215061201
Discrete Mathematics

12 week

Advanced Graph Theory

Lecture Contents

- Shortest-Path Problems
- Graph Coloring

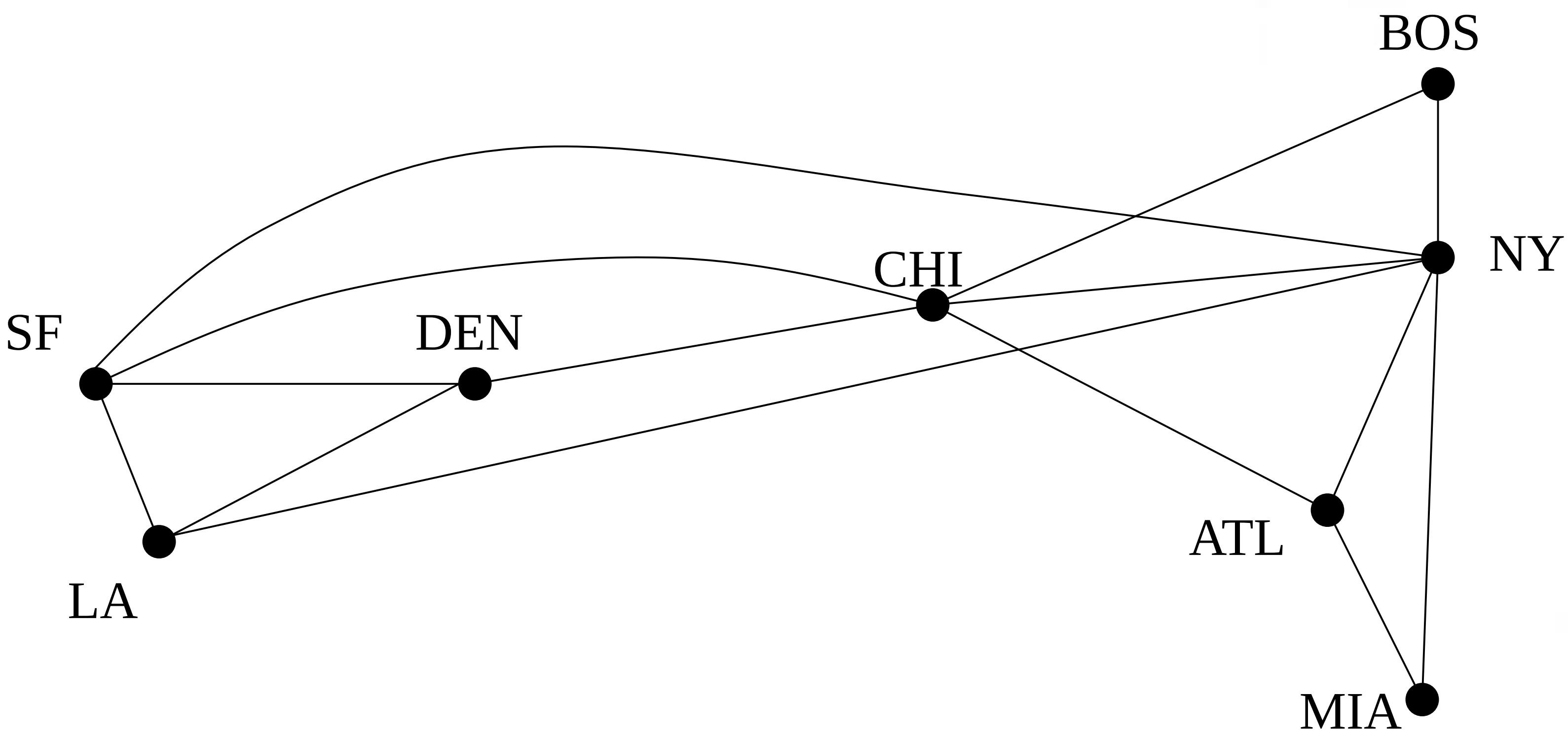
Shortest-Path Problems

SOONGSIL UNIVERSITY

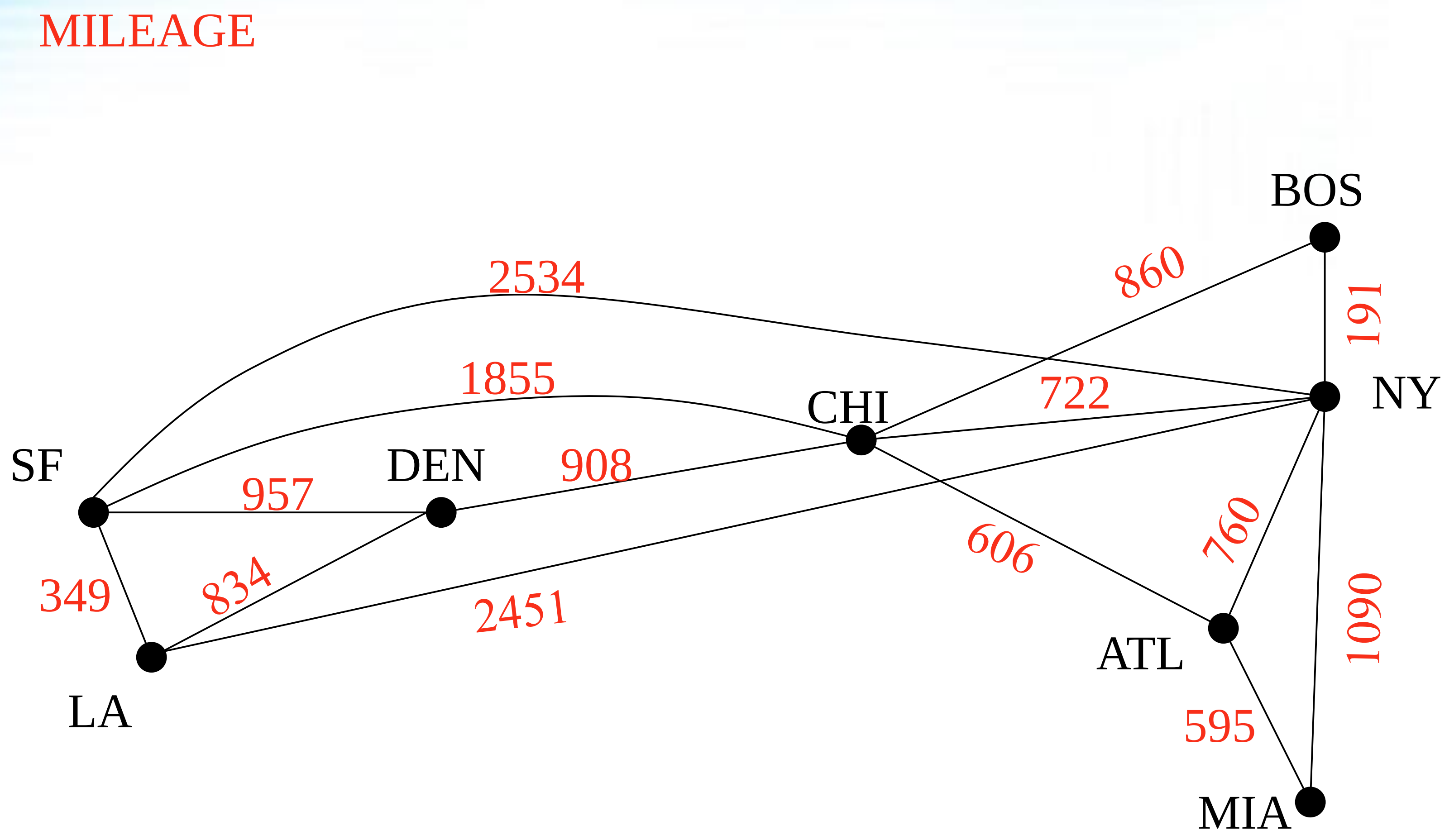
Textbook Chapter 10.6

Weighted Graphs

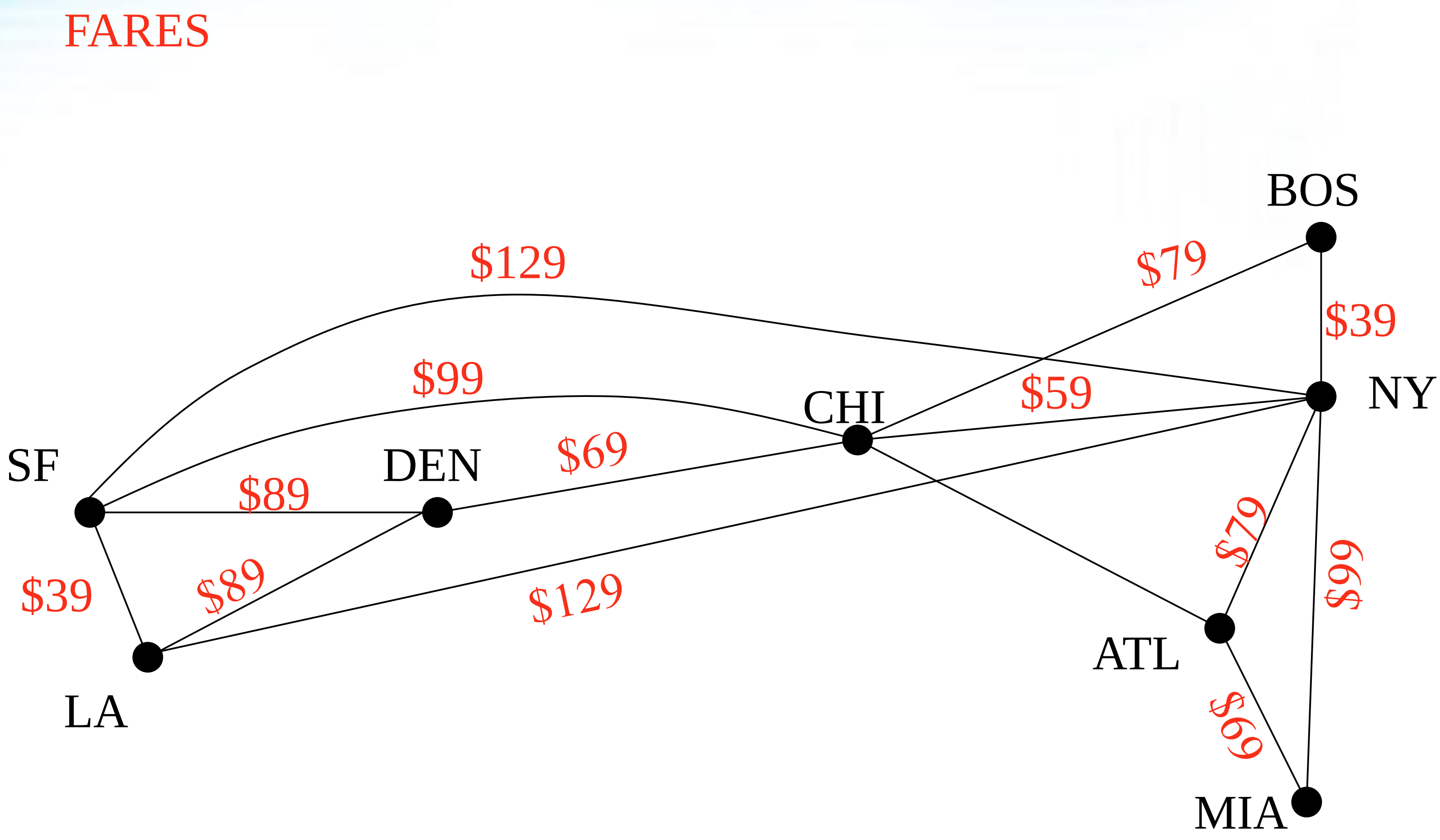
❖ 각 에지에 값(숫자)가 할당된 그래프를 가중치 그래프(*weighted graphs*) 라 함.



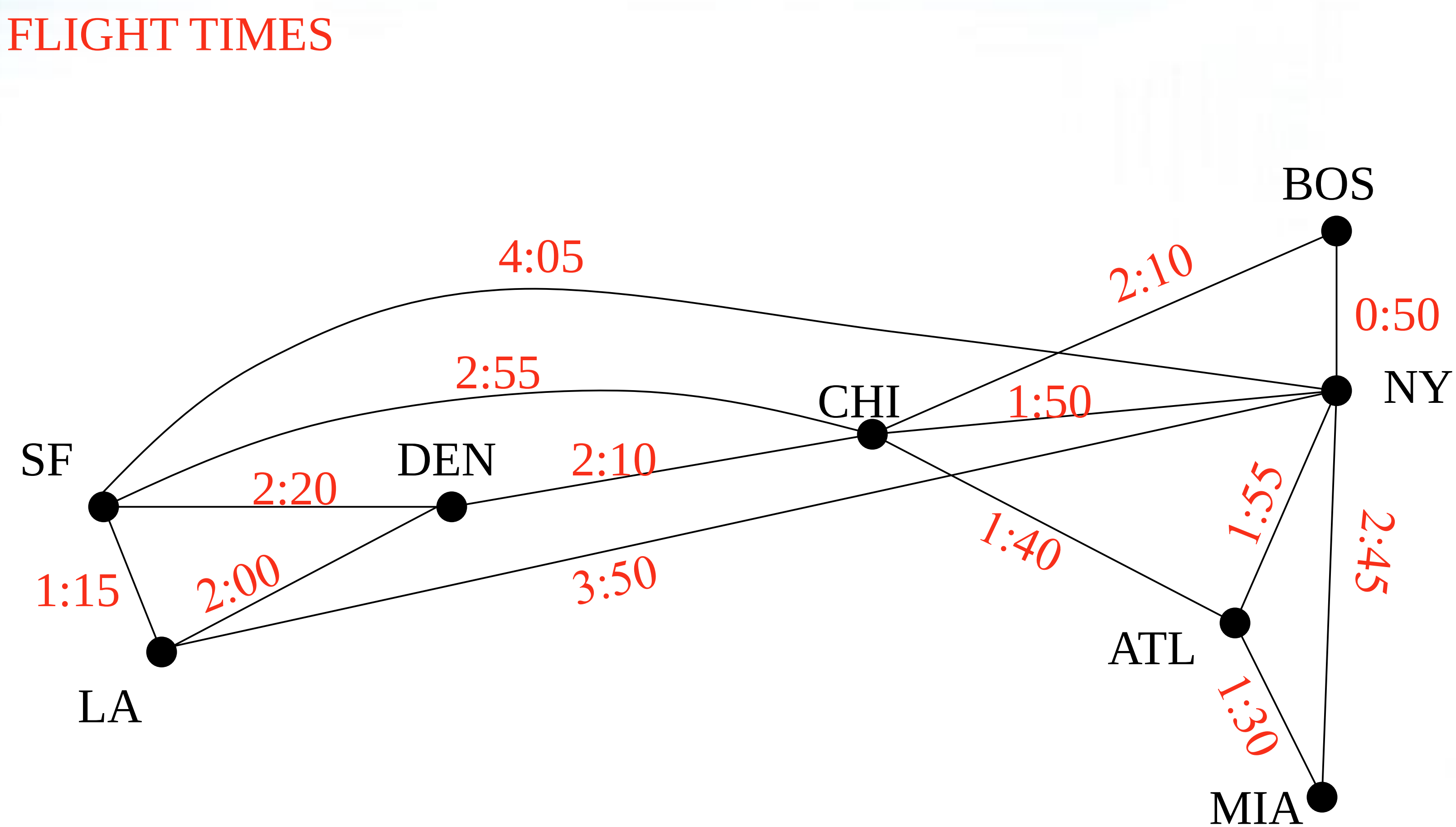
Weighted Graphs



Weighted Graphs



Weighted Graphs



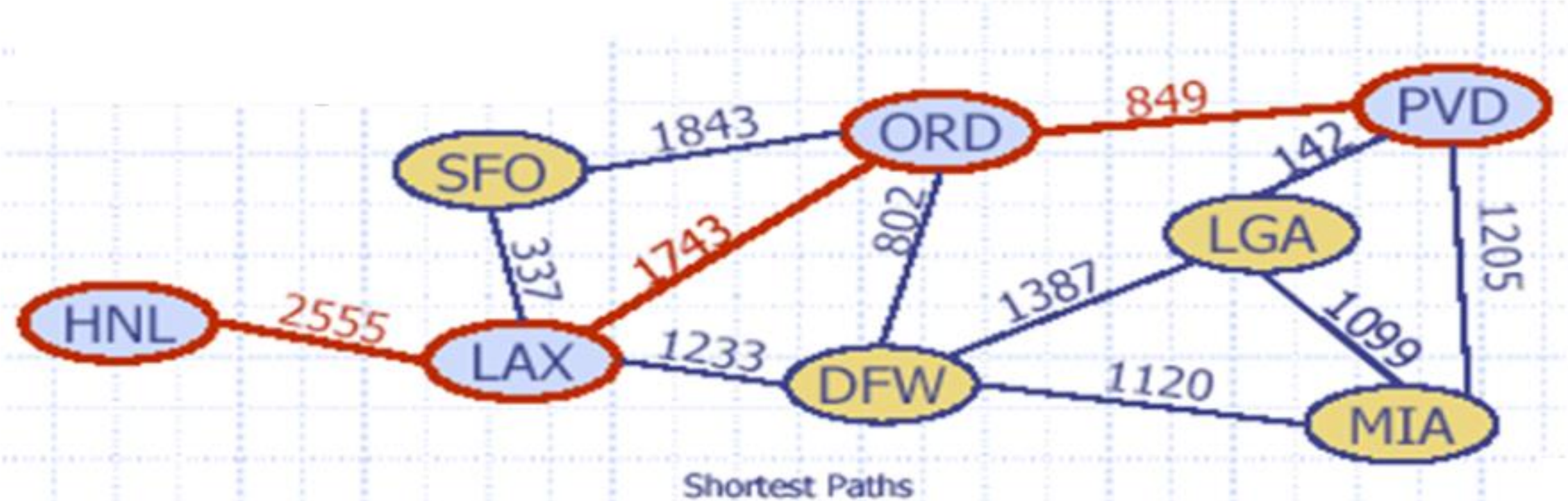
Weighted Graphs

- 가중 그래프는 각 에지 *(u, v)* 에 가중치 *w(u, v)*가 있는 그래프이다. 각 가중치는 실수이다.
- 가중치는 거리, 비용, 시간, 용량 등을 나타낼 수 있다.
- 가중 그래프의 경로 길이는 에지 가중치의 합이다.
- Dijkstra의 알고리즘은 그래프의 임의의 두 노드 사이의 최단 경로를 찾아준다.

Shortest-Path Problem

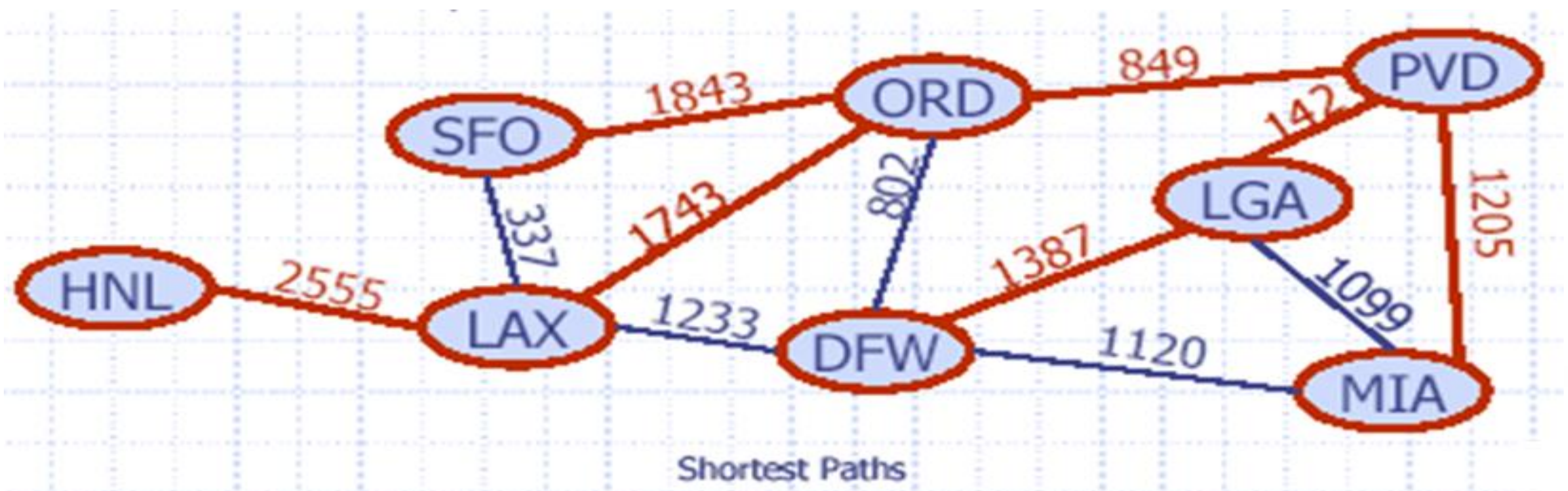
❖ 주어진 가중치 그래프의 2개의 노드사이의 경로의 가중치 합이 최소인 경로 (최단 경로) 찾기

❖ 예; 프로비던스와 호놀루루 사이의 최단 경로



Shortest Path Properties

- Property 1:
  - 최단경로의 서브경로도 최단경로이다.
  - 즉, 최단 경로의 시작노드에서 중간 노드까지의 경로 자체도 시작노드에서 그 중간노드까지의 경로중 최단 경로이다.
- Property 2:
  - 시작노드에서 그래프의 모든 다른 노드까지의 최단경로들의 트리가 있다.
- 예: 프로비던스로부터 최단 경로들의 트리



Dijkstra's Algorithm

- Dijkstra's algorithm 은 시작노드에서 그래프의 모든 노드까지의 최단경로 거리를 계산함.
- Dijkstra's algorithm 의 기본 가정
  - 무방향 연결 단순 그래프 (undirected connected simple graph)
  - 모든 에지의 가중치(weight)는 positive value
- 알고리즘은 시작 노드 (a) 를 시작으로 반복 단계마다 새로운 노드를 추가하면서 증가하는 노드세트(S_k)를 구성해가면서 수행되어 간다. 이 노드세트는 최종적으로는 그래프의 모든 노드를 포함하게 된다.
  - 노드세트 S_k ($S_0 = \varnothing$) 은 S_{k-1} 에 없는 노드 v 를 추가함으로써 업데이트한다.
  - 알고리즘 반복의 각 단계 (k) 때에, 노드세트 S_{k-1} *에 포함되어 있는 않은* 각 노드 v 에 대해 시작노드 (a)로부터 노드 v 까지의 최단경로 길이를 나타내는 레이블 $L_k(v)$ 를 다음과 같이 계산한다. (초기값 $L_0(a) = 0$, $L_0(v)=\infty$). 이때, 시작노드 a 로부터 노드 v 까지의 경로를 구성하는 노드들은 모두 노드세트 S_k 에 속한 노드들이어야 한다

$$L_k(v) = \min\{L_{k-1}(v), L_{k-1}(u) + w(u, v)\}$$

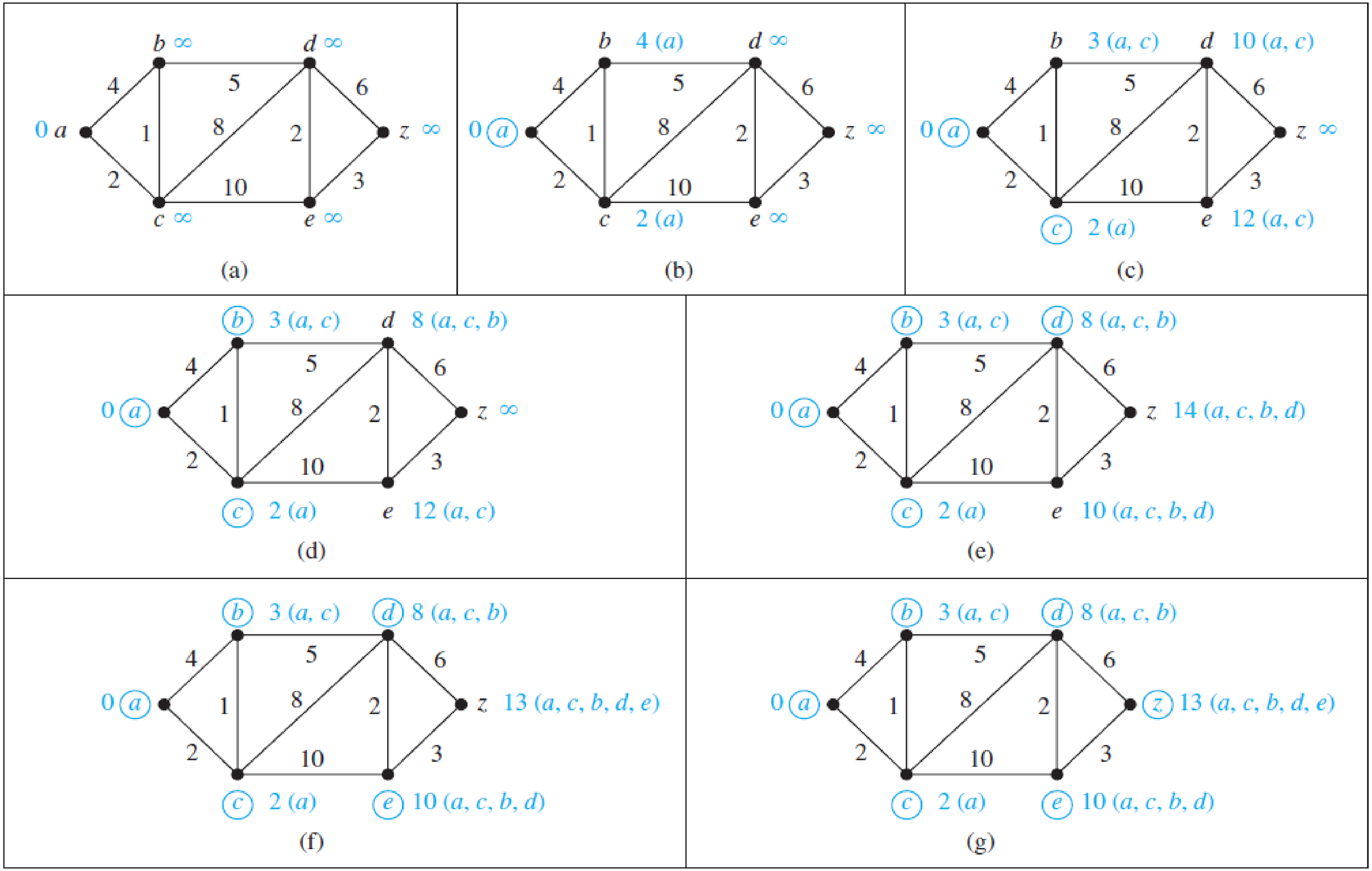
(여기서, 노드 u 는 S_{k-1} 에 속한 노드들 중, 노드 v 의 인접 노드이며, $w(u,v)$는 노드 u 에서 노드 v 까지의 에지의 가중치임.)

  - 상기 계산된 *$L_k(v)$ 들 중 가장 작은 값을 갖는* 노드 v 를 S_{k-1} 에 추가하여 S_k 로 업데이트하고 상기 과정을 S_k 가 모든 그래프 노드들을 포함할 때까지 계속 반복한다.

ALGORITHM 1 Dijkstra's Algorithm.

procedure *Dijkstra*(G: weighted connected simple graph, with
all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$
where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
    $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all
other labels are ∞, and S is the empty set}
while $z \notin S$
    $u :=$ a vertex not in S with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
        {this adds a vertex to S with minimal label and updates the
        labels of vertices not in S}
return $L(z)$ {$L(z) =$ length of a shortest path from a to z}

Ex. of Dijkstra's Algorithm

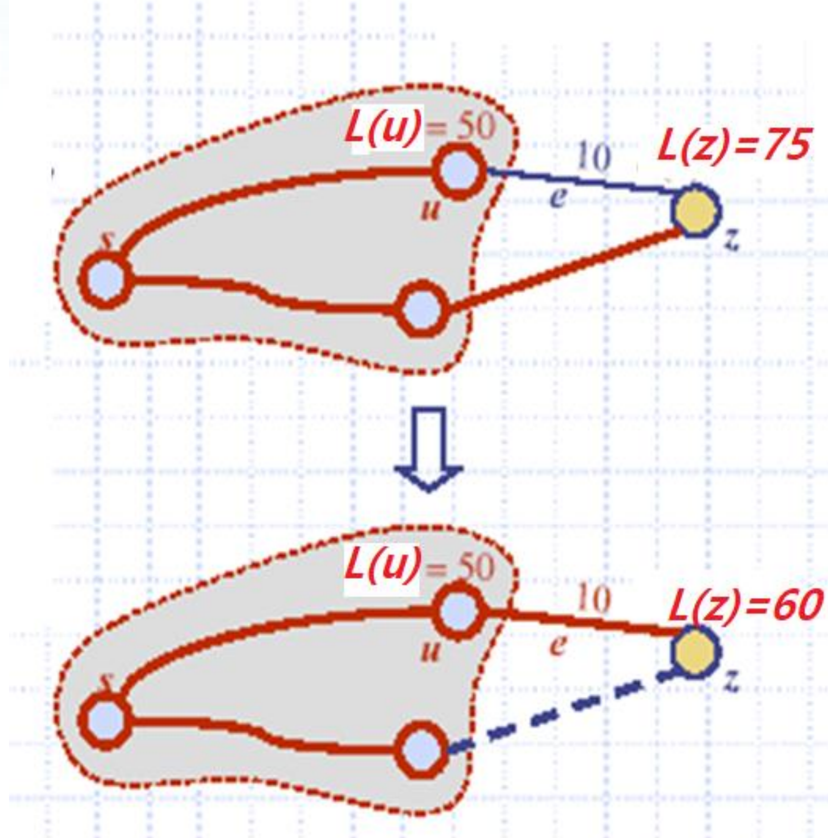


Edge Relaxation

❖ 에지 *e=(u, z)* . 여기서, *u* 는 최근에 노드세트에 추가된 노드, *z* 는 아직 노드세트에 없는 노드 .

❖ 에지 *e 의 이완(relaxation)*은 거리 *L(z)* 를 다음과 같이 업데이트함.

L(z)=min{L(z), L(u)+weight(e) }

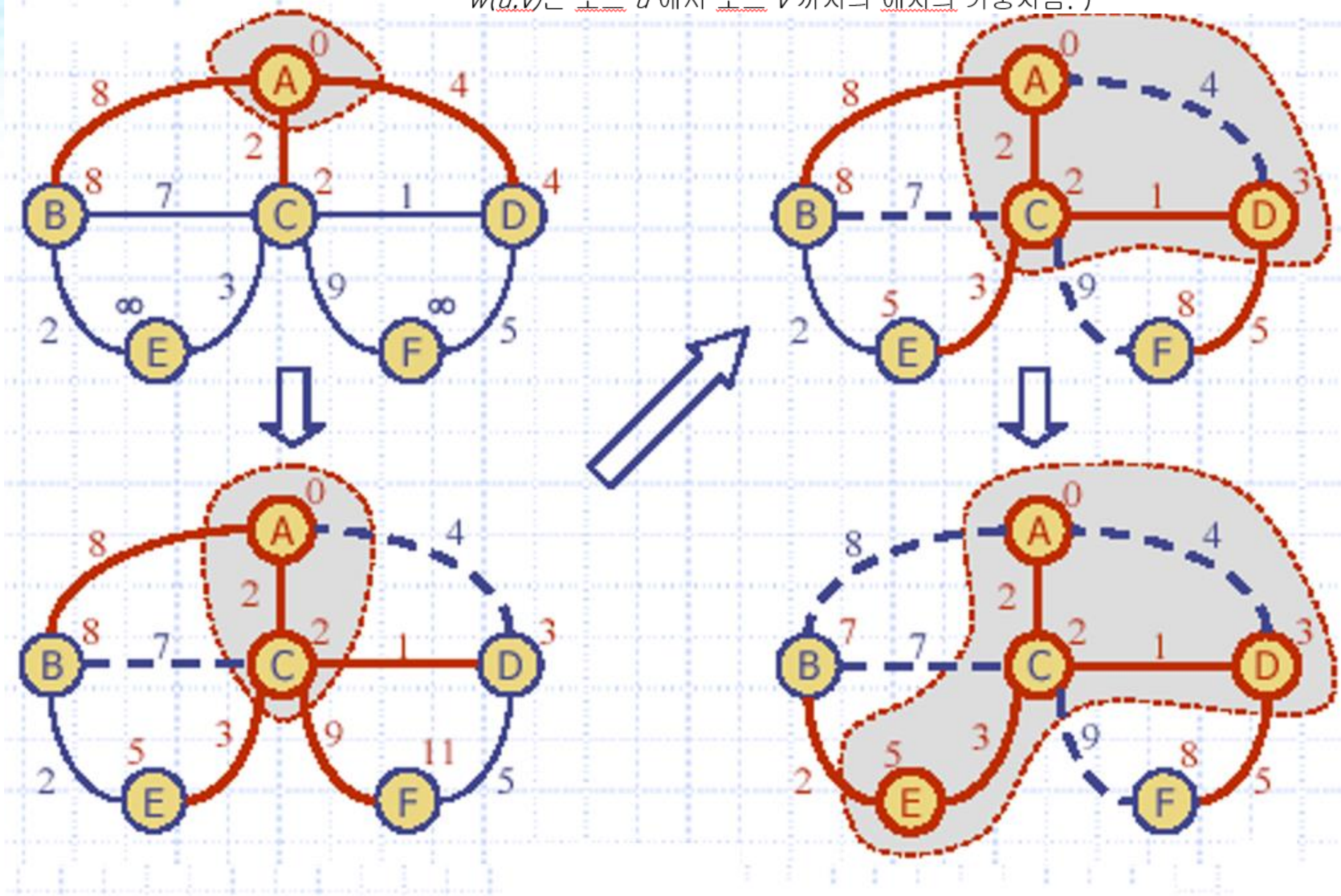


Example

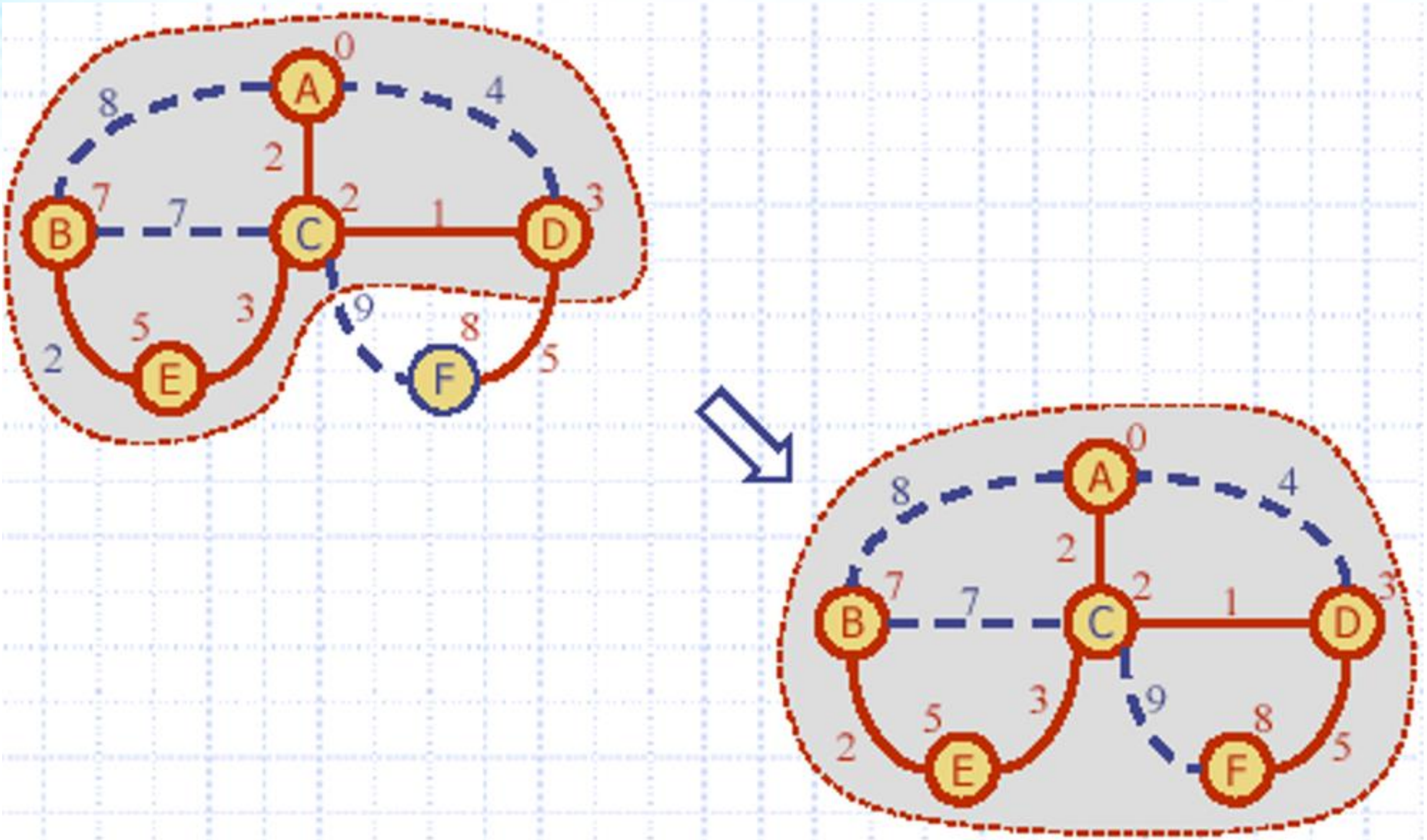
이때, 노드세트 S_{k-1} 에 속하지 않은 노드 v 에 대해, 레이블 $L_k(v)$ 를 다음과 같이 계산한다

$$L_k(v) = \min\{L_{k-1}(v),\, L_{k-1}(u) + w(u, v)\}$$

(여기서, 노드 u 는 S_{k-1} 에 속한 노드들 중, 노드 v의 인접 노드이며, $w(u,v)$는 노드 u에서 노드 v까지의 에지의 가중치임.)

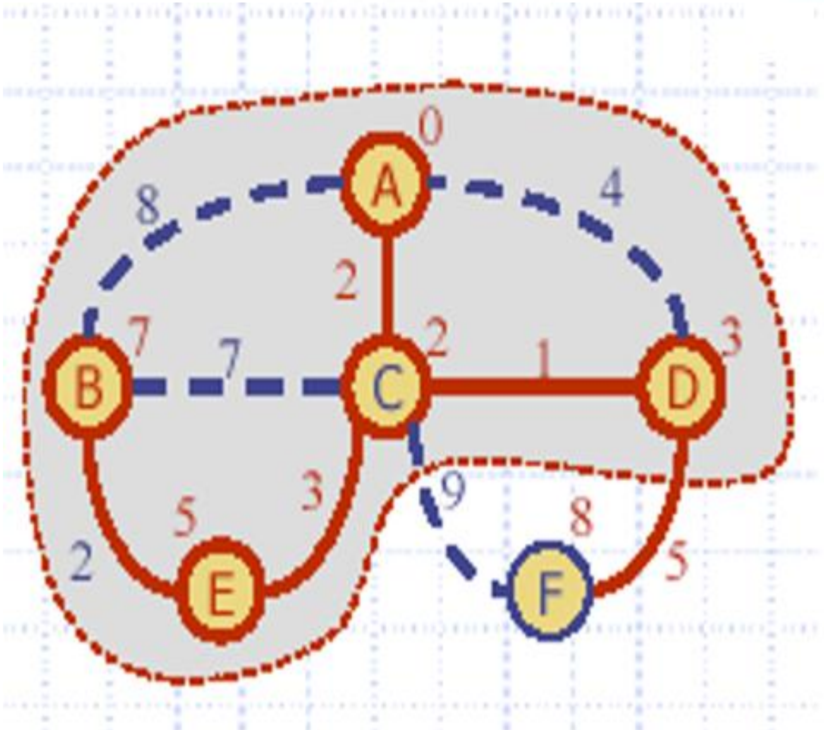


Example (cont.)



Why Dijkstra's algorithm Works?

- Dijkstra의 알고리즘은 탐욕(greedy) 방법을 기반으로 한다. 거리를 늘림으로써 노드를 추가한다.
  - 최단 거리를 모두 찾지 못했다고 가정한다. F는 첫 번째 잘못된 노드라 하자.
  - 진짜 최단 경로에서 있는 이전 노드 D 를 고려했을 때, 여기까지의 거리는 옳음.
  - 하지만 이 때, 에지 (D, F)는 이완됨
  - 따라서 *L(F)> = L(D)* 이면, F의 거리는 틀릴 수 없다. 즉, 잘못된 노드가 없다.



정리

- Dijkstra의 알고리즘은 연결된 단순 무방향 가중치 그래프 $G=(V,E)$ 에서 두 노드 사이의 최단 경로 길이를 찾는다.

- 다익스트라 알고리즘의 시간 복잡도는 $O(|V|^2)$.

- 만일 $\{v \in V \backslash S_i : L(v) < \infty\}$ (*여기서*, S_i 는 반복 i 이후에의 집합 S 임) 정보를 보관하기 위해 힙이 사용되는 경우에, 이 시간 복잡도는 $O(|E|\log|V|)$ *로 축소될 수있음.*

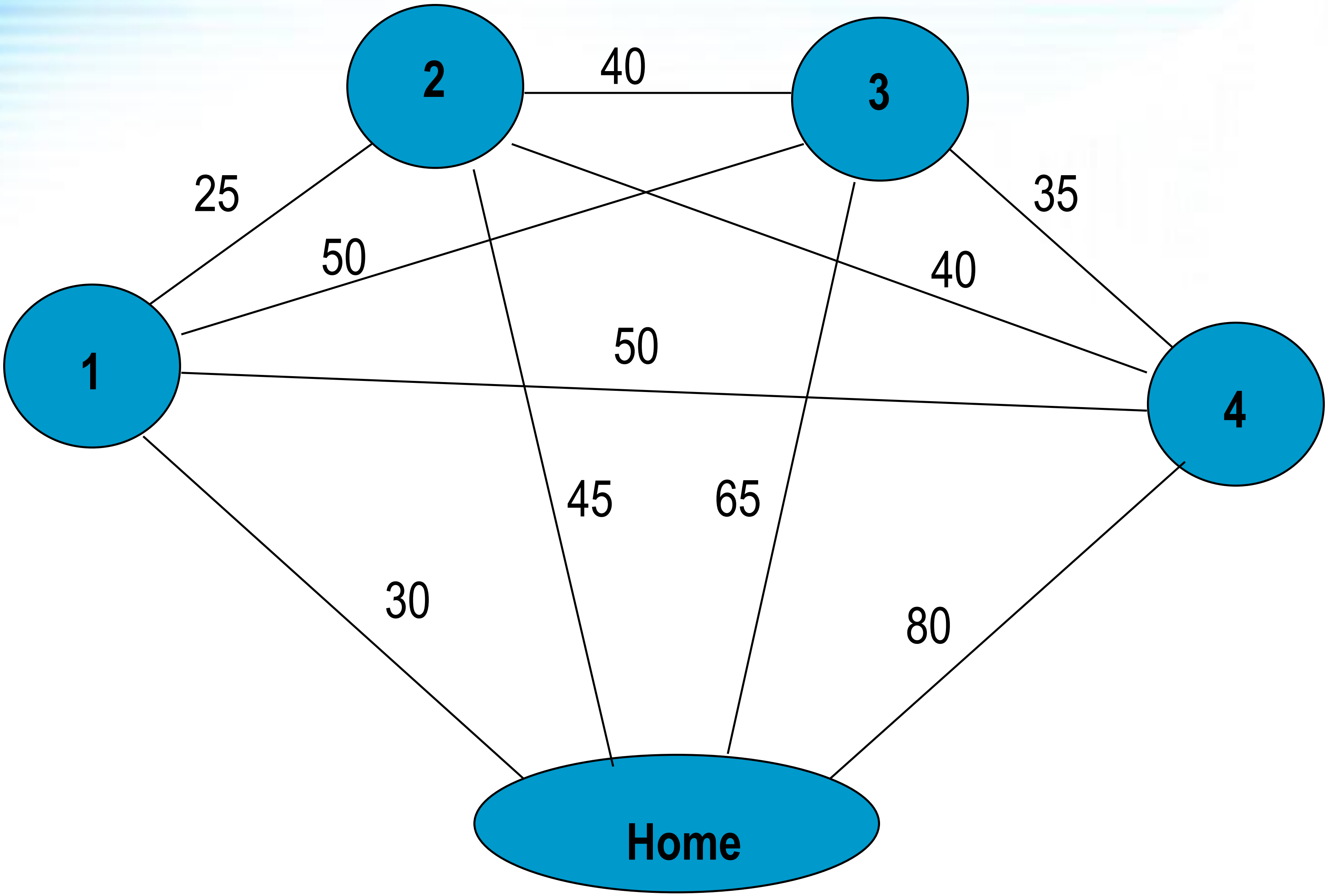
The Traveling Salesman Problem

- 여행세일즈맨 문제는 컴퓨터 과학에서 유명한 고전적인 문제 중 하나이다.
- 여행세일즈맨이 여러 도시를 방문한 다음, 그의 출발점으로 돌아간다. 물론 그는 시간 및 에너지를 절약하고자 하므로, 여행의 최단 사이클을 결정하려고 한다.
- 우리는 도시와 그들 사이의 거리를 가중치 무방향 그래프로 표현할 수있다. .
- 가장 짧은 사이클 (정확히 하나의 노드(시티)을 방문하는 여행 사이클가운데 가중치 합이 가장 적은 길이를 갖는 사이클)을 발견하는 하는 것이 문제이다.
- 가장 짧은 사이클을 찾는 것은 Dijkstra의 최단 경로 찾는 것과는 다르다. 이 문제는 더 어렵고, 다항식 시간 알고리즘이 존재하지 않는 것으로 잘 알려져 있다!

The Traveling Salesman Problem

- Importance:
  - 여행 세일즈맨 문제로 다양한 스케줄링 응용 문제를 해결할 수 있다.
  - 예제:
    - 드릴 프레스에서 드릴 위치 조정.
    - 스쿨 버스 라우팅.
  - 이 문제는 NP-hard 문제로 알려진 어려운 문제 클래스를 대표하기 때문에 이론적으로 중요하다.

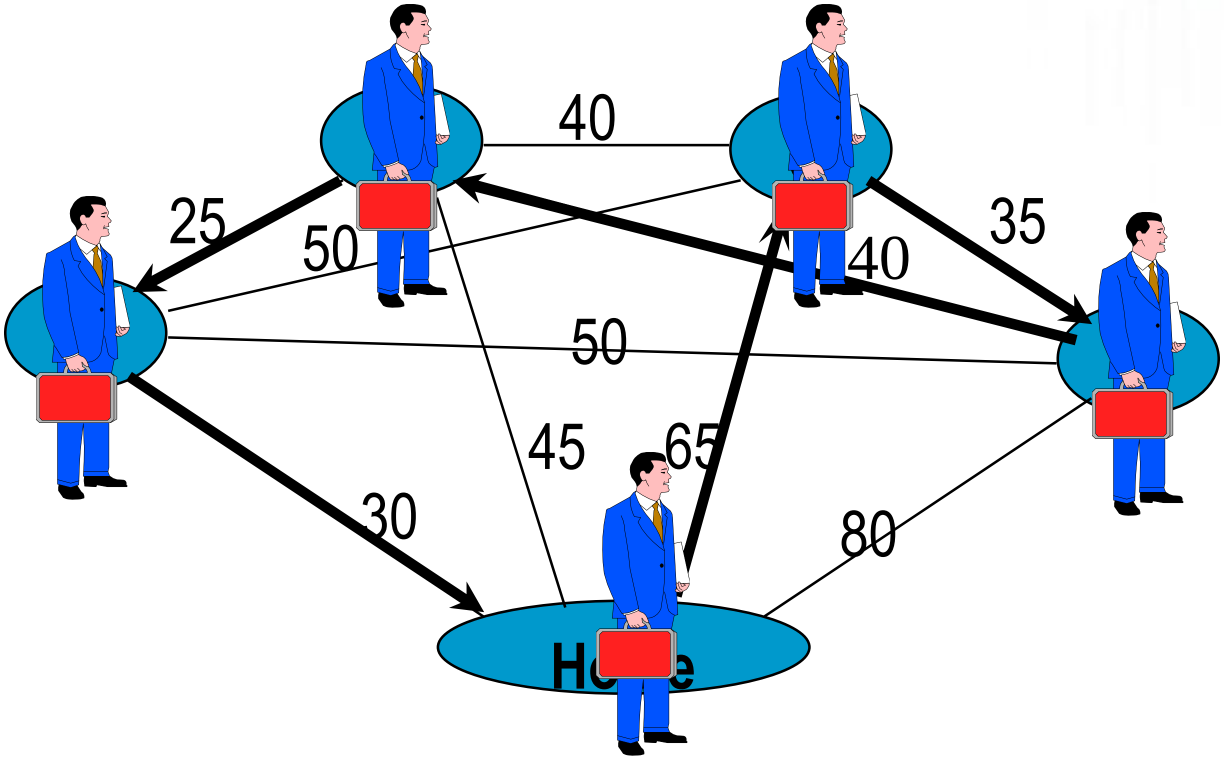
TSP 예제



Traveling Salesman

- **Solution approaches**
  - Brutal force; 모든 가능한 사이클을 모두 고려하여 비교.
    - 노드가 m 개 인 경우에, *(m-1)!* 사이클 존재.
    - 노드 개수가 작은 경우가 아니면, 이러한 방법으로 문제 해결팩 구하는 것은 불가능함.

TSP – optimal solution



The Traveling Salesman Problem

- 불행히도 여행세일즈맨을 해결하는 알고리즘은 없다.
- 다항식 최악의 경우 시간 복잡도 를 갖는 해결책이 없으므로, 이것은 많은 수의 노드를 갖는 그래프에서 여행세일즈맨 문제 풀이는 비현실적이다.
- 이 경우 효율적인 *근사 알고리즘*을 사용할 수 있다. 이 경우의 알고리즘에서는 길이가 약간 더 길 수있는 경로를 찾을 수있다.

Graph Coloring

SOONGSIL UNIVERSITY

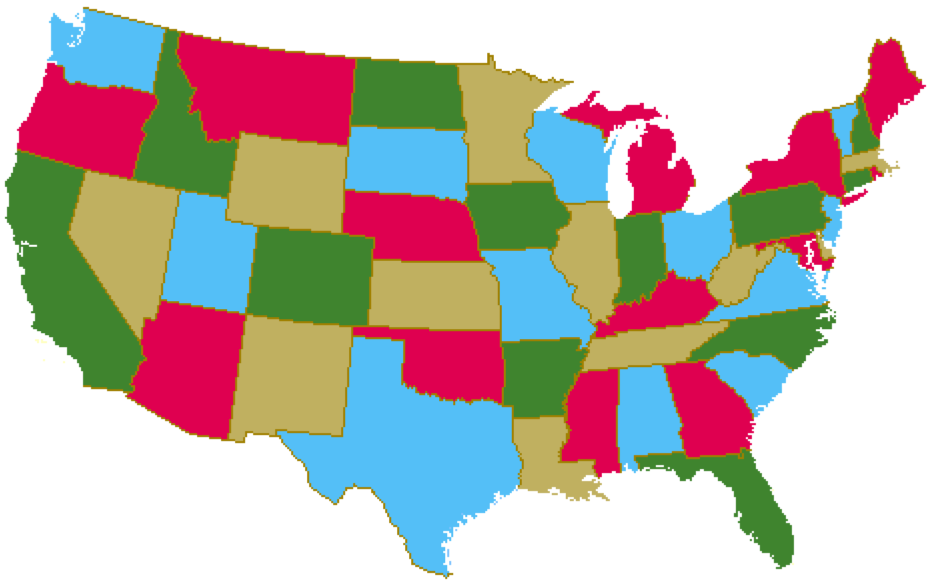
Textbook Chapter 10.8

지도 색칠 문제

- 지도에 색칠할 때, 인접한 2개의 지역은 서로 다른 색을 칠하여야 한다.
- 그런데, 가능한 한 가장 적은 색 종류를 사용하는 게 바람직하다.

4-Color Map Theorem

- 인접 영역은 서로 다른 색깔을 칠한 다는 조건을 만족하면서, 4가지 색상 이하를 사용하여 2차원 지도를 색칠할 수있다.

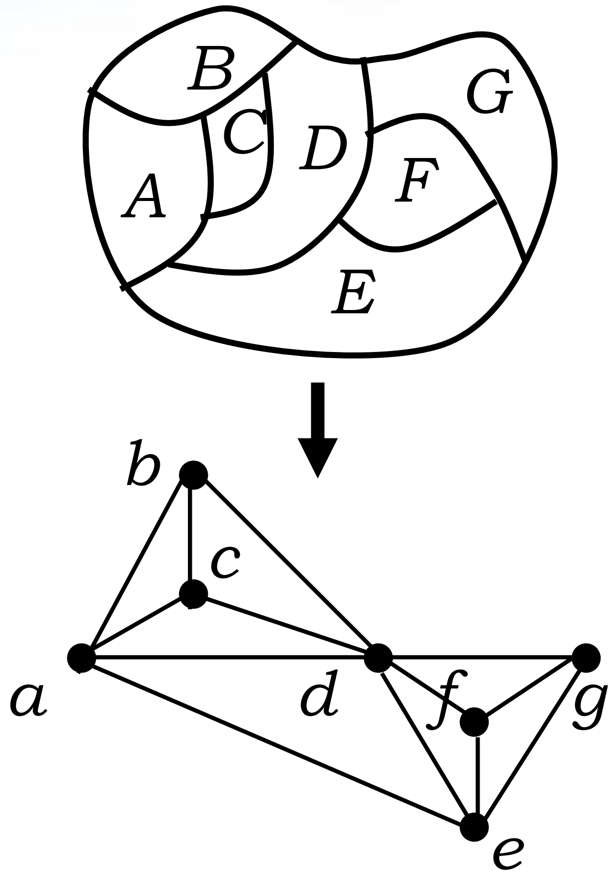


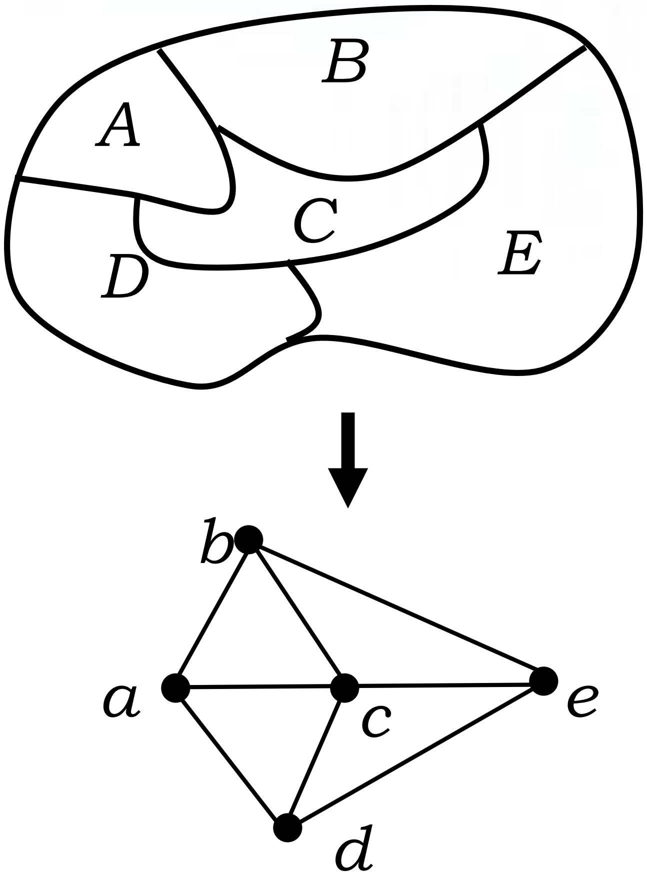
- 4개 컬러면 미국 지도를 색칠하는 데 충분하다.

Dual Graph

- 각 2차원 지도는 그래프로 표현될 수있다.
  - 각 영역은 노도로 대응.
  - 인접 영역은 영역을 나타내는 노드 간의 에지로 표현.
  - 한점에서 터치되는 2개 영역은 인접 영역으로 간주되지 않음.
- 결과되는 그래프는 지도의 ‘*dual graph*’ 라 함.

Dual Graph Examples





Graph Coloring

- 단순 그래프 컬러링은 인접 노드에 동일한 색상이 할당되지 않도록 그래프의 각 노드에 색상을 할당하는 것을 말한다.

- 그래프의 *chromatic number* 는 그래프의 채색에 필요한 최소 색상 수이다.

- 이분 그래프의 색수는 무엇입니까? 2

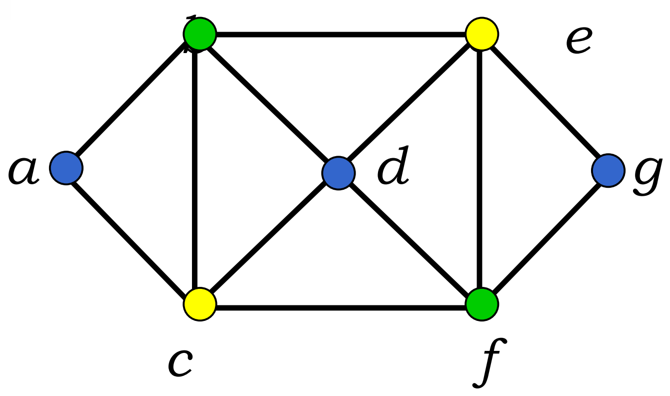
The Four Color Theorem

- **4 컬러 정리 ; 평면 그래프(planar graph)의 chromatic number 는 4 보다 크지 않다.**

- 원래 1850 년대에 추측으로 제기되었다. 1976 년 두 명의 미국 수학자 케네스 애플과 볼프강 하켄에 의해 증명되었다. 이것은 컴퓨터의 도움으로 입증된 첫 번째 수학적 정리이다. 그들은 정리가 거짓이면 약 2000 가지 유형 중 하나의 반례가 존재하게 되는 데, 컴퓨터를 사용하여 이러한 반례가 존재하지 않음을 증명하였다.

예제

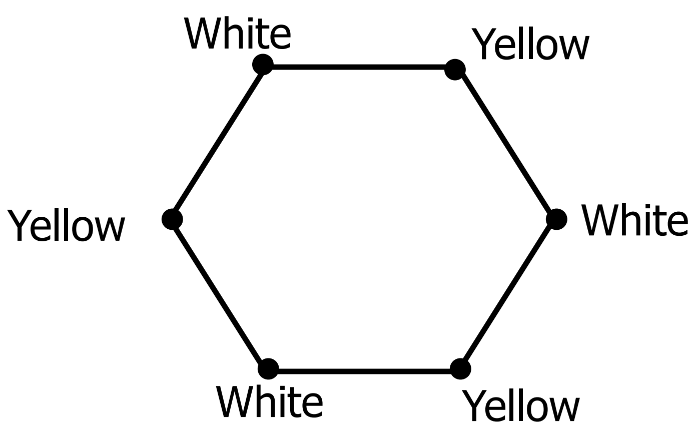
❖ 아래 그래프의 chromatic number 는 ?

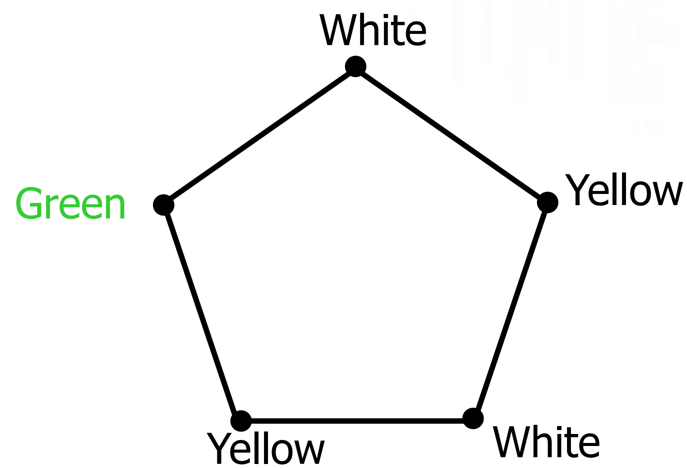


❖ a, b, c는 서로 다른 색상을 담당해야하므로 chromatic number 는 3 이상 이어야 한다. 그러니 먼저 3 색을 시도해 보자. 3 가지 색상으로 채색이 가능하므로 이 그래프의 chromatic number 는 3 이다.

예제

- 다음 그래프들의 **chromatic number ?**





Chromatic number: 2

Chromatic number: 3

Time Complexity

- ❖ 그래프의 chromatic number 를 찾는 것으로 알려진 최고의 알고리즘은 지수 최악의 경우 시간 복잡도 (그래프의 정점 수)을 가짐.
- ❖ 그래프의 chromatic number 에 대한 근사치를 찾는 문제조차도 어렵다.
- ❖ 이것은 기말 시험 시간을 예약하는 것이 왜 그렇게 어려운지 설명한다. 즉, 학생이 동시에 두 개의 시험을 보지 않도록 대학에서 기말 시험을 어떻게 예약 할 수 있을 까 ?