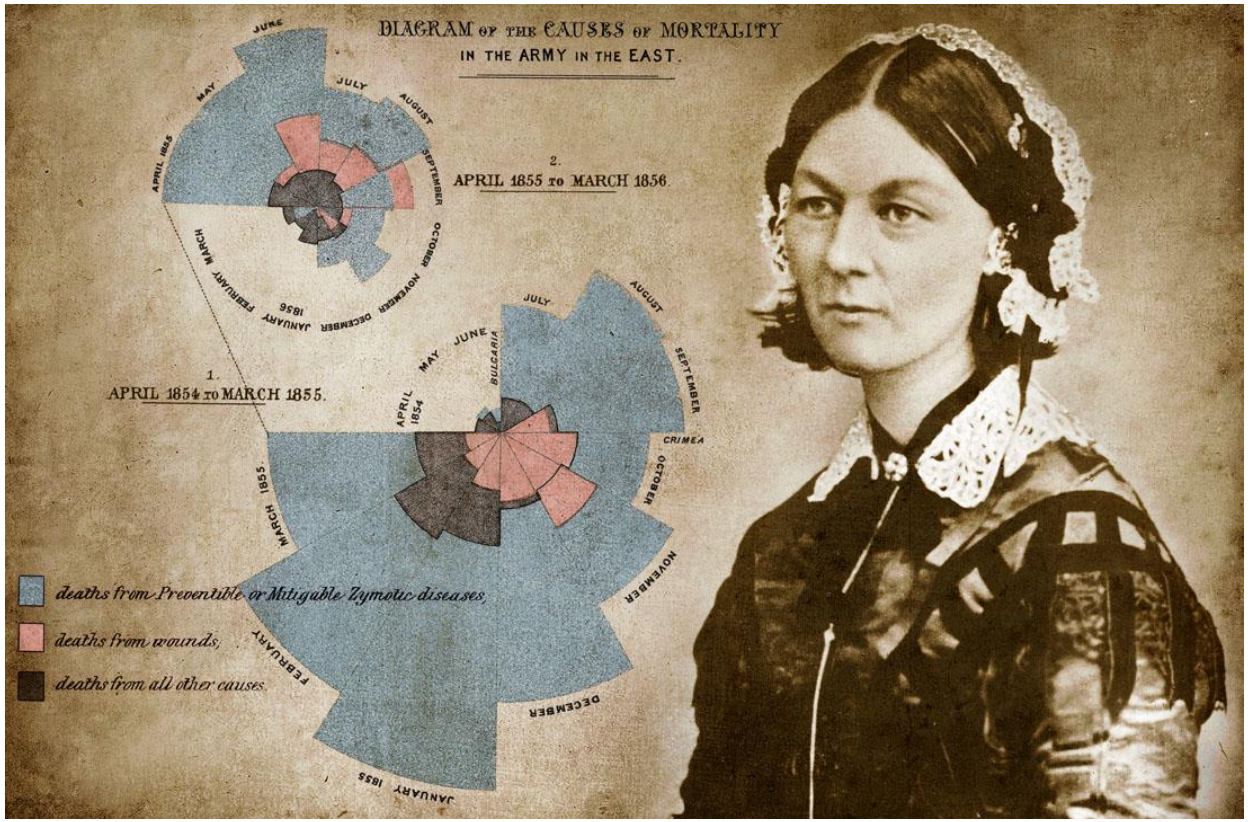
❖ 그래프(도표)

➢ 플로렌스 나이팅게일의 장미 다이어그램

✓ 크리미아 전쟁(1853.10-1856.2, 러시아와 오스만제국 동맹)

사망률:

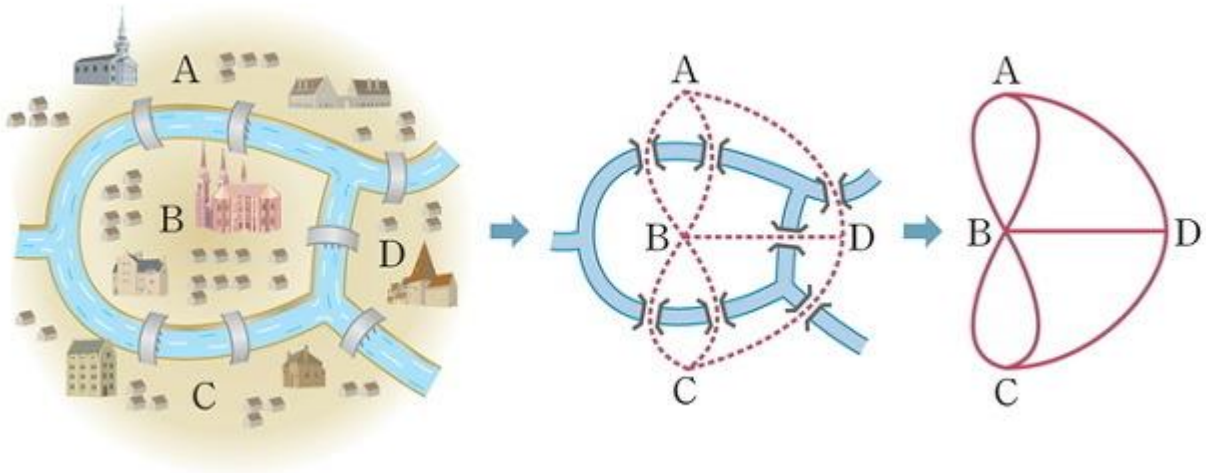
42% -> 2%



6.1 그래프 개요

❖ 그래프(수학)

➢ 쾨니히스베르크의 다리



➢ 1736년 오일러(그래프 안에서 경로나 순환을 찾는 방법 연구)

✓ 오일러 경로: G(V, E)의 모든 변을 꼭 한 번씩 지나는 경로(홀수차수 0 또는 2개)

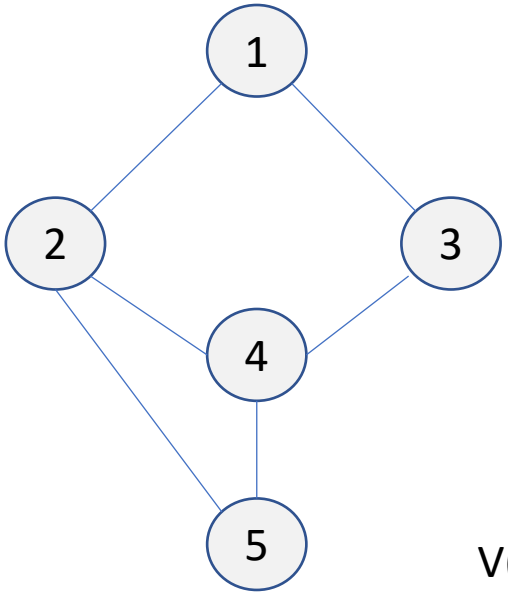
✓ 오일러 회로: 정점 v에서 시작해 모든 변을 꼭 한번씩 지나 v 로 돌아오는 경로

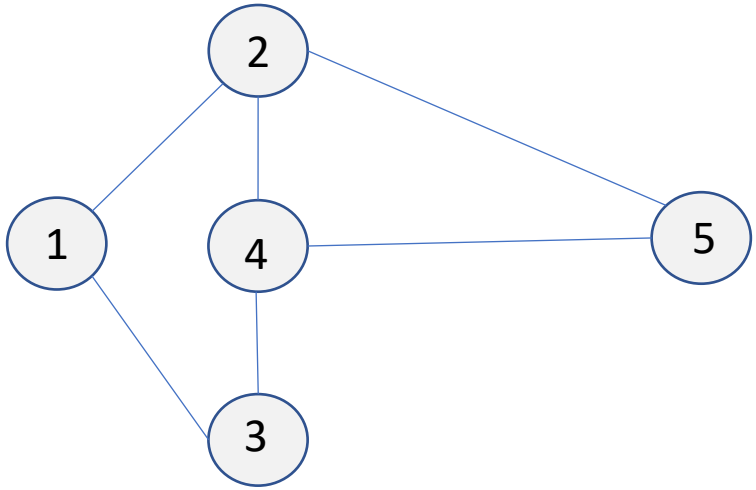
✓ 오일러 그래프: 오일러 회로를 포함하는 그래프(모든 꼭지점차수 짝수)

6.1 그래프 개요

❖ 그래프의 수학적 정의

- ➢ 그래프 : G(V, E)
  - ✓ V(G): 정점(vertex) 또는 노드(node)의 집합
  - ✓ E(G): 간선(edge) 또는 링크(link)의 집합(정점들 간의 관계 의미)



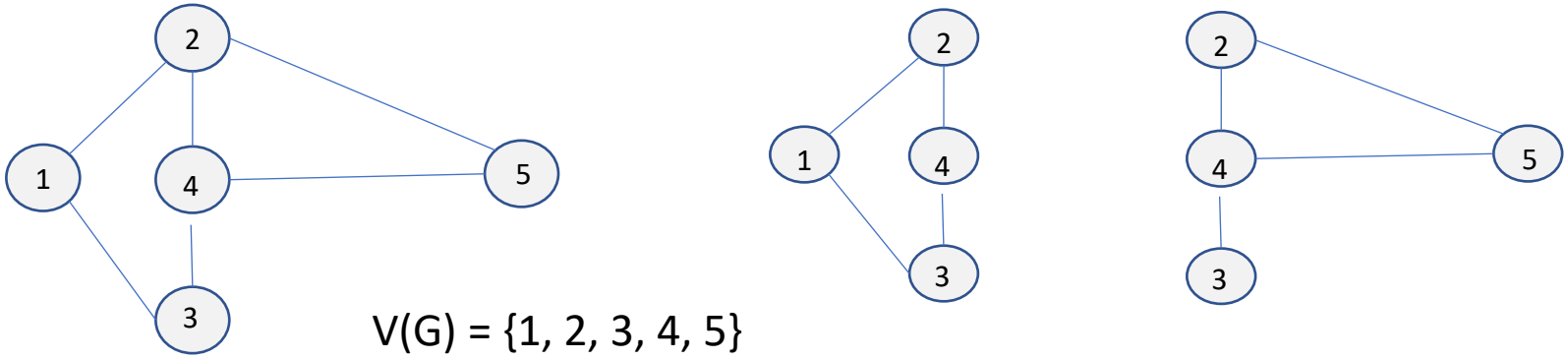


V(G) = {1, 2, 3, 4, 5}
E(G) = {(1,2), (1,3), (2,4), (2,5), (3,4), (4,5)}

6.1 그래프 개요

❖ 그래프의 종류

- ➢ 간선의 종류에 따라
  - ✓ 무방향 그래프(undirected graph) : (1, 2) = (2, 1)
  - ✓ 방향 그래프(directed graph) : <A, B> ≠ <B, A>
- ➢ 가중치 그래프, 네트워크
  - ✓ 간선에 비용이나 가중치가 할당된 그래프
- ➢ 부분 그래프

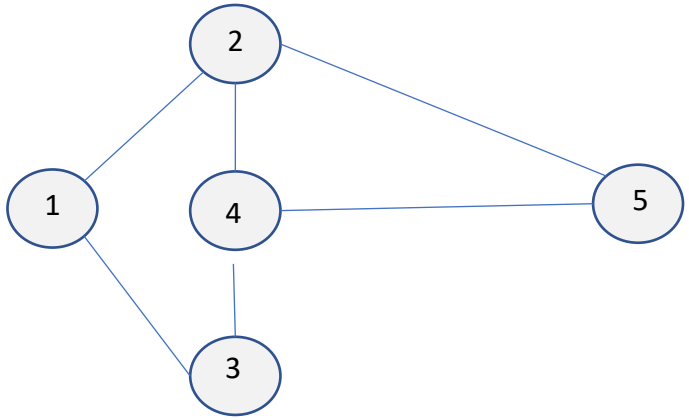


V(G) = {1, 2, 3, 4, 5}
E(G) = {(1,2), (1,3), (2,4), (2,5), (3,4), (4,5)}

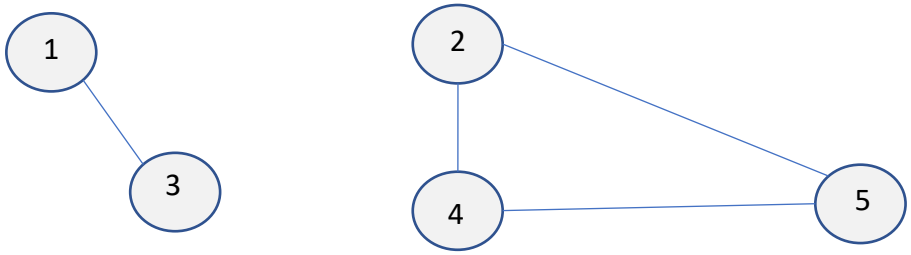
6.1 그래프 개요

❖ 그래프의 용어

- ➢ 인접 정점 : 간선에 의해 직접 연결된 정점
- ➢ 차수(degree) : 정점에 연결된 간선의 수
  - ✓ 무방향 그래프
    - 차수의 합은 간선 수의 2배
  - ✓ 방향 그래프
    - 진입차수, 진출차수
    - 모든 진입(진출) 차수의 합은 간선의 수



- ➢ 연결성분 : 그래프에서 정점들이 서로 연결되어 있는 부분



2개 연결성분

[1,2], [2,4,5]

6.1 그래프 개요

❖ 그래프의 용어

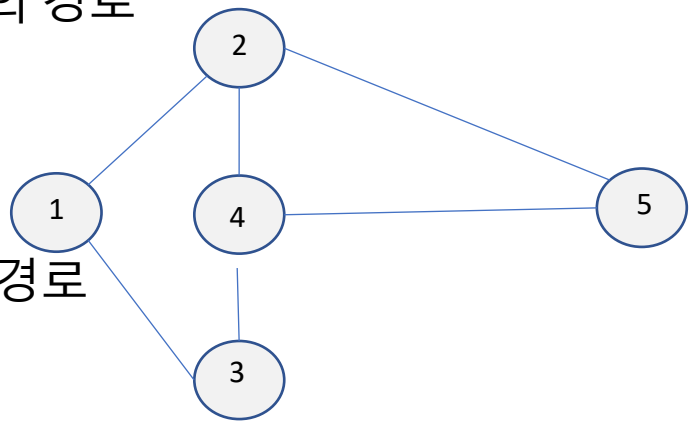
➢ 그래프의 경로(path)

✓ 무방향 그래프의 정점 1로부터 정점 5까지의 경로

- 정점의 나열 1,2, 5
- 반드시 간선 (1, 2), (2, 5) 존재해야 함

✓ 방향 그래프의 정점 1로부터 정점 5까지의 경로

- 정점의 나열 1, 2, 4, 5
- 반드시 간선 <1, 2>, <2, 4>, <4, 5> 존재해야 함



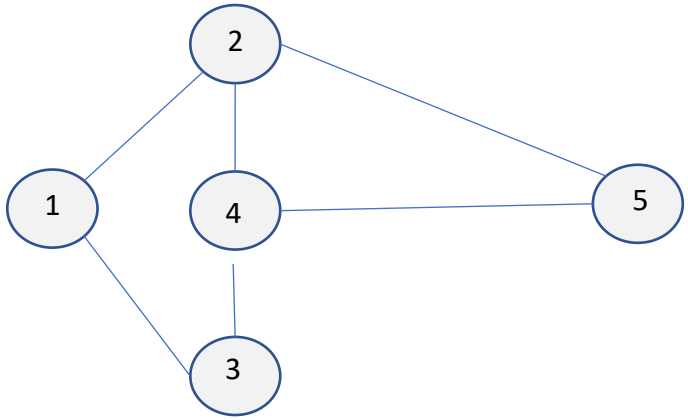
➢ 경로의 길이(length)

✓ 경로를 구성하는데 사용된 간선의 수

6.1 그래프 개요

❖ 그래프의 용어

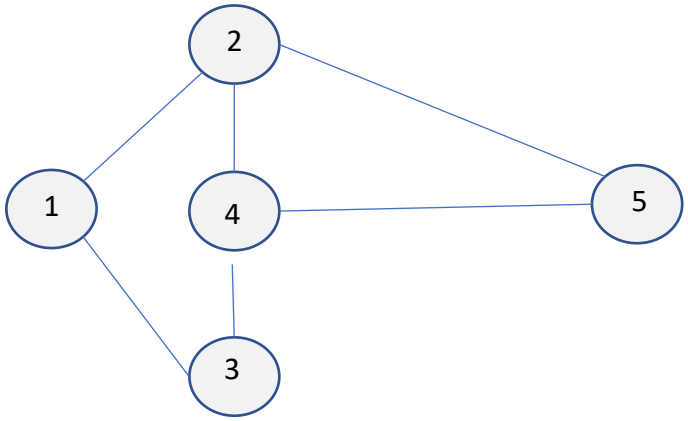
- ➢ 단순 경로(simple path)
  - ✓ 경로 중에서 반복되는 정점이 없는 경로 (1,2,4,5)
- ➢ 사이클(cycle)
  - ✓ 시작 정점과 종료 정점이 동일한 경로(1,2,4,3,1)
- ➢ 연결 그래프(connected graph)
  - ✓ 모든 정점들 사이에 경로가 존재하는 그래프
- ➢ 트리(tree)
  - ✓ 사이클을 가지지 않는 연결 그래프
- ➢ 완전 그래프(complete graph)
  - ✓ 모든 정점 간에 간선이 존재하는 그래프
  - ✓ n개의 정점을 가진 무방향 완전그래프의 간선의 수 = n×(n-1)/2



6.1 그래프 개요

❖ 그래프 표현

- ➢ 그래프를 자료구조로서 저장하는 방법
  - ✓ 인접 행렬(Adjacency Matrix)
  - ✓ 인접 리스트(Adjacency List)
- ➢ 인접행렬
  - ✓ N개의 정점을 가진 그래프의 인접 행렬은 2차원 NxN 리스트에 저장
  - ✓ 정점들을 0, 1, 2,…, N-1로 하여, 정점 i와 j 사이에 간선이 없으면 a[i][j] = 0, 간선이 있으면 a[i][j] = 1로 표현
  - ✓ 가중치 그래프는 1 대신 가중치 저장



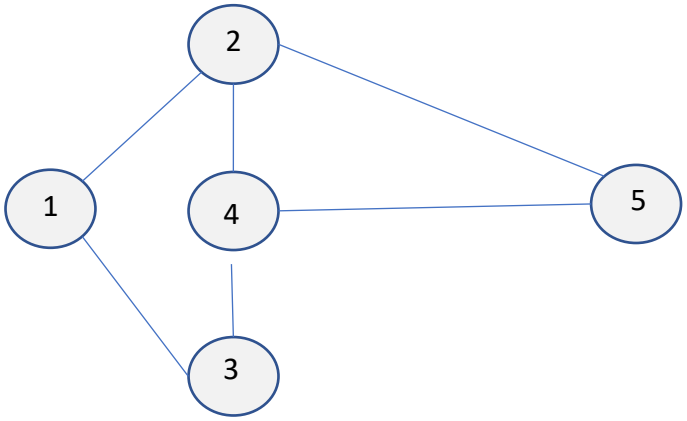
	1	1		
1			1	1
1			1	
	1	1		1
	1		1	

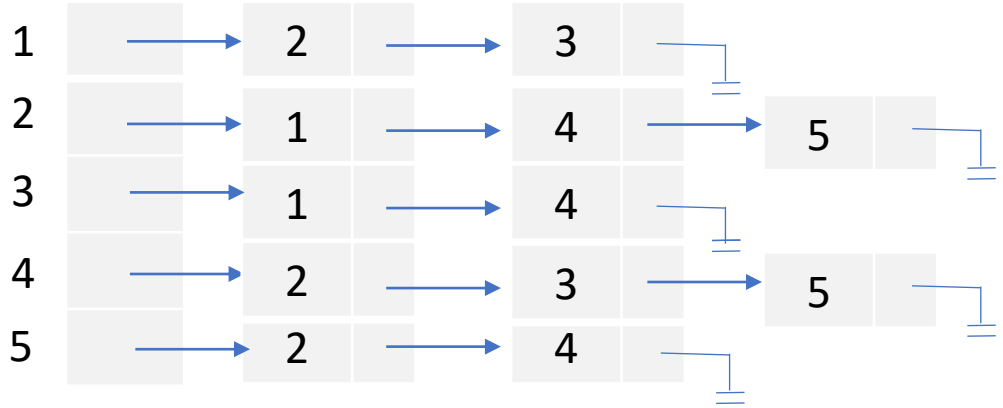
6.1 그래프 개요

❖ 그래프 표현

➢ 인접리스트

- ✓ n 개의 연결리스트로 그래프를 표현
- ✓ 각 정점에 대하여 정점과 연결된 간선을 한 개의 연결리스트에 저장
- ✓ 전체 연결리스트는 n개가 되며, 이 연결리스트를 배열로 표현





vertex → link

6.1 그래프 개요

❖ 그래프 표현

- ➢ 실세계의 그래프는 대부분 정점의 평균 차수가 작은 희소 그래프(Sparse Graph)이다.
- ➢ 희소그래프의 간선 수는 최대 간선 수인 N(N-1)/2보다 훨씬 작으므로 인접리스트에 저장하는 것이 매우 적절
  - ✓ 무방향 그래프를 인접리스트를 사용하여 저장할 경우 간선 1 개당 2개의 Edge 객체를 저장하고, 방향 그래프의 경우 간선 1 개당 1개의 Edge 객체만 저장하기 때문
- ➢ 조밀 그래프(Dense Graph): 간선의 수가 최대 간선 수에 근접한 그래프

6.1 그래프 개요

❖ 그래프 탐색

➢ 그래프 자료구조와 연산모델

✓ 그래프를 표현하고 그래프에 관한 연산을 구현

✓ 그래프 자료구조 = 그래프 자료선언 + 그래프 연산

- 그래프 자료구조는 자료의 선언과 자료에 대한 연산으로 구성
- 그래프 자료의 선언과 저장은 인접 행렬이나 인접리스트로 표현
- 그래프 연산은 자료 탐색(traversal)과 갱신에 관한 연산으로 이루어 짐
- 탐색은 원하는 노드를 1개 혹은 일부를 찾는 작업
- 갱신은 정점과 간선의 삽입(insert), 삭제(delete), 수정을 말함

6.1 그래프 개요

❖ 그래프 탐색

➢ 그래프 ADT(추상 자료형)

✓ 데이터: 정점과 간선의 집합

✓ 연산

- isEmpty() : 그래프가 공백 상태인지 확인
- countVertex() : 정점의 수
- countEdge() : 간선의 수
- getEdge(u,v) : 정점 u 에서 정점 v 로 연결된 간선
- Degree(v) : 정점 v 의 차수
- Adjacent(v) : 정점 v 에 인접한 모든 정점의 집합을 반환
- insertVertex(v), insertEdge(u,v), deleteVertex(v), deleteEdge(u,v,)

6.1 그래프 개요

❖ 그래프 탐색

- ➢ 그래프의 각 정점을 방문하는 것으로 그래프 관련 연산중 가장 중요한 것
- ➢ 탐색에서 노드의 방문 순서에 따른 방법
  - ✓ 깊이우선탐색 – DFS (Depth First Search)
    - 갈수있는데 까지 가보는 방문 방법
    - 트리의 전위탐색 방법을 그래프에 적용한 것 (preorder tree traversal)
  - ✓ 너비우선탐색 – BFS (Breadth First Search)
    - 거리가 가까운 곳부터 가보는 방문 방법
    - 트리의 레벨우선탐색을 그래프에 적용한 것(level order tree traversal)

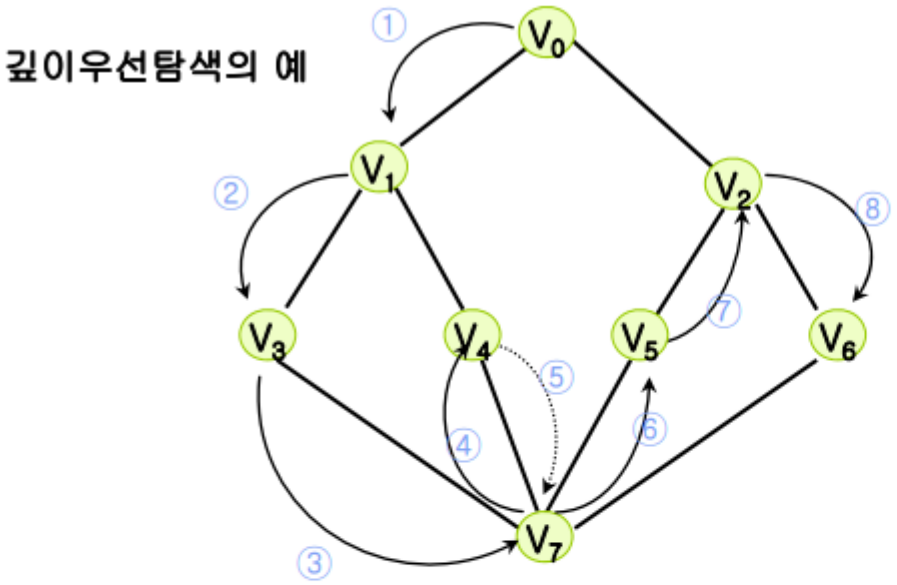
6.1 그래프 개요

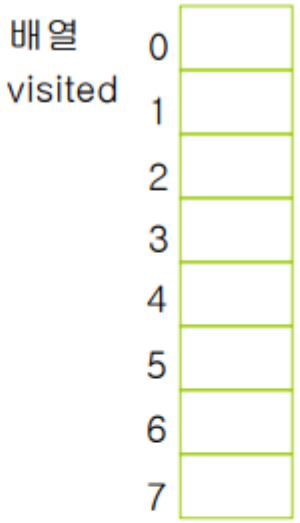
❖ 그래프 탐색: 깊이우선탐색 (Depth First Search)

- ➢ 방법
  - ✓ 한 방향으로 끝까지 가다가 더 이상 갈 수 없게 되면 가장 가까운 갈림길로 돌아와서 다른 방향으로 다시 탐색 진행
  - ✓ 탐색 중 방문 가능한 곳은 스택 자료를 이용하여 저장
    - 되돌아가기 위해서는 스택 필요
    - 순환함수 호출로 묵시적인 스택 이용
- ➢ 절차
  - ✓ 1단계 : 하나의 노드를 택함
  - ✓ 2단계 : 노드를 방문하여 필요한 작업을 한 다음 연결된 다음 노드를 찾고, 현재 방문 노드는 스택에 저장. 2단계를 반복하면서 방문을 계속. 막히면 3단계로 감
  - ✓ 3단계 : 더 이상 방문할 노드가 없으면 스택에서 노드를 빼내 다음 방문 노드를 찾아 2)번과정을 다시 반복

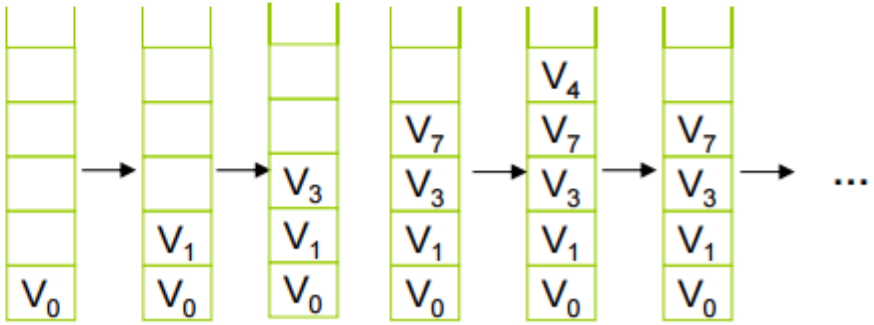
6.1 그래프 개요

❖ 그래프 탐색: 깊이우선탐색 (Depth First Search)





- ①②③④⑥⑦⑧ : 방문
- ⑤ : 되돌아옴
- 방문할 곳이 여러 곳일 때 노드번호가 작은 쪽 방문



스택의 변화

6.1 그래프 개요

❖ 그래프 탐색: 너비우선탐색(breadth first search)

➢ 방법

- ✓ 시작 정점으로부터 가까운 정점을 먼저 방문하고 멀리 떨어져 있는 정점을 나중에 방문하는 순회 방법
- ✓ 어느 지점에서 가까운 곳을 방문하고 다음 방문 가능 지점은 모두 저장해 둠
- ✓ 다음 방문지는 항상 저장해둔 곳에서 순서대로 꺼냄
- ✓ 큐 자료구조를 이용하여 다음 방문지를

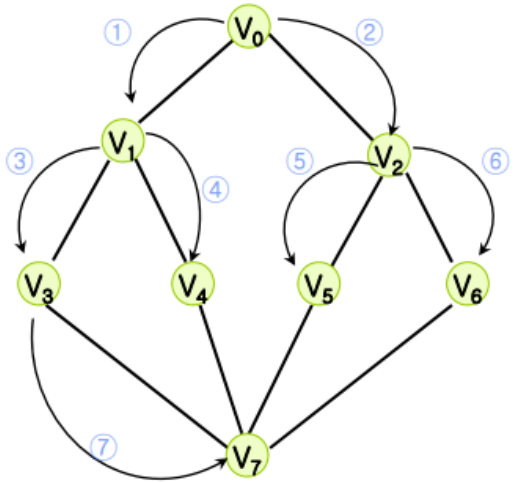
➢ 절차

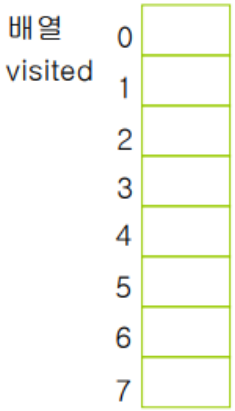
- ✓ 1단계 : 하나의 노드를 택함
- ✓ 2단계 : 노드를 방문하여 필요한 작업을 한 다음 연결된 다음 노드를 찾고, 노드들을 큐에 저장. 더 이상 방문할 곳이 없으면 3단계로 감
- ✓ 3단계 : 큐의 맨 앞의 노드를 빼내 2단계를 반복

6.1 그래프 개요

❖ 그래프 탐색: 너비우선탐색(breadth first search)

➢ 예

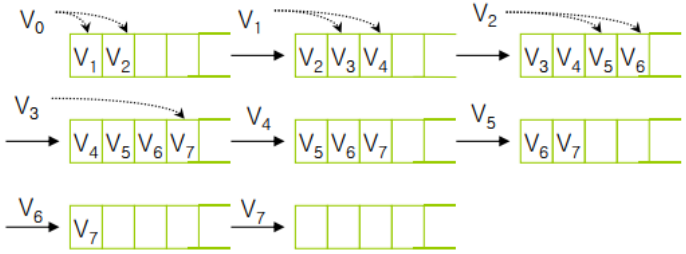




①②③④⑤⑥⑦ : 방문

방문할 곳이 여러 곳일 때 노드번호가 작은 쪽 방문

너비 우선 탐색의 예 – 큐의 모양



6.1 그래프 개요

❖ 그래프 탐색

➢ 성능

✓ 정점의 수 n, 간선의 수 e

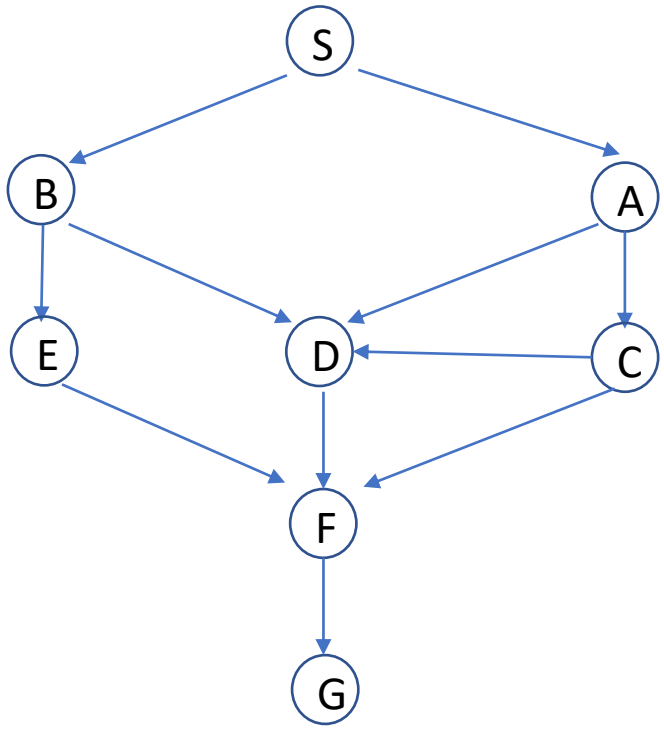
	인접행렬	인접리스트
메모리 공간	n^2	n+2e
getEdge(u, v)	O(1)	O(정점 u의 차수)
Degree(u)	O(n)	O(정점 u의 차수)
Adjacent(u)	O(n)	O(정점 u의 차수)
모든 간선의 수	$O(n^2)$	O(n+e)

❖ 위상정렬

➢ 방향 그래프에 대해 정점들의 선행 순서를 위배하지 않으면서 모든 정점을 나열하는 것

✓ 싸이클이 없는 방향그래프에서 정점을 선형순서(즉, 정점들을 일렬)로 나열하는 것

과목번호	과목명	선수과목
S	인공지능 개론	없음
A	파이썬	S
B	인공지능 수학	S
C	알고리즘	A
D	최적화이론	A,B,C
E	인공지능 통계	B
F	기계학습	C,D,E
G	딥러닝	F



6.1 그래프 개요

❖ 위상정렬

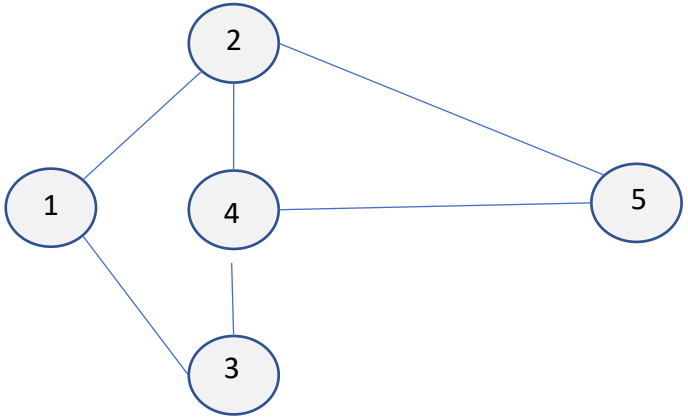
- ➢ 일반적으로 작업(Task)들 사이에 의존관계가 존재할 때 수행 가능한 작업 순서를 도식화하는데에 위상 정렬을 사용
- ➢ 수행시간
  - ✓ 위상 정렬 알고리즘의 수행 시간은 DFS의 수행 시간과 동일한 O(N+M)
  - ✓ 기본적으로 DFS를 수행하며 추가로 소요되는 시간은 정점을 리스트에 저장하고, 모든 탐색이 끝나면 리스트를 역순으로 만드는 시간으로 이는 O(N)
  - ✓ 따라서 위상 정렬 알고리즘의 수행 시간은

$$O(N+M) + O(N) = O(N+M)$$

6.2 네트워크 모형 개요

❖ 네트워크 정의

- ➢ 호(arc)와 호들에 의해 연결되는 마디(node)들의 집합
  - ✓ 노드는 작업의 시작이나 완료를 의미하는 원으로 표현
  - ✓ 아크는 노드들을 연결하는 선으로 표현
  - ✓ G(N,A)
  - ✓ N = {1, 2, 3, 4, 5}
  - ✓ A = {(1,2), (1,3), (2,4), (2,5), (3,4), (4,5)}



6.2 네트워크 모형 개요

❖ 네트워크 모형의 등장과 발전

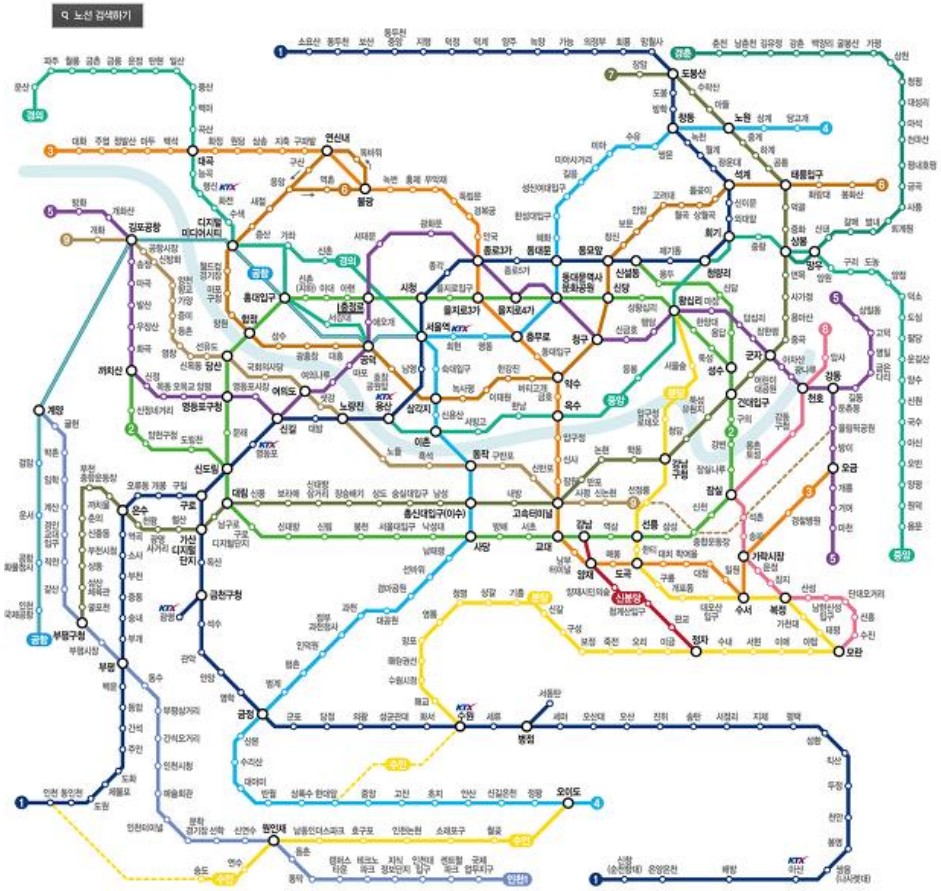
- ➢ 전자공학에서 회로의 분석에서 처음으로 개발되어 이용
- ➢ 경제 사회 시스템으로 확대
  - ✓ 현실세계의 다양한 의사결정 상황을 네트워크 형태로 구현하기가 용이
  - ✓ 시각적이고 효율적으로 의사결정상황을 표현함으로써 의사소통을 원활하게 함
- ➢ 네트워크 모형은 일련의 활동(가지)과 그 활동의 결과로서 나타내는 여러 단계(마디)를 선형그래프화 한 것
- ➢ 작업공정이나 프로젝트를 관리하는데 있어 각 활동의 일정계획을 최적화하고 과업이 계획대로 진행되고 있는지를 파악하는데 매우 효율적으로 활용됨

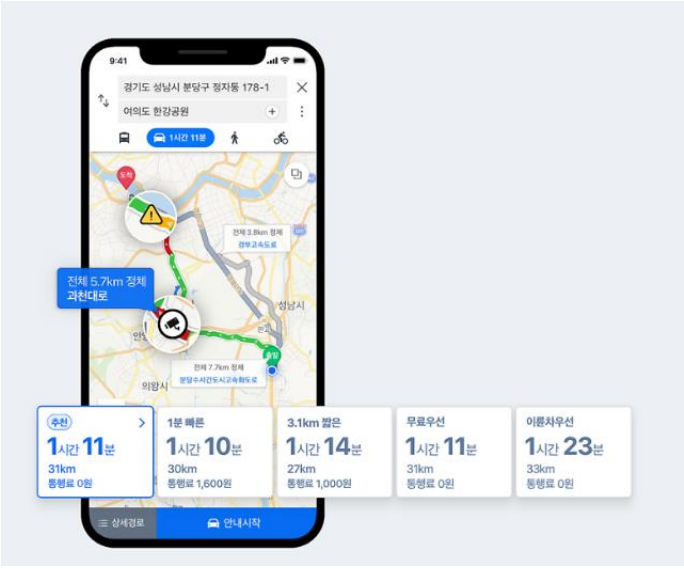
6.2 네트워크 모형 개요

❖ 네트워크 모형의 응용분야

6.2 네트워크 모형 개요

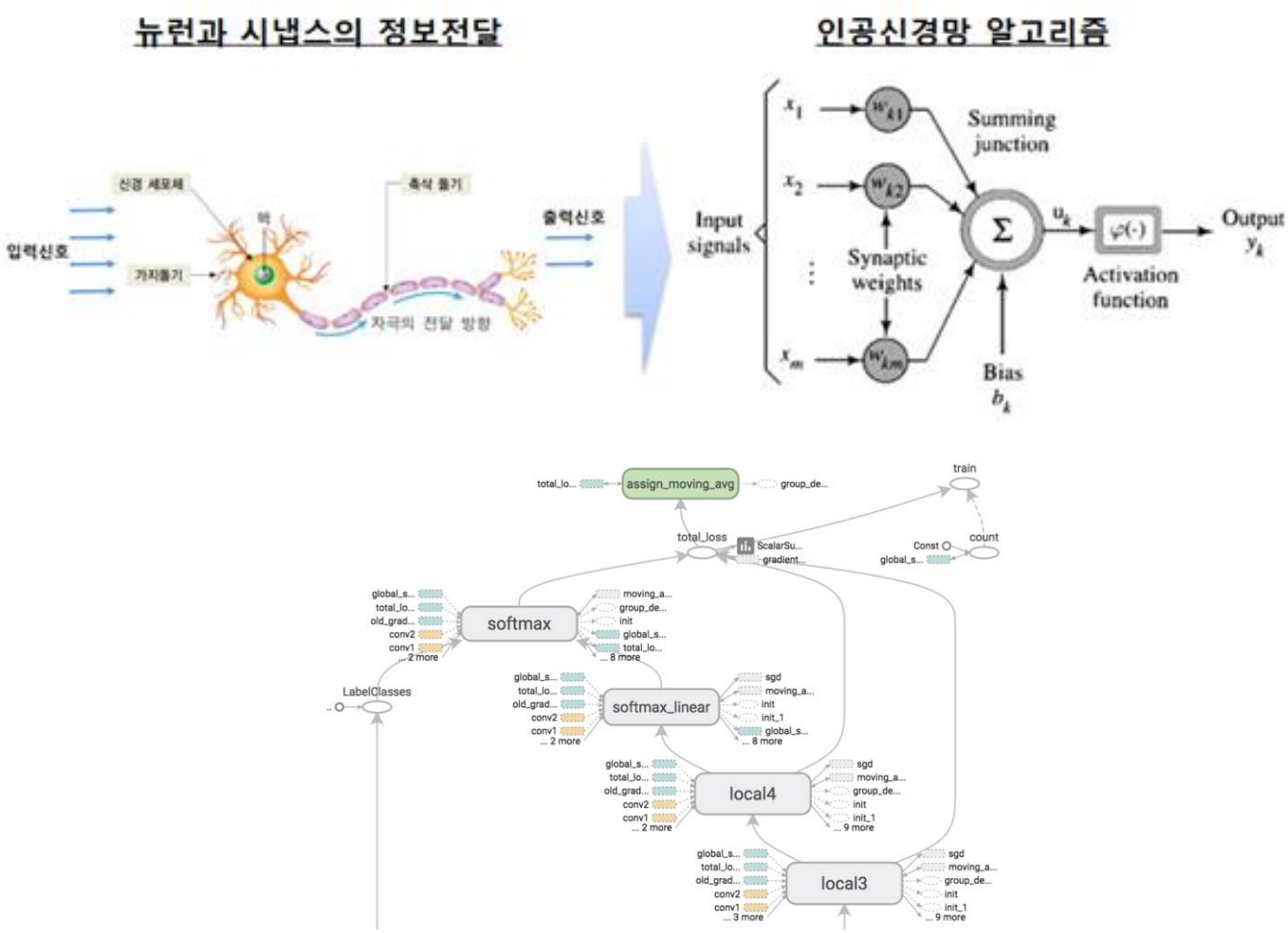
❖ 네트워크 모형의 응용분야





6.2 네트워크 모형 개요

❖ 네트워크 모형의 응용분야



6.2 네트워크 모형 개요

❖ 네트워크 최적화 알고리즘

- 최소걸침나무 알고리즘(minimum spanning tree algorithm)
- 최단경로 알고리즘(shortest path algorithm)
- 최대흐름 알고리즘(maximal flow algorithm)
- 최소비용 용량제약 네트워크 흐름 알고리즘(minimum-cost capacitated network flow algorithm)
- PERT-CPM

6.3 최소걸침나무 문제

❖ 걸침나무(Spanning Tree)

- 순환고리를 형성하지 않도록 하는 연결네트워크의 모든 마디들을 연결하는 나무
- 무방향 그래프 G = (V,E)의 일종으로써, 비 순환 (루프 순환이 없음) 이고, 모든 노드들이 연결되어 있어야 하며, 간선 갯수가 |V|-1 개 임

❖ 최소걸침나무 (최소신장트리)

- 최단거리로 연결되는 호들을 사용하여 네트워크의 마디들을 연결
- 하나의 연결성분으로 이루어진 무방향 가중치 그래프에서 간선의 가중치의 합이 최소인 신장 트리
- 통신망, 도로망, 상하수도망 등

6.3 최소걸침나무 문제

❖ 기호

- N = 마디들의 집합 = $\{1,2,\cdots,n\}$
- CN_k = 반복 k 에서 영구적으로 연결되는 마디들의 집합
- $\overline{CN_k}$ = 반복 k 에서 영구적으로 연결되지 않은 마디들의 집합

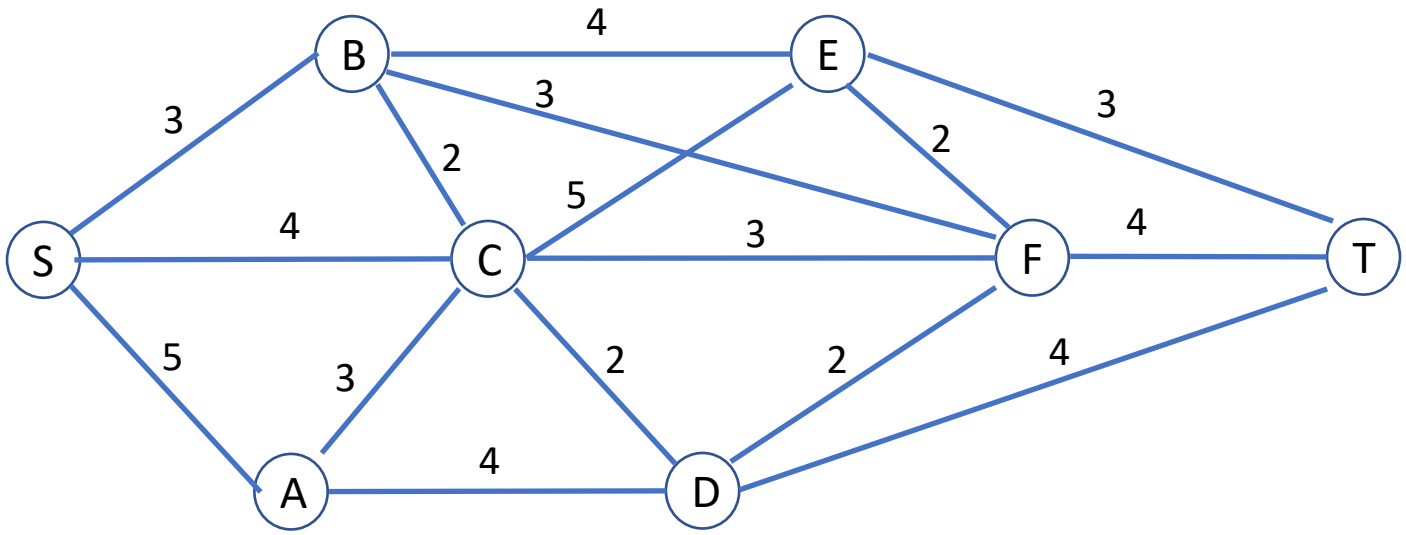
❖ 알고리즘

- 단계 0: $CN_0 = \emptyset, \overline{CN_0} = N$ 으로 놓는다.
- 단계 1: $\overline{CN_0}$ 내의 임의의 마디 m 를 선택하고 , CN_1 ={m} , $\overline{CN_1} = N - \{m\}$ 로 놓는다.
  그리고 $k = 2$ 로 놓는다.
- 단계2 : CN_{k-1} 와 최단거리의 호로 연결될 수 있는 $\overline{CN_{k-1}}$ 내의 마디 n를
  선택하고 , CN_k = CN_{k-1} +{n}, $\overline{CN_k}$ = $\overline{CN_{k-1}}$ -{n} 로 놓는다.
  만일 $\overline{CN_k} = \emptyset$ 이면, 이 절차를 멈추고,
  그렇지 않다면 $k = k + 1$ 로 놓고 이 단계를 반복

6.3 최소걸침나무 문제

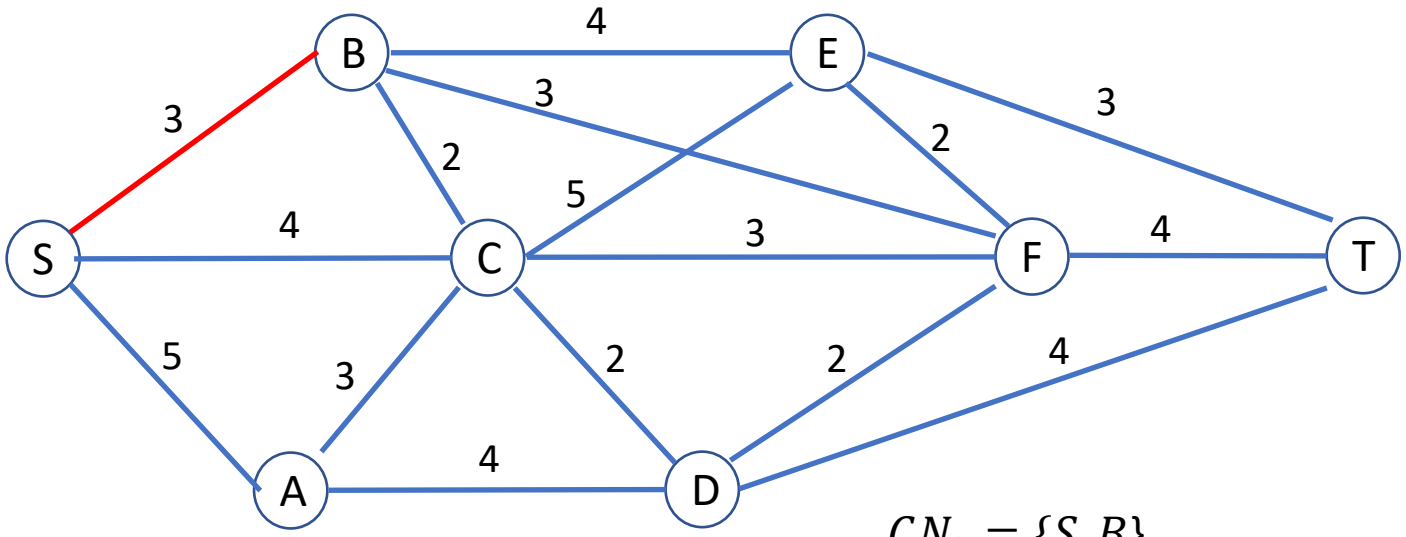
❖ 예제

다음 네트워크는 여러도시를 연결하는 인터넷망이며, 가지위의 숫자는 도시를 연결하는 인터넷망 구축 비용이라고 할 때, 모든 도시를 연결하는 가장 경제적인 인터넷망 설계방안?



❖ 알고리즘 적용

- 단계 0 : $CN_0 = \emptyset, \overline{CN_0} = N = \{\mathrm{S}, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}, \mathrm{T}\}$
- 단계 1 : $CN_1 = \{S\}, \ \overline{CN_0} = N - \{S\} = \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}, \mathrm{T}\}$
- 단계 2 :



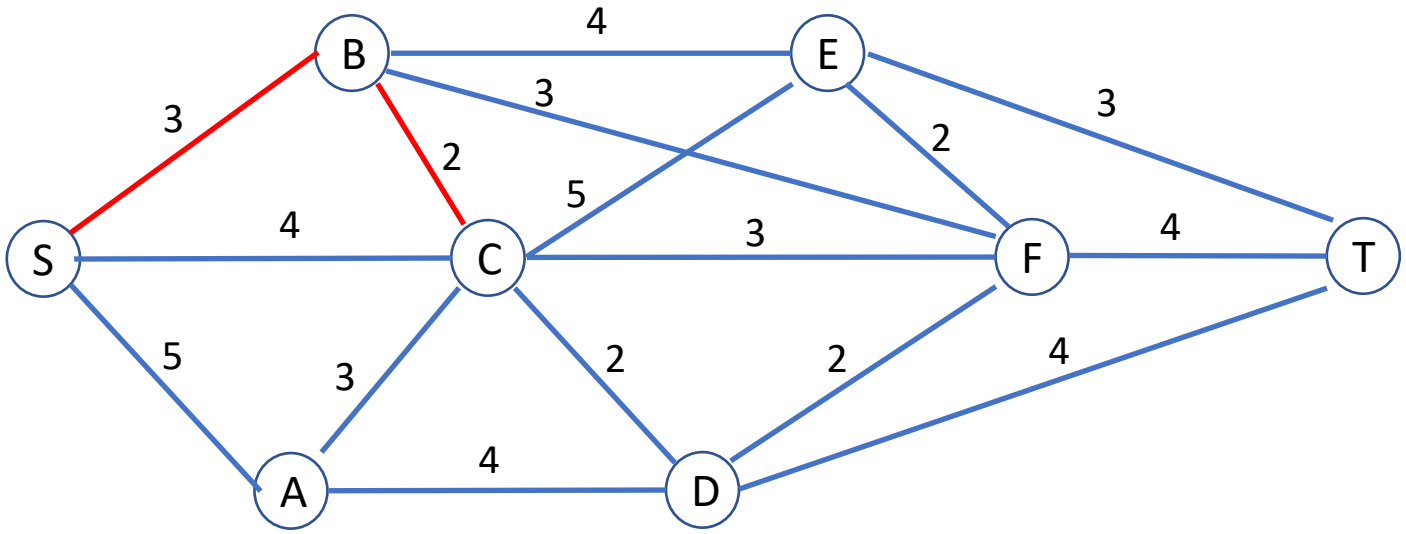
$CN_2 = \{S, B\}$

$\overline{CN_2} = N - \{S, B\} = \{\mathrm{A}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}, \mathrm{T}\}$

6.3 최소걸침나무 문제

❖ 알고리즘 적용

➢ 단계 2 :



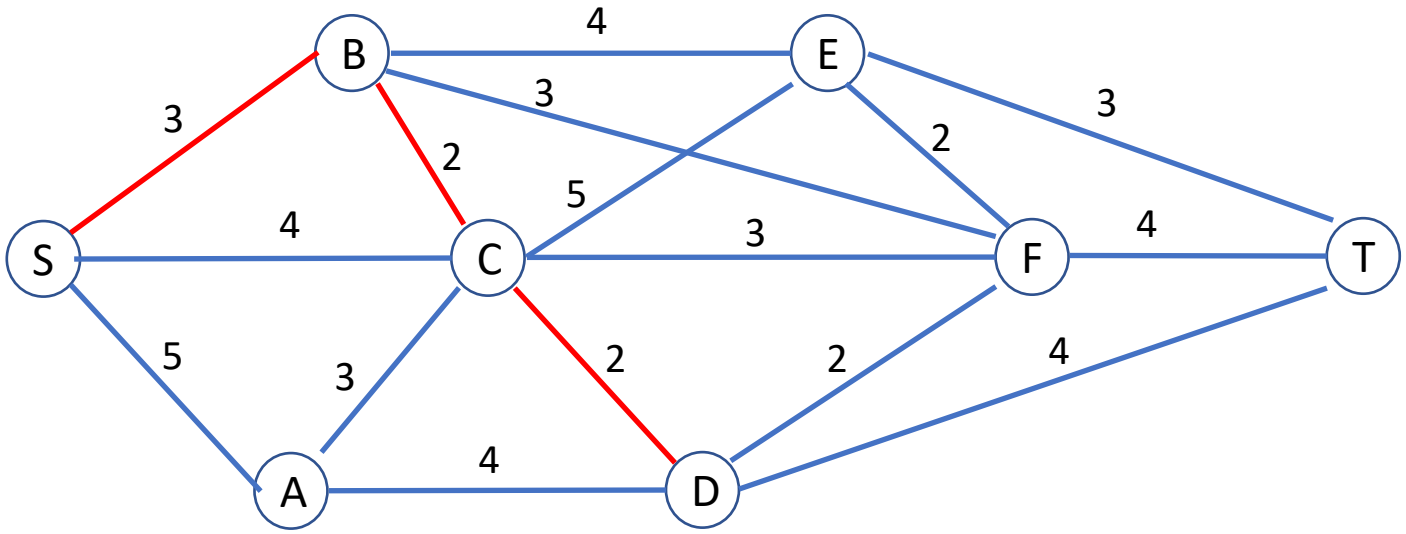
$$CN_3 = \{S, B, C\}$$

$$\overline{CN_3} = N - \{S, B, C\} = \{\mathrm{A}, \mathrm{D}, \mathrm{E}, \mathrm{F}, \mathrm{T}\}$$

6.3 최소걸침나무 문제

❖ 알고리즘 적용

➢ 단계 2 :



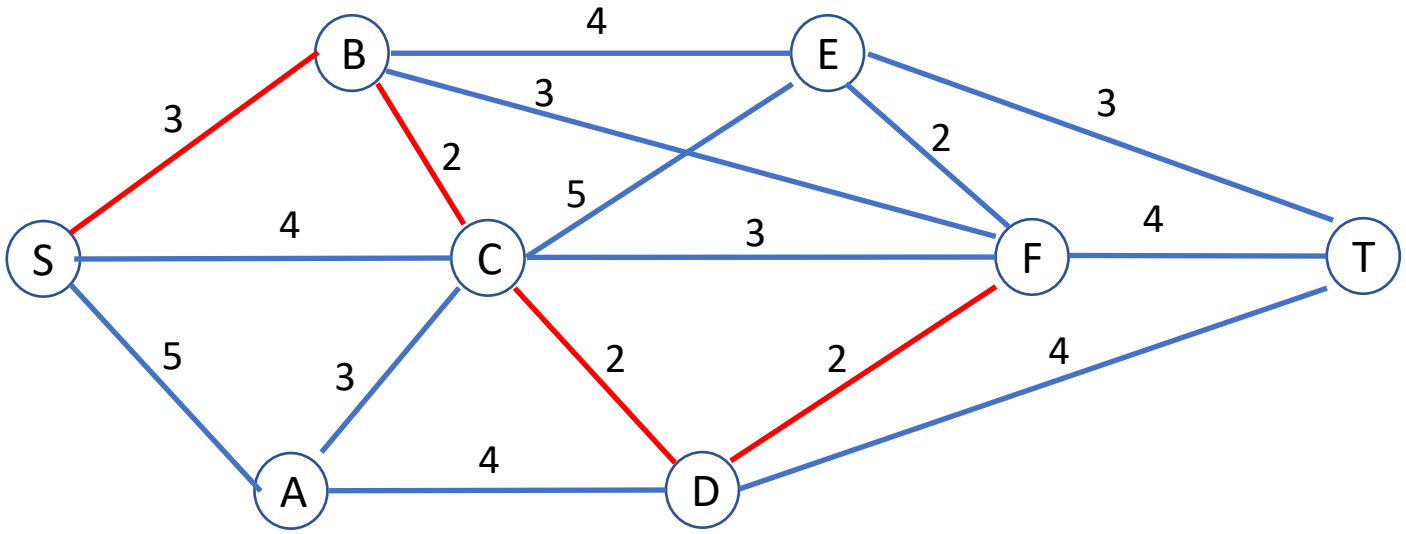
$$CN_4 = \{S, B, C, D\}$$

$$\overline{CN_4} = N - \{S, B, C, D\} = \{\mathrm{A}, \mathrm{E}, \mathrm{F}, \mathrm{T}\}$$

6.3 최소걸침나무 문제

❖ 알고리즘 적용

➢ 단계 2 :



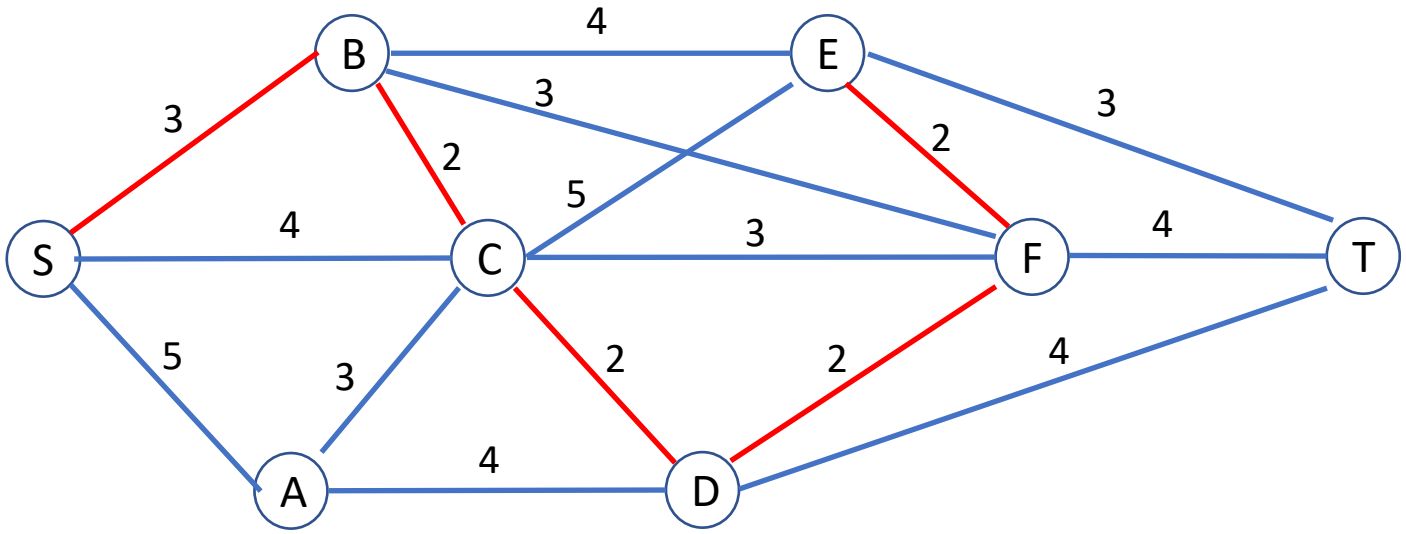
$$CN_5 = \{S, B, C, D, F\}$$

$$\overline{CN_5} = N - \{S, B, C, D, F\} = \{\mathrm{A}, \mathrm{E}, \mathrm{T}\}$$

6.3 최소걸침나무 문제

❖ 알고리즘 적용

➢ 단계 2 :



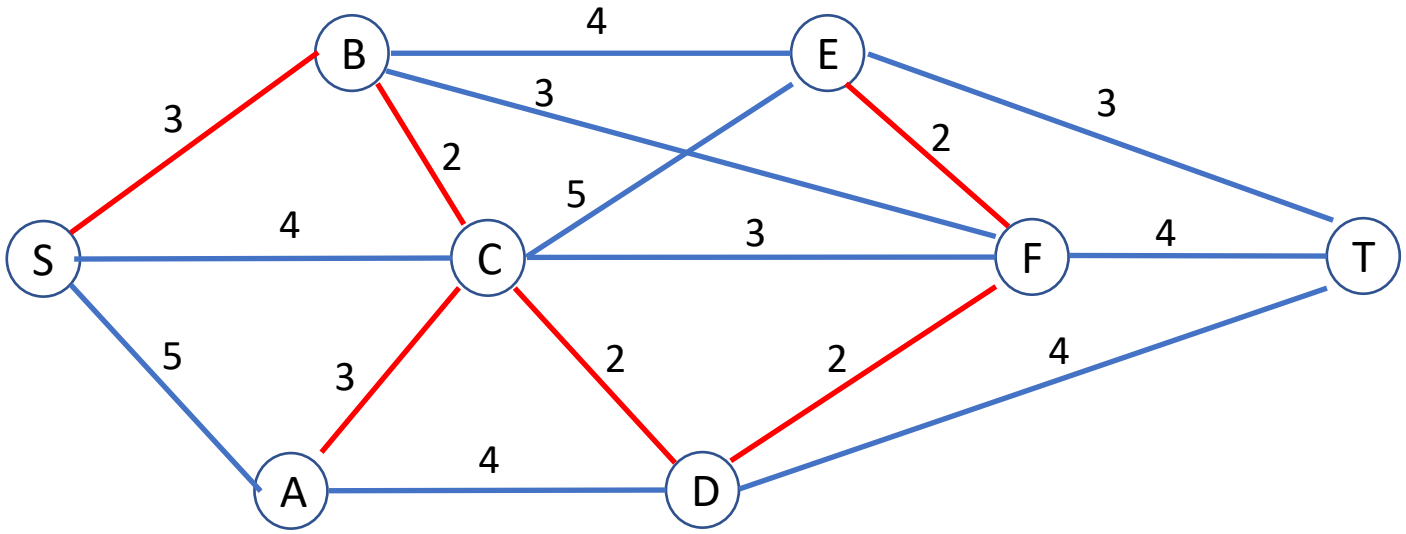
$$CN_6 = \{S, B, C, D, F, E\}$$

$$\overline{CN_6} = N - \{S, B, C, D, F, E\} = \{\mathrm{A}, \mathrm{T}\}$$

6.3 최소걸침나무 문제

❖ 알고리즘 적용

➢ 단계 2 :



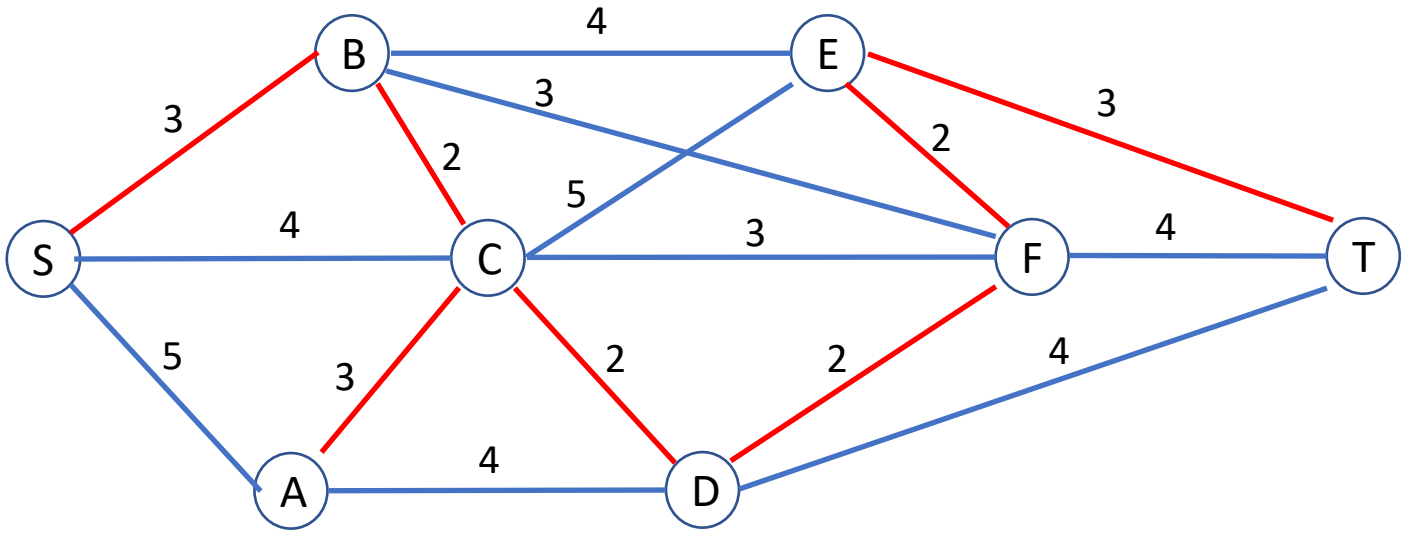
$$CN_7 = \{S, B, C, D, F, E, A\}$$

$$\overline{CN_7} = N - \{S, B, C, D, F, E, A\} = \{\mathrm{T}\}$$

6.3 최소걸침나무 문제

❖ 알고리즘 적용

➢ 단계 2 :



인터넷망의 구축비용
= 3+2+2+2+2+3+3=17

$$CN_8 = \{S, B, C, D, F, E, A, T\}$$

$$\overline{CN_7} = N - \{S, B, C, D, F, E, A, T\} = \emptyset$$

6.3 최소걸침나무 문제

❖ Kruskal 알고리즘

- 간선들을 가중치가 감소하지 않는 순서로 정렬한 후,
- 가장 가중치가 작은 간선을 트리에 추가하여 싸이클을 만들지 않으면 트리 간선으로 선택하고, 싸이클을 만들면 버리는 일을 반복하여 N-1개의 간선이 선택되었을 때 알고리즘을 종료

❖절차

가중치가 감소하지 않는 순서로 간선 리스트 L을 만든다.

while 트리의 간선 수 < N-1:

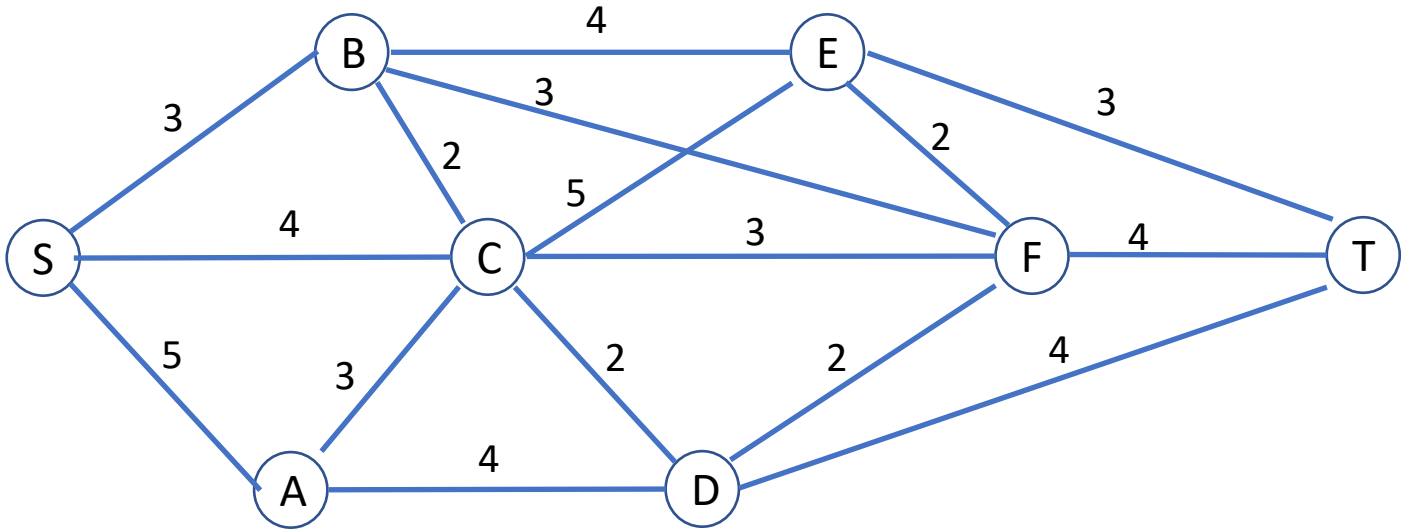
    L에서 가장 작은 가중치를 가진 간선 e를 가져오고, e를 L에서 제거

    if 간선 e가 T에 추가하여 싸이클을 만들지 않으면:

        간선 e를 T에 추가

6.3 최소걸침나무 문제

❖ Kruskal 알고리즘

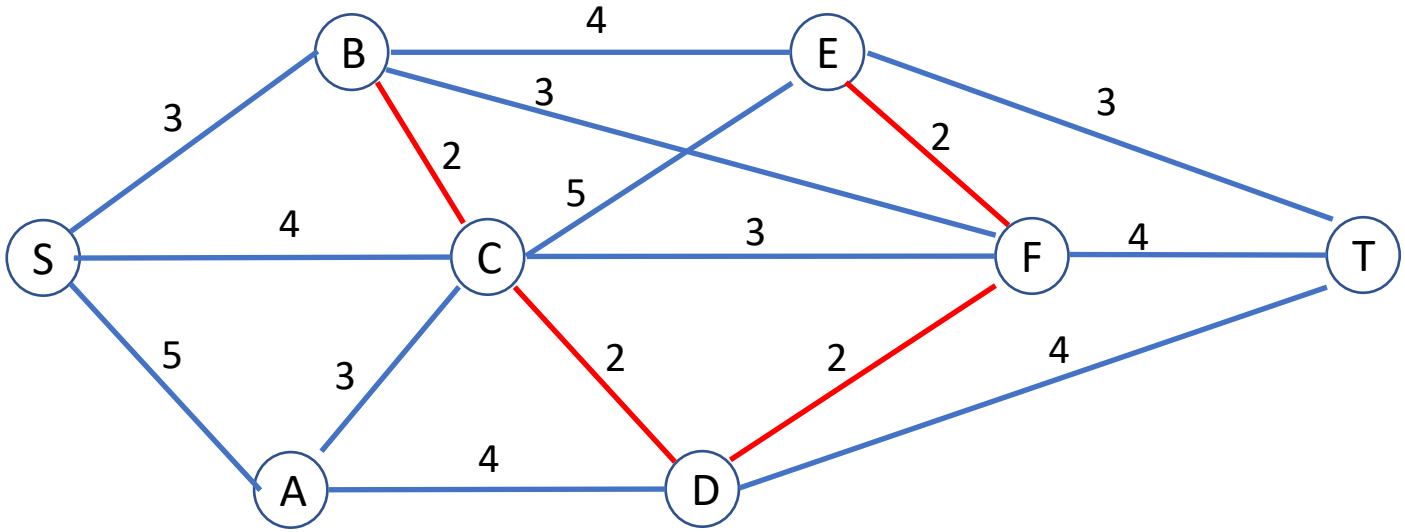


정렬 리스트

(B, C) 2
(C, D) 2
(D, F) 2
(E, F) 2
(S, B) 3
(A, C) 3
(B, F) 3
(C, F) 3
(E, T) 3
(S, C) 4
(A, D) 4
(B, E) 4
(D, T) 4
(F, T) 4
(S, A) 5
(C, E) 5

6.3 최소걸침나무 문제

❖ Kruskal 알고리즘

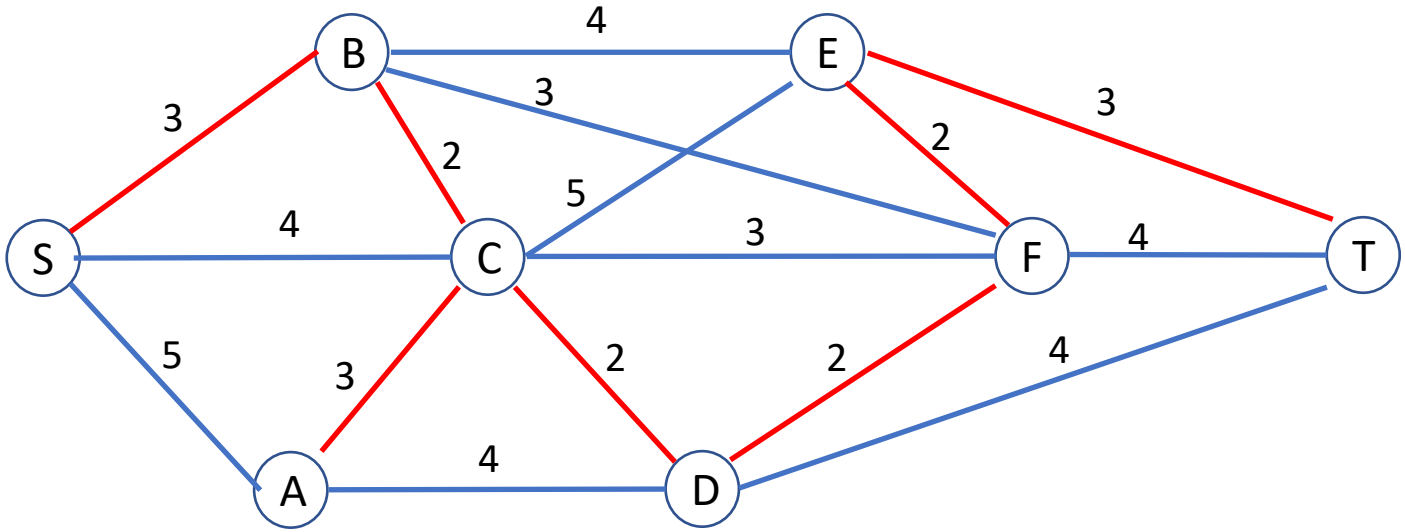


정렬 리스트

(B, C) 2
(C, D) 2
(D, F) 2
(E, F) 2
(S, B) 3
(A, C) 3
(B, F) 3
(C, F) 3
(E, T) 3
(S, C) 4
(A, D) 4
(B, E) 4
(D, T) 4
(F, T) 4
(S, A) 5
(C, E) 5

6.3 최소걸침나무 문제

❖ Kruskal 알고리즘

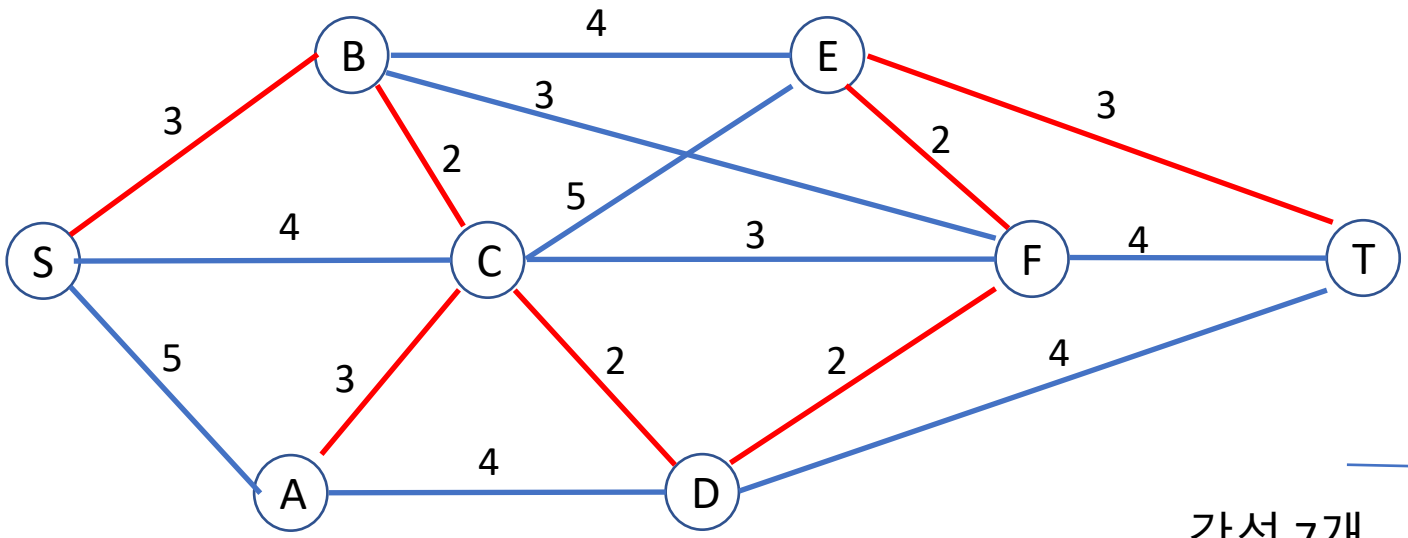


정렬 리스트

(B, C) 2
(C, D) 2
(D, F) 2
(E, F) 2
(S, B) 3
(A, C) 3
~~(B, F) 3~~
~~(C, F) 3~~
(E, T) 3
(S, C) 4
(A, D) 4
(B, E) 4
(D, T) 4
(F, T) 4
(S, A) 5
(C, E) 5

6.3 최소걸침나무 문제

❖ Kruskal 알고리즘



간선 7개

정렬 리스트

(B, C) 2
(C, D) 2
(D, F) 2
(E, F) 2
(S, B) 3
(A, C) 3
~~(B, F) 3~~
~~(C, F) 3~~
(E, T) 3
~~(S, C) 4~~
~~(A, D) 4~~
~~(B, E) 4~~
~~(D, T) 4~~
~~(F, T) 4~~
~~(S, A) 5~~
~~(C, E) 5~~

6.3 최소걸침나무 문제

❖ Kruskal 알고리즘

- 추가하려는 간선이 싸이클을 만드는지의 여부는 집합과 관련된 연산인 union(합집합) 연산과 주어진 원소에 대해 어느 집합에 속해 있는지를 찾는 find 연산을 사용
- 특히 어느 두 집합도 중복된 원소를 갖지 않는 경우, 이러한 집합들을 상호 배타적 집합(Disjoint Set)이라고 한다.
- 상호 배타적 집합들은 1차원 리스트로 표현 가능하며, 모든 집합들의 원소를 0, 1, 2, …, N-1로 놓으면 이를 리스트의 인덱스로 활용할 수 있기 때문이다.

6.3 최소걸침나무 문제

❖ Kruskal 알고리즘

- 간선을 정렬(또는 우선순위큐의 삽입과 삭제)하는데 소요되는 시간인 O(MlogM) = O(MlogN)과 트리에 간선을 추가하려 할 때 find와 union을 수행하는 시간인 O((M+N)log*N)의 합이다.
- 즉, O(MlogN) + O((M+N)log*M) = O(MlogN)이다.
- union 연산은 단순히 하나의 루트가 다른 루트의 자식이 되는 것이므로 O(1) 시간 소요
- 실제로 조금 복잡한 분석 방법을 통해서 union과 find 연산들의 수행 시간인 O((M+N)log*N)이 계산된다

6.3 최소걸침나무 문제

❖ Prim알고리즘

- Prim 알고리즘은 임의의 시작 정점에서 가장 가까운 정점을 추가하여 간선이 하나의 트리를 만들고, 만들어진 트리에 인접한 가장 가까운 정점을 하나씩 추가하여 최소신장트리를 만든다.
- Prim의 알고리즘에서는 초기에 트리 T는 임의의 정점 s만을 가지며, 트리에 속하지 않은 각 정점과 T의 정점(들)에 인접한 간선들 중에서 가장 작은 가중치를 가진 간선의 끝점을 찾기 위해 리스트 D를 사용

❖절차

D를 ∞로 초기화한다. 시작 정점 s의 D[s] = 0

while T의 정점 수 < N:

T에 속하지 않은 각 정점 i에 대해 D[i]가 최소인 정점 min_vertex를 찾아 T에 추가

for T에 속하지 않은 각 정점 w에 대해서:

if 간선 (min_vertex, w)의 가중치 < D[w]:

D[w] = 간선 (min_vertex, w)의 가중치

6.3 최소걸침나무 문제

❖ Prim알고리즘

- Prim 알고리즘은 N번의 반복을 통해 min_vertex를 찾고 min_vertex에 인접하면서 트리에 속하지 않은 정점에 해당하는 D 의 원소 값을 갱신
- min_vertex를 배열 D에서 탐색하는 과정에서 O(N) 시간이 소요되고, min_vertex에 인접한 정점들을 검사하여 D의 해당 원소를 갱신하므로 O(N) 시간이 소요된다.
- 따라서 총 수행 시간은 Nx(O(N) +O(N)) = O(N^2)

6.3 최소걸침나무 문제

❖ Prim알고리즘

- min_vertex 찾기 위해 이진힙을 사용하면 각 간선에 대한 D의 원소를 갱신하며 힙 연산을 수행해야 하므로 총 O(MlogN) 시간이 필요, M은 그래프 간선의 수
- 이진힙은 각 정점에 대응되는 D원소를 저장하므로 힙의 최대 크기는 N
- 또한 가중치가 갱신되어 감소되었을 때의 힙 연산에는 O(logN) 시간이 소요
- 입력 그래프가 희소 그래프라면, 예를 들어, M = O(N)이라면, 수행시간이 O(MlogN) = O(NlogN)이 되어 이진힙을 사용하는 것이 매우 효율적

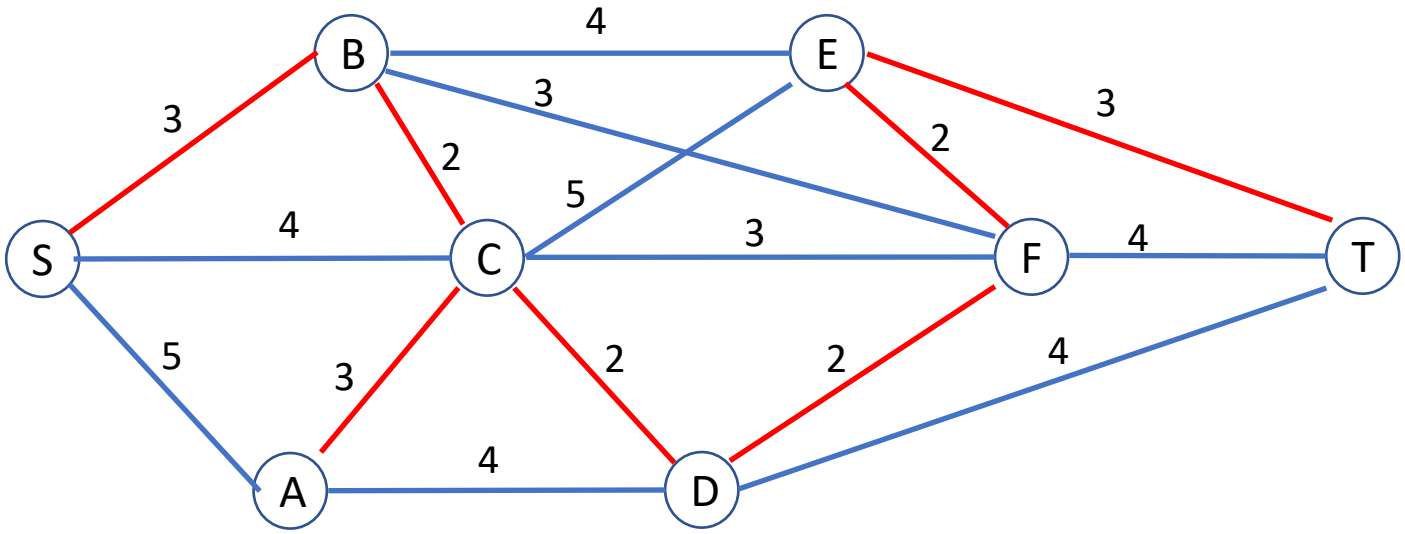
6.3 최소걸침나무 문제

❖ Sollin 알고리즘

- 각 정점을 독립적인 트리로 간주하고, 각 트리에 연결된 간선들 중에서 가장 작은 가중치를 가진 간선을 선택한다. 이때 선택된 간선은 2 개의 트리를 1개의 트리로 만든다.
- 같은 방법으로 한 개의 트리가 남을 때까지 각 트리에서 최소 가중치 간선을 선택하여 연결
- Sollin 알고리즘은 병렬알고리즘(Parallel Algorithm)으로 구현이 쉽다는 장점을 가짐

6.3 최소걸침나무 문제

❖ Sollin 알고리즘



6.3 최소걸침나무 문제

❖ Sollin 알고리즘

- repeat-루프가 예제와 같이 각 쌍의 트리가 서로 연결된 간선을 선택하는 경우 최대 logN번 수행
- 루프 내에서는 각 트리가 자신에 닿아 있는 모든 간선들을 검사하여 최소 가중치를 가진 간선을 선택하므로 O(M) 시간이 소요
- 따라서 알고리즘의 수행 시간은 O(MlogN)

6.4 그리디 알고리즘

❖ 그리디 알고리즘(greedy Algorithm)

- 그리디 : 탐욕스러운, 욕심 많은
- 문제를 해결하는 과정에서 매 순간, 최적이라 생각되는 해답(locally optimal solution)을 찾으며, 이를 토대로 최종 문제의 해답(globally optimal solution)에 도달하는 문제 해결 방식
- 순간마다 하는 선택은 그 순간에 대해 지역적으로는 최적이지만, 그 선택들을 계속 수집하여 최종적(전역적)인 해답을 만들었다고 해서, 그것이 최적이라는 보장은 없다.
- 하지만 탐욕 알고리즘을 적용할 수 있는 문제들은 지역적으로 최적이면서 전역적으로 최적인 문제들

6.4 그리디 알고리즘

❖ 탐욕 알고리즘 적용의 2가지 조건

- 탐욕적 선택 속성(Greedy Choice Property) : 앞의 선택이 이후의 선택에 영향을 주지 않는다.
- 최적 부분 구조(Optimal Substructure) : 문제에 대한 최종 해결 방법은 부분 문제에 대한 최적 문제 해결 방법으로 구성된다.

❖ 이러한 조건이 성립하지 않는 경우에는 탐욕 알고리즘은 최적해를 구하지 못함

- 하지만, 이러한 경우에도 탐욕 알고리즘은 근사 알고리즘으로 사용이 가능할 수 있으며, 대부분의 경우 계산 속도가 빠르기 때문에 실용적으로 사용할 수 있다.

❖ 어떤 특별한 구조(매트로이드)가 있는 문제에 대해서는 탐욕 알고리즘이 언제나 최적해를 찾아낼 수 있다.

❖ 탐욕 알고리즘 문제를 해결하는 방법

- 선택 절차(Selection Procedure): 현재 상태에서의 최적의 해답을 선택
- 적절성 검사(Feasibility Check): 선택된 해가 문제의 조건을 만족하는지 검사
- 해답 검사(Solution Check): 원래의 문제가 해결되었는지 검사하고, 해결되지 않았다면 선택 절차로 돌아가 위의 과정을 반복

6.4 그리디 알고리즘

❖ 동전 거스름돈 문제

- 남은 액수를 초과하지 않는 조건하에 '욕심내어' 가장 큰 액면의 동전을 취하는 것
- 동전 거스름돈 문제의 최소 동전 수를 찾는 그리디 알고리즘
  - ✓ 동전의 액면은 500원, 100원, 50원, 10원, 1원
  - ✓ 500원 동전을 처리하는데 100원, 50원, 10원, 1원 동전을 몇 개씩 거슬러 주어야 할 것인지에 대해서는 전혀 고려하지 않는다
- CoinChange 알고리즘은 항상 최적의 답을 주지 못함(동전 체계에 의존)
  - ✓ 그러나 실제로는 거스름돈에 대한 그리디 알고리즘이 적용되도록 동전이 발행됨

CoinChange 알고리즘의 결과

최소 동전의 거스름돈

6.4 그리디 알고리즘

❖ 0-1 배낭 문제

- ➢ 탐욕적 전략으로 최적해를 구하지 못함
  - ➢ 탐욕 1: 무게와 상관없이 가장 비싼 물건부터 넣어보는 방법
  - ➢ 탐욕 2: 가장 가벼운 물건부터 넣어보는 방법
  - ➢ 탐욕 3:단위 무게당 가격이 가장 높은 물건부터 넣어보는 방법

3kg 60만원

6kg 20만원

5kg
10만원

4kg
40만원

탐욕1: 100만원

탐욕2: 100만원

탐욕3: 100만원

6.4 그리디 알고리즘

❖ 0-1 배낭 문제 (분할 가능, factional knapsack)

➢ 탐욕적 전략으로 최적해를 구하지 못함

➢ 무게당 가격이 비싼 물건부터 순서대로 배낭의 남은 용량을 넘지 않는 최대한으로 채움

3kg 60만원

6kg 20만원

5kg
10만원

4kg
40만원

3kg 60만원 : 20만원/kg

6kg 20만원 : 3.3만원/kg 10만원

5kg 10만원 : 2만원/kg

4kg 40만원 : 10만원/kg

6.4 그리디 알고리즘

❖ 집합커버문제 (Set Cover)

- ➢ 집합 F에서 선택하는 집합들의 수를 최소화하는 문제
  - ✓ n개의 원소를 가진 집합 U가 있고,
  - ✓ U의 부분집합들을 원소로 하는 집합 F가 주어질 때,
  - ✓ F의 원소들인 집합들 중에서 어떤 집합들을 선택하여 합집합하면 U와 같게 되는가?
- ➢ 신도시 소방서 배치 사례
  - ✓ 10개의 마을이 신도시에 만들어질 계획
  - ✓ 다음 조건이 만족되도록 소방서 위치를 선정
    - 소방서는 마을에 위치
    - 출동 거리는 10분 이내
  - ✓ 어느 마을에 소방서를 신설해야 소방서의 수가 최소가 되는가?

6.4 그리디 알고리즘

❖ 집합커버문제 (Set Cover)

- ➢ 집합 F에서 선택하는 집합들의 수를 최소화하는 문제
  - ✓ F에 있는 집합들의 모든 조합을 1개씩 합집합하여 U가 되는지 확인하고,
  - ✓ U가 되는 조합의 집합 수가 최소인 것을 찾는다.
  - ✓ F={S1, S2, S3}일 경우 모든 조합
  - ✓ S1, S2, S3, S1∪S2, S1∪S3, S2∪S3, S1∪S2∪S3
  - ✓ 집합이 1개인 경우 3개 = 3C1
  - ✓ 집합이 2개인 경우 3개 = 3C2
  - ✓ 집합이 3개인 경우 1개 = 3C3
  - ✓ 총합은 3+3+1= 7 = 2^3-1 개 (n이 커지면 최적해를 찾는 것은 실질적으로 불가능, 근사해 접근)

6.4 그리디 알고리즘

❖ 집합커버문제 (Set Cover)

➢ 신도시 소방서 배치 사례

- ✓ 신도시 계획 문제를 집합 커버 문제로 변환
- ✓ U = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} // 신도시의 마을 10개
- ✓ F = {S1, S2, S3, S4, S5, S6, S7, S8, S9, S10}

  // Si는 마을 i에 학교를 배치했을 때 커버되는 마을의 집합

- ✓ S1={1, 2, 3}, S2={1, 2, 3, 4}, S3={1, 2, 3, 4, 8}, S4={2, 3, 4, 5, 7, 8} , S5={4, 5, 6, 7}
- ✓ S6={5, 6, 7, 9, 10} , S7={4, 5, 6, 7}, S8={1, 2, 4, 8}, S9 ={6,7, 9}, S10={9, 10}

- ✓ Si 집합들 중에서 어떤 집합들을 선택해야 그들의 합집합이 U와 같은가?
- ✓ S3 U S6

6.4 그리디 알고리즘

❖ 집합커버문제 (Set Cover)

➢ 알고리즘

- ✓ 입력: U, F = {Si}, i=1,⋯,n
- ✓ 출력: 집합 커버 C
- ✓ 1. C = Ø
- ✓ 2. while U ≠ Ø
- ✓ 3. U의 원소를 가장 많이 가진 집합 Si를 F에서 선택
- ✓ 4. U = U - Si
- ✓ 5. Si를 F에서 제거하고, Si를 C에 추가
- ✓ 6. return C

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