Graph에 기반한 최적 Tree 산출

WEEK 12 / SPRING 2023

STACK & QUEUE

TREE & GRAPH

DATA STRUCTURE

고려대학교 2023학년도 1학기 <자료구조> 03분반 유용재 | 정보대학 컴퓨터학과 | yooy@korea.ac.kr

WEEK 12 / SPRING 2023

Graph에 기반한 최적 Tree 산출

NOW

THEME TODAY

01 최소 신장 Tree 문제

02 Prim Algorithm의 실제

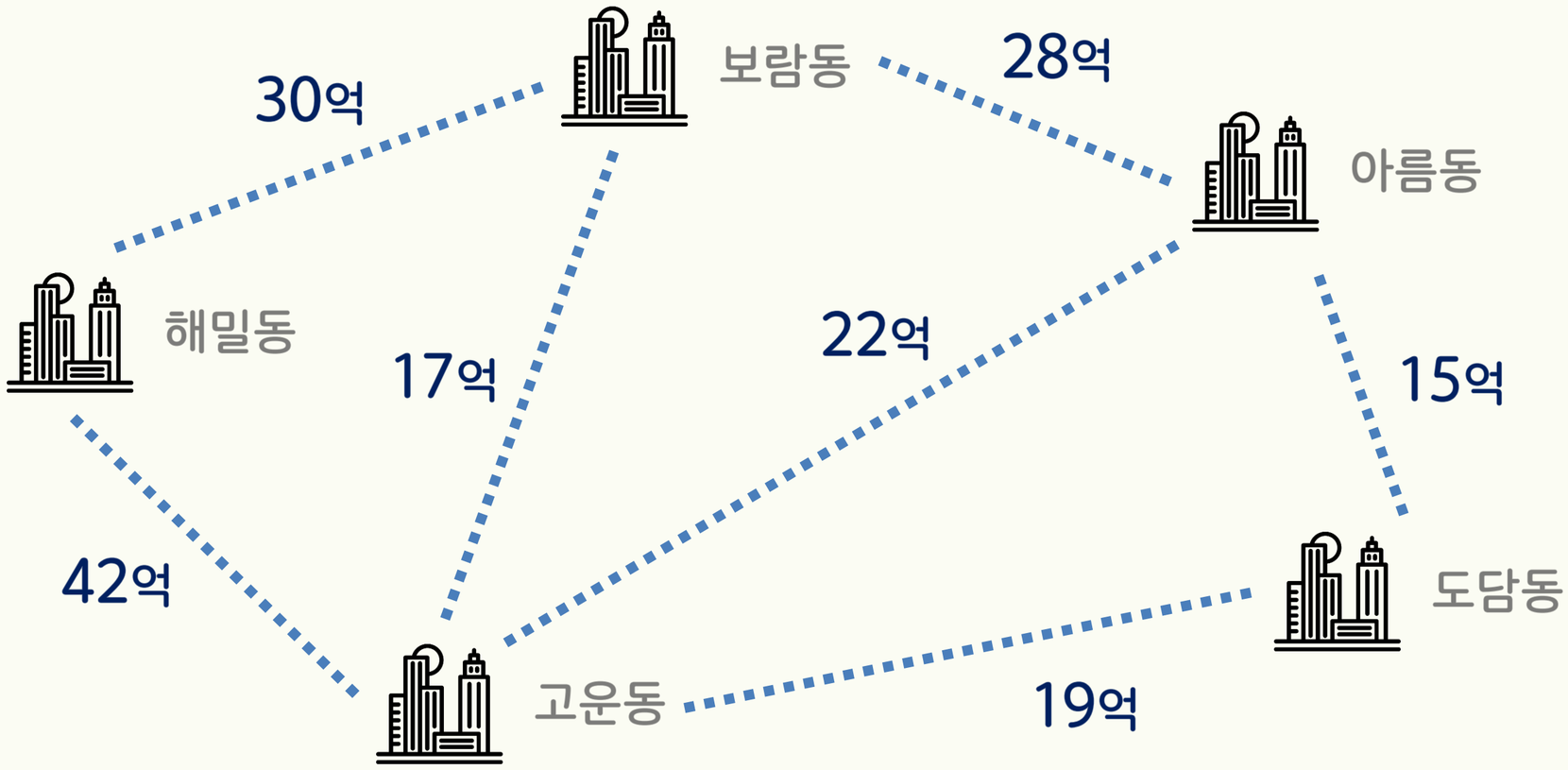
03 Prim Algorithm의 개선

04 Kruskal Algorithm의 이해

05 Kruskal Algorithm의 구현

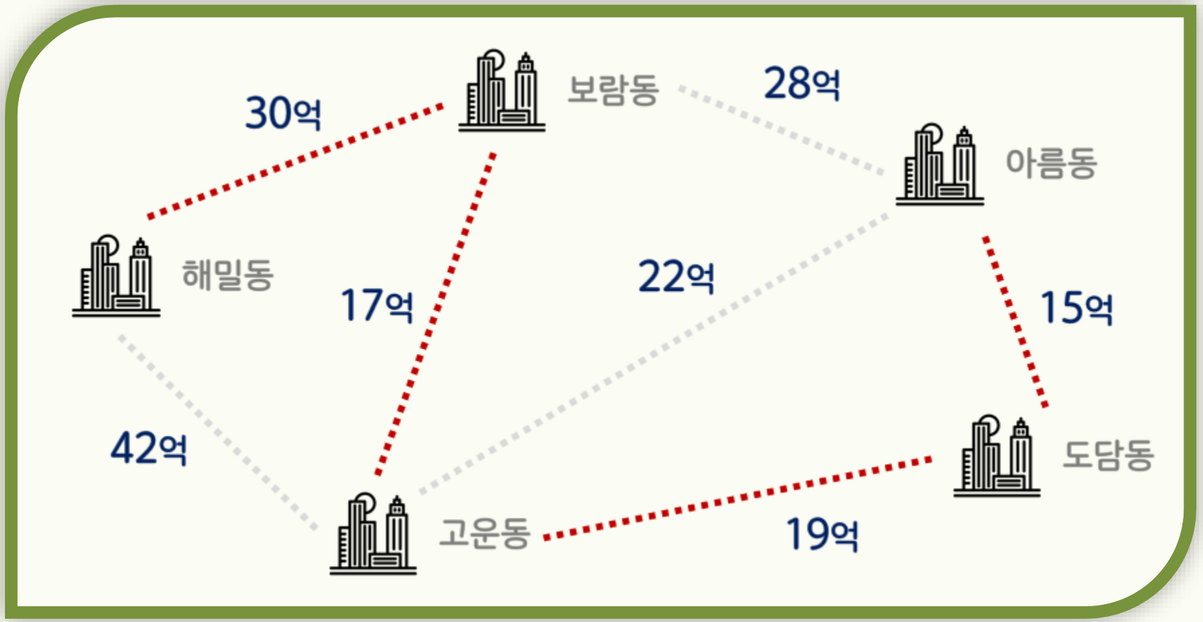
01 최소 신장 Tree 문제

서로 떨어져 있는 동네들…
가장 값싸게 연결하려면 어떻게 해야 하나?



01 최소 신장 Tree 문제

가중 그래프 응용의 시작, 최소 신장 Tree 찾기



Minimum Spanning Tree

- ✓ 주어진 가중 Graph에 존재하는 모든 Node를 연결되게끔 하는 Tree 가운데, 가중치의 합이 최소인 것

Q 가중치의 합이 최소인 구조 가운데 Tree만을 고려하는 이유는?

WEEK 12 / SPRING 2023

Graph에 기반한 최적 Tree 산출

THEME TODAY

NOW

01 최소 신장 Tree 문제

02 Prim Algorithm의 실제

03 Prim Algorithm의 개선

04 Kruskal Algorithm의 이해

05 Kruskal Algorithm의 구현

02 Prim Algorithm의 실제

정점에서 시작해 정점으로 끝낸다,
Prim Algorithm을 이용한 최소 신장 트리 구성

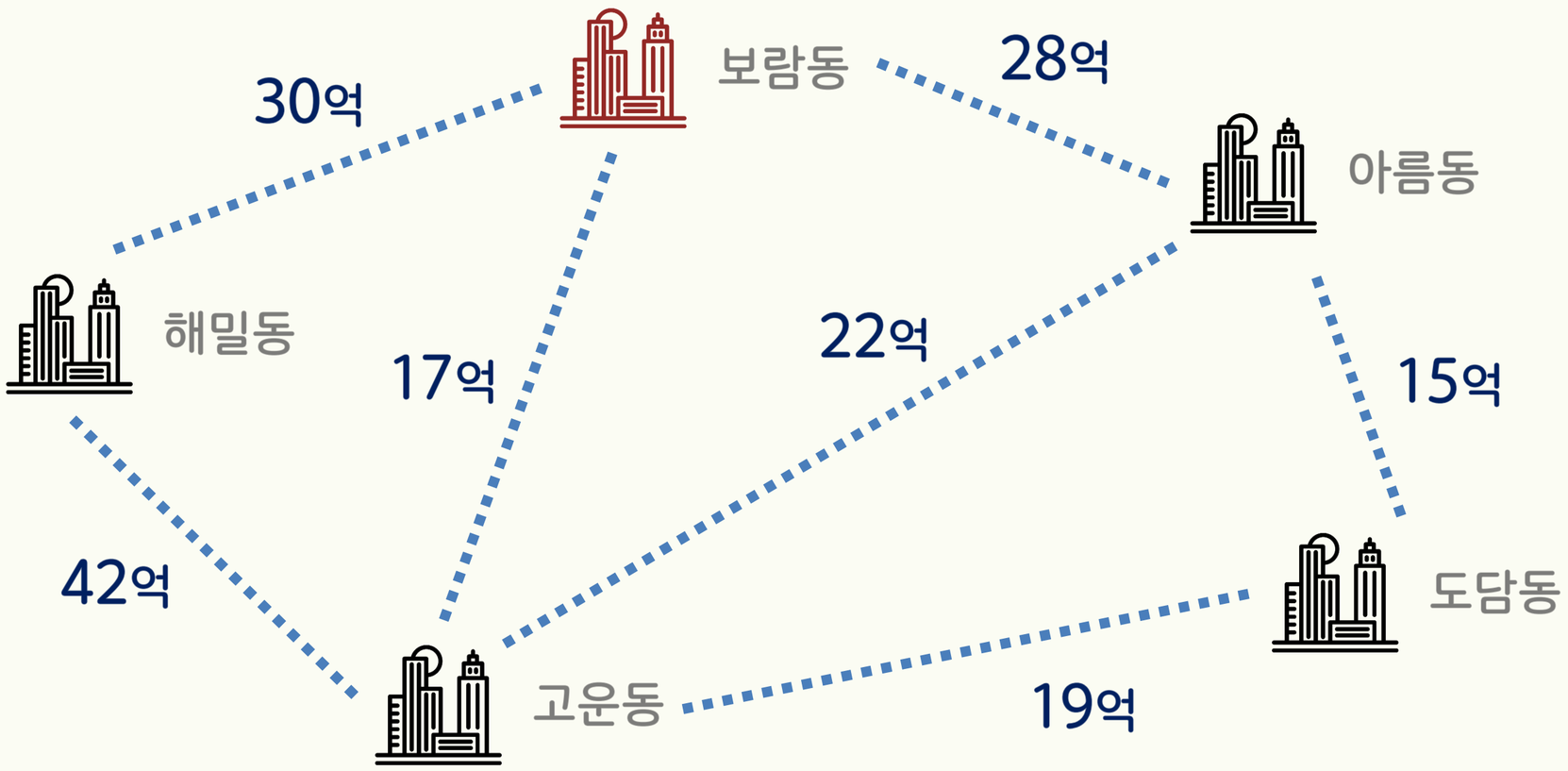
Prim Algorithm

- ✓ 임의의 정점 하나를 선택한 후, 가장 적은 비용으로
  해당 정점과 이을 수 있는 또 다른 정점 하나를 선택

- ✓ 두 정점이 선택된 상태에서, 역시나 가장 적은 비용으로
  두 정점 중 하나와 이을 수 있는 또 다른 정점 하나를 선택 및 반복

02 Prim Algorithm의 실제

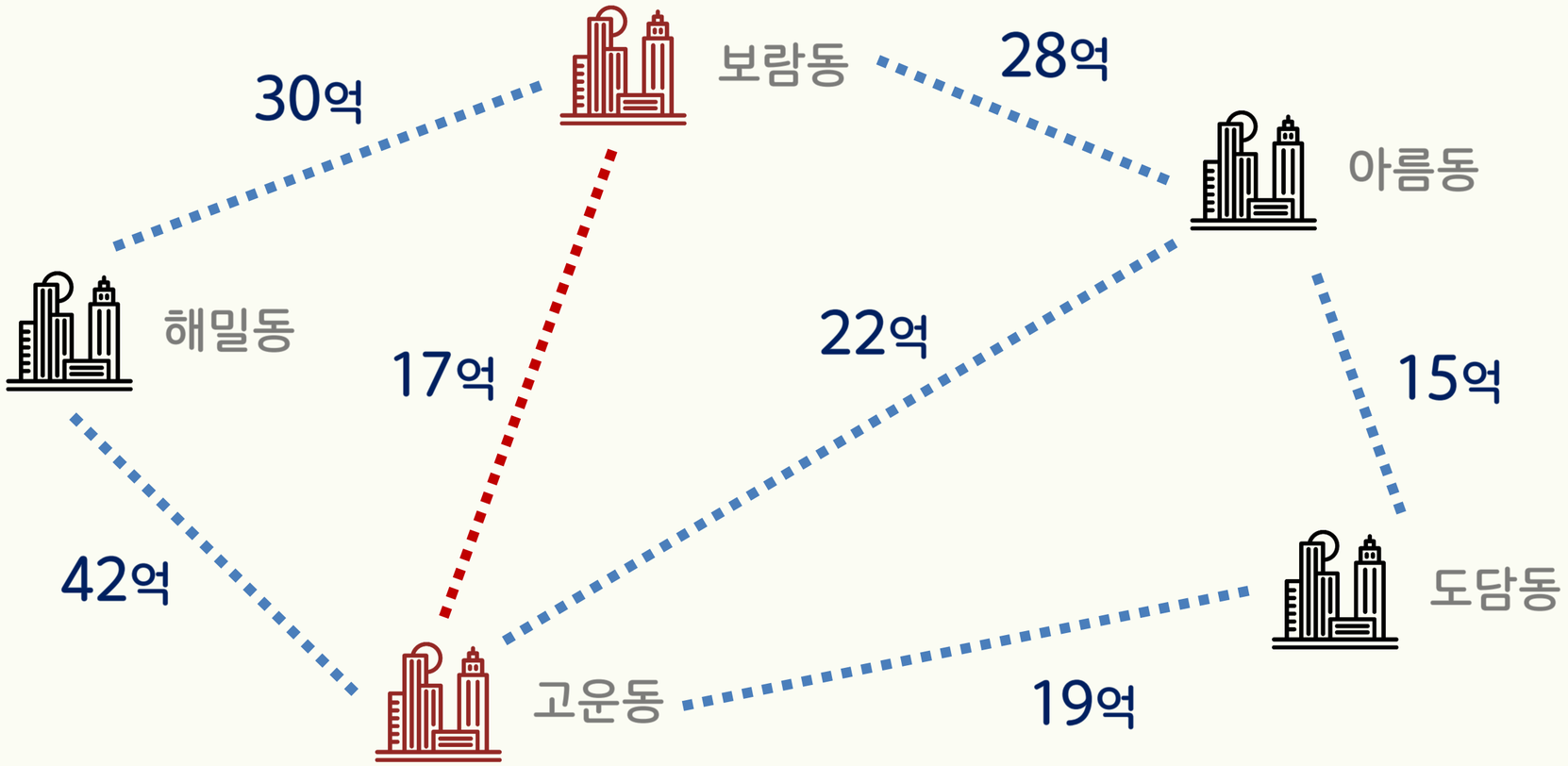
Prim Algorithm을 이용하여 최적의 노선도를 결정하면?



- 임의의 최초 정점을 보람동으로 선택
- 가장 값싸게 연결 가능한 곳은 고운동

02 Prim Algorithm의 실제

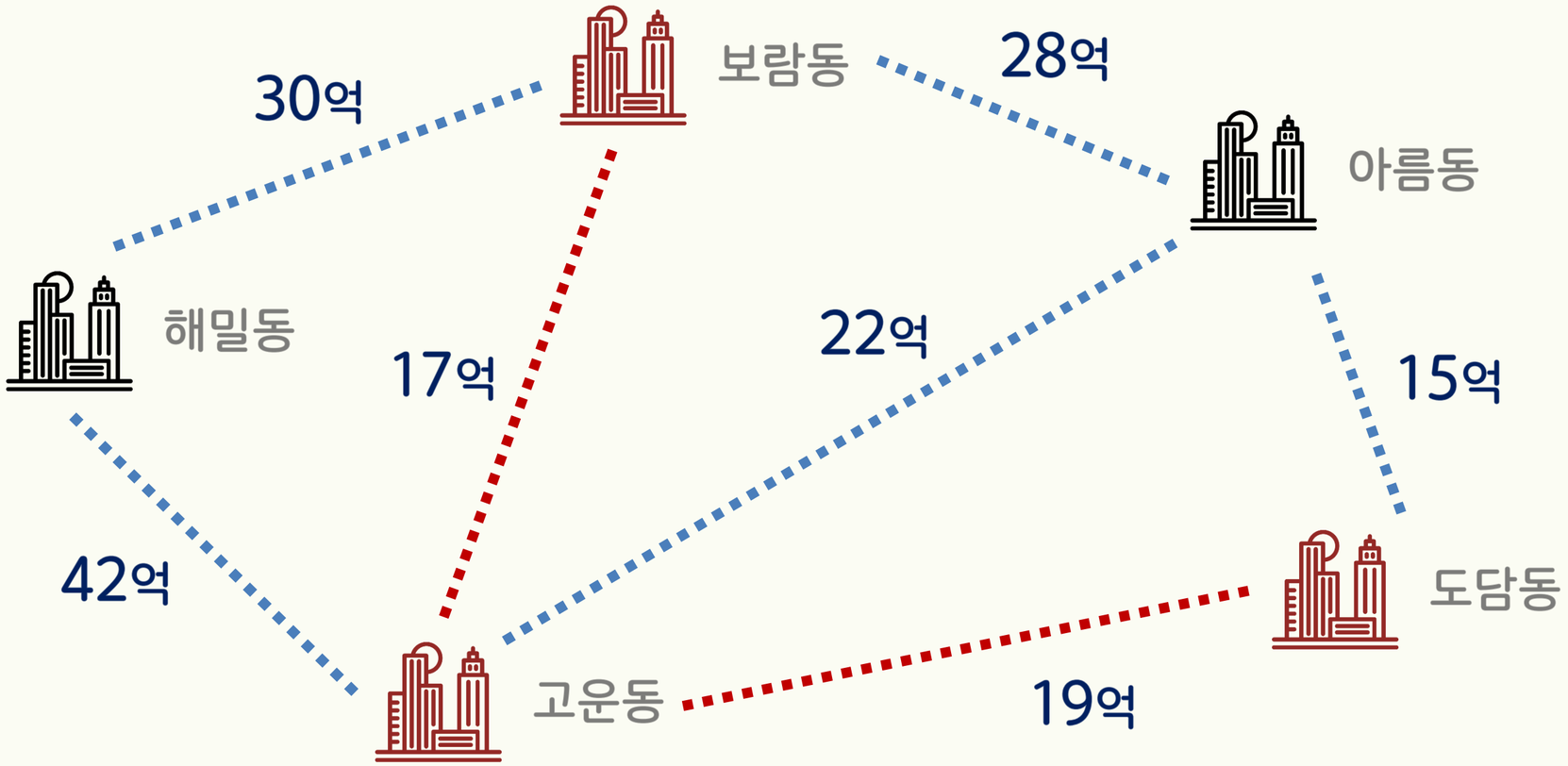
Prim Algorithm을 이용하여 최적의 노선도를 결정하면?



- 보람동과 고운동 중 아무 곳이나 가장 값싸게 연결해야 함
- 도담동을 고운동과 연결하는 것이 최적

02 Prim Algorithm의 실제

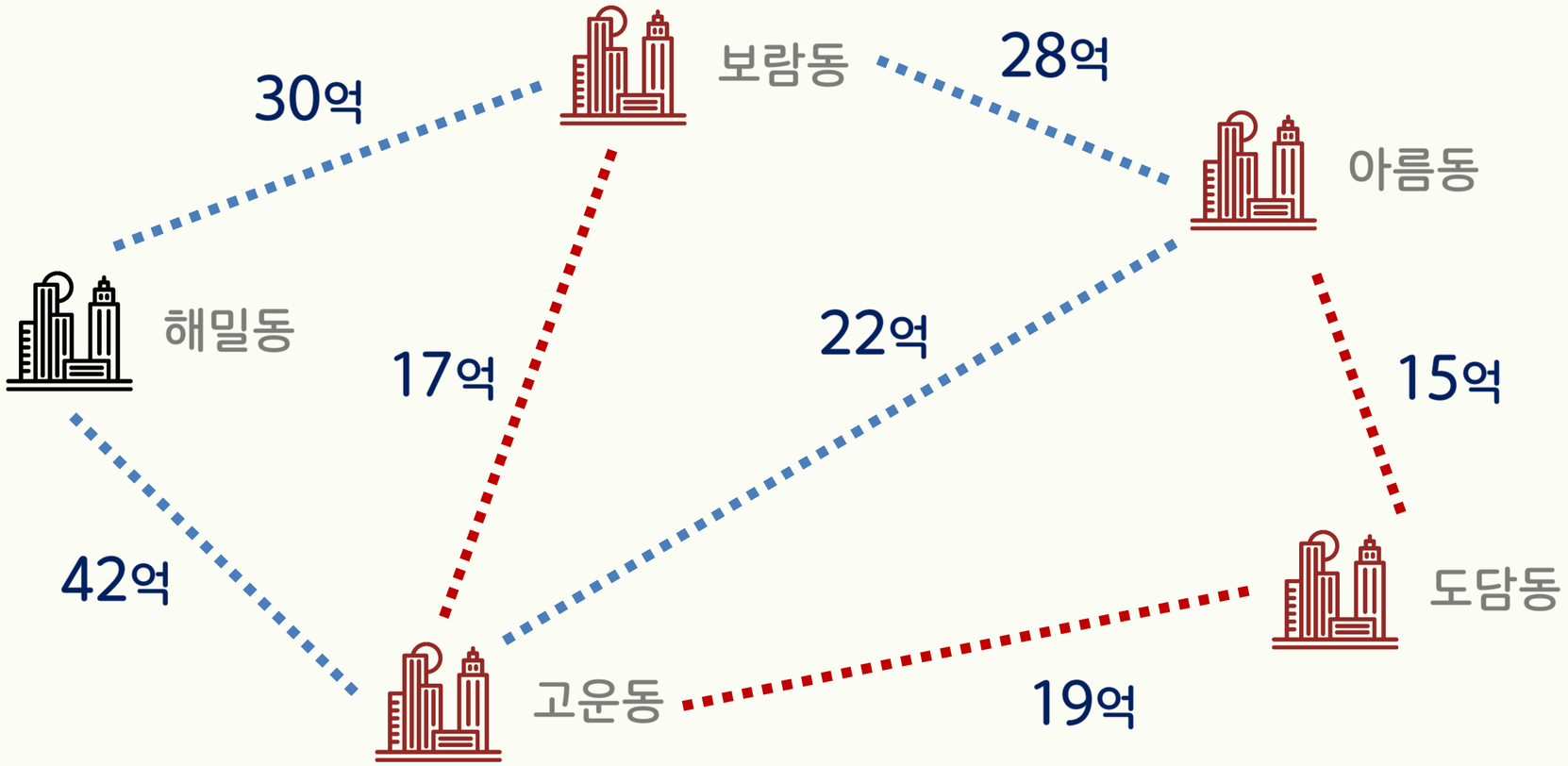
Prim Algorithm을 이용하여 최적의 노선도를 결정하면?



- 보람동과 고운동과 도담동 중 한 곳을 가장 값싸게 연결
- 아름동을 도담동과 연결하는 것이 최적

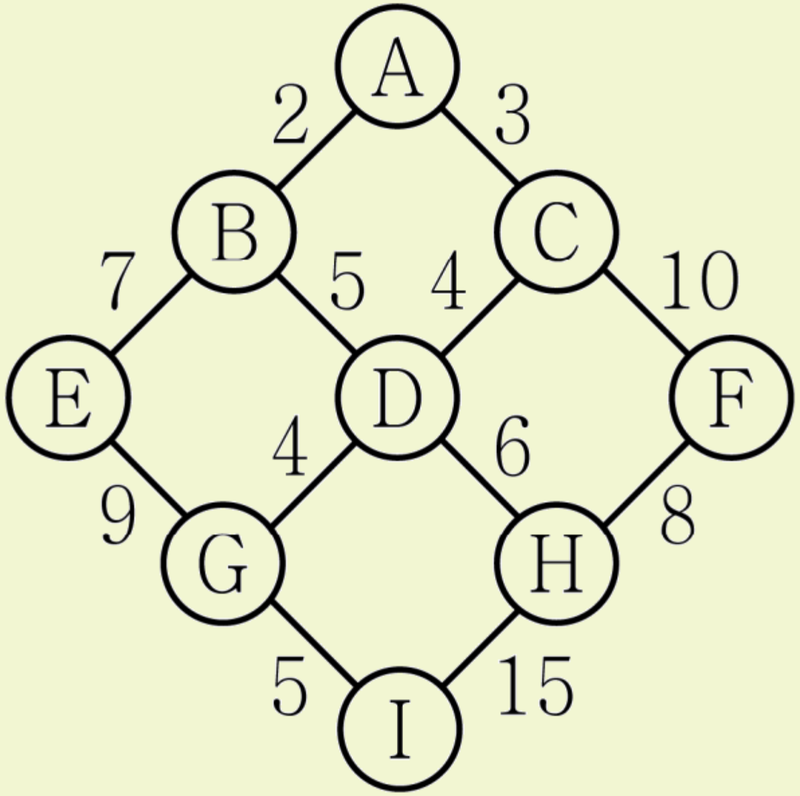
02 Prim Algorithm의 실제

Prim Algorithm을 이용하여 최적의 노선도를 결정하면?



- 마찬가지의 논리로 해밀동을 보람동과 연결하는 것이 최적
- 최소 신장 트리 산출

EXERCISE 1



왼쪽과 같은 가중 그래프에서 최소 비용 신장 트리를 구성하였을 때, 가중치의 합을 구하면?

WEEK 12 / SPRING 2023

Graph에 기반한 최적 Tree 산출

THEME TODAY

NOW

01 최소 신장 Tree 문제

02 Prim Algorithm의 실제

03 Prim Algorithm의 개선

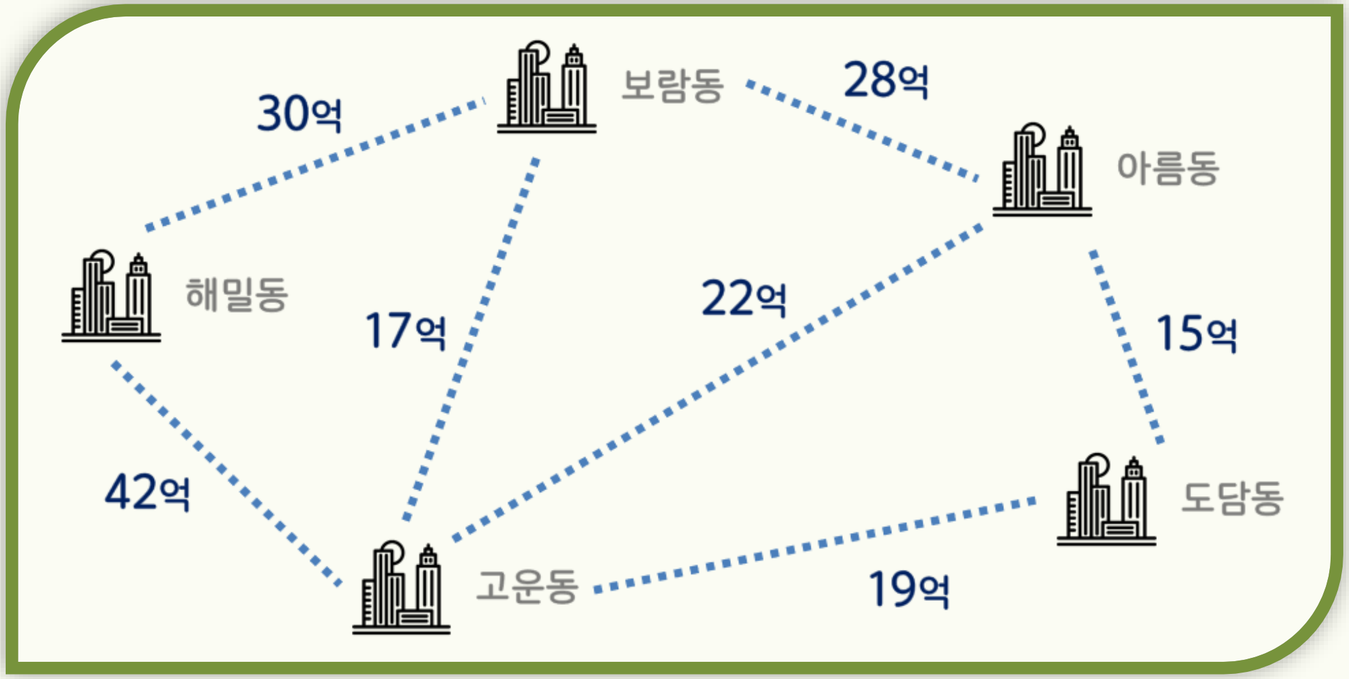
04 Kruskal Algorithm의 이해

05 Kruskal Algorithm의 구현

03 Prim Algorithm의 개선

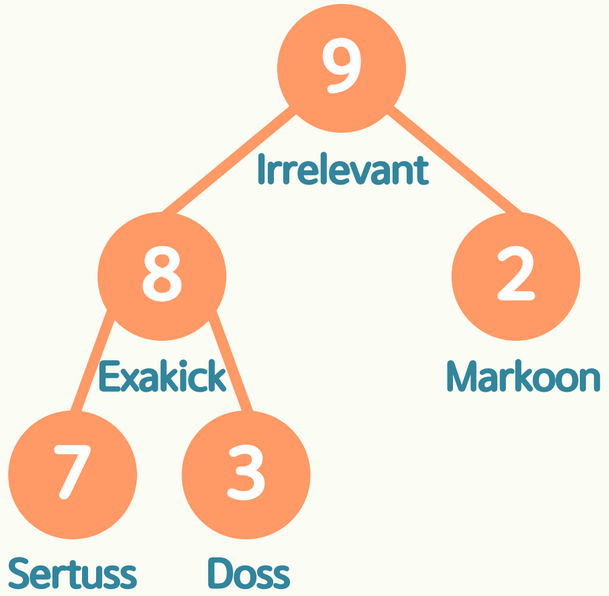
최소 신장 트리를 구하는 Prim Algorithm, 가장 직관적으로 구현해보면?

- 아직 연결되지 않은 노드들에 대하여 연결 비용을 update
- 그 중 최솟값을 취함



03 Prim Algorithm의 개선

[Flashback]
최솟값과 최댓값을 빠르게 찾는 자료구조는?



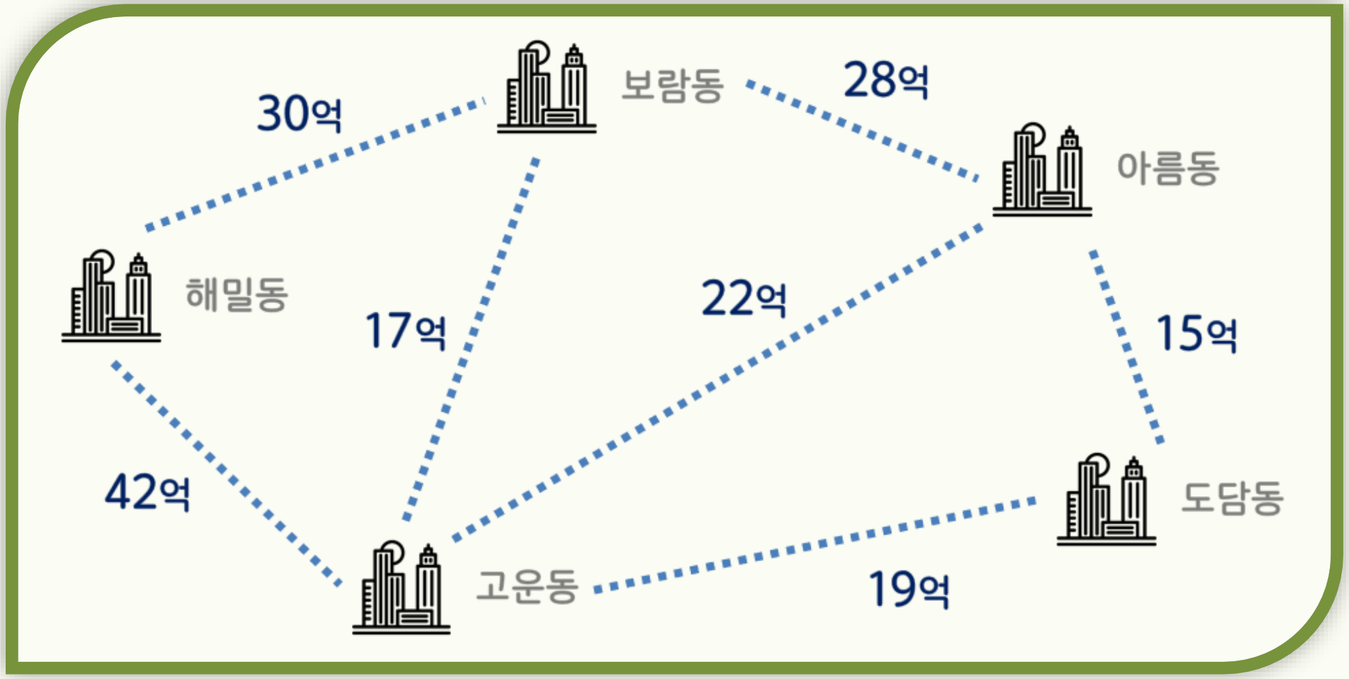
Heap을 이용한 우선순위 Queue의 구현

- ✓ Heap은 하나의 원소를 삽입할 때도, Root Node를 삭제할 때도 공히 O(logN)의 시간복잡도를 가진다는 특성을 지님
- ✓ 따라서, Heap을 이용해서 우선순위 Queue를 구현할 경우 삽입과 삭제를 모두 빠르게 안정적으로 수행할 수 있음

03 Prim Algorithm의 개선

$O(V^2)$보다 조금 더 빠를 수는 없나…
최소 비용을 찾기 위한 더 나은 접근

- 연결 비용 update 및 최솟값 탐색에서 우선순위 Queue 활용
- $O(V^2)$ 얼마나 개선?



03 Prim Algorithm의 개선

우선순위 Queue를 활용한 Prim Algorithm 구현의 실제

보다 효율적인 Prim Algorithm의 구현 Heap 활용을 위한 준비 및 connect 정의

```
from heapq import heappush
from heapq import heappop

def Prim(graph, graph_v, start) :

  h = list()
  connect = list()
  for i in range(0, len(graph)) :
    connect.append(False)

  heappush(h, (0, start))
  total_weight = 0
  vertex_count = 0
```

03 Prim Algorithm의 개선

우선순위 Queue를 활용한 Prim Algorithm 구현의 실제

보다 효율적인 Prim Algorithm의 구현 반복을 통한 최소 신장 Tree의 구성

```
while (vertex_count < len(graph)) :
  pop_info = heappop(h)
  pop_weight = pop_info[0]
  pop_node = pop_info[1]
  if connect[pop_node] == False :
    connect[pop_node] = True
    total_weight = total_weight + pop_weight
    vertex_count = vertex_count + 1
    print('새로 연결된 곳 :', graph_v[pop_node])
    print('누적 가중치 합 :', total_weight)
    for i in range(0, len(graph)) :
      if (graph[pop_node][i] != 0) and (connect[i] == False) :
        heappush(h, (graph[pop_node][i], i))
```

03 Prim Algorithm의 개선

우선순위 Queue를 활용한 Prim Algorithm 구현의 실제

Graph와 Node를 정의하고 Prim 함수를 호출하기

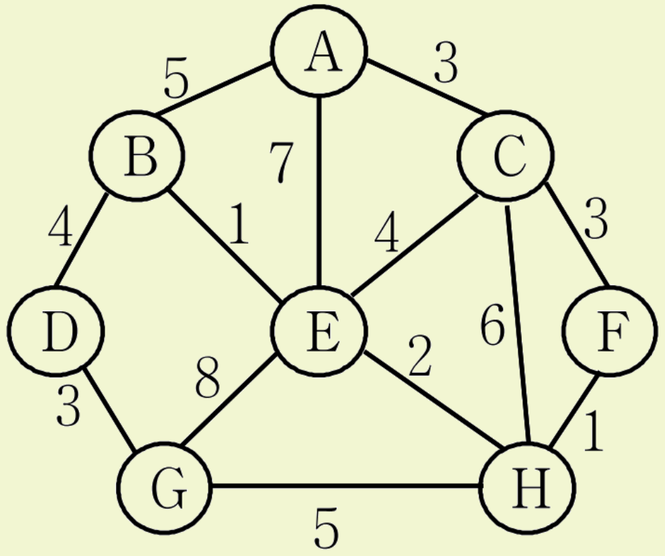
```
G = [[0, 30, 28, 17, 0],
     [30, 0, 0, 42, 0],
     [28, 0, 0, 22, 15],
     [17, 42, 22, 0, 19],
     [0, 0, 15, 19, 0]]

V = ['보람동', '해밀동', '아름동', '고운동', '도담동']

Prim(G, V, 0)
```

EXERCISE 2

오른쪽 가중 그래프에 대하여 Prim 알고리즘을 적용해 최소 신장 트리를 구성한다고 할 때, 옳은 것만을 〈보기〉에서 있는 대로 고르면?



〈보기〉

ㄱ. 우선순위 큐를 이용하여 구현할 수 있다.

ㄴ. 탐욕 기법에 기초하여 알고리즘이 동작한다.

ㄷ. 마지막으로 선택되는 정점은 G이다.

WEEK 12 / SPRING 2023

Graph에 기반한 최적 Tree 산출

THEME TODAY

01 최소 신장 Tree 문제

02 Prim Algorithm의 실제

03 Prim Algorithm의 개선

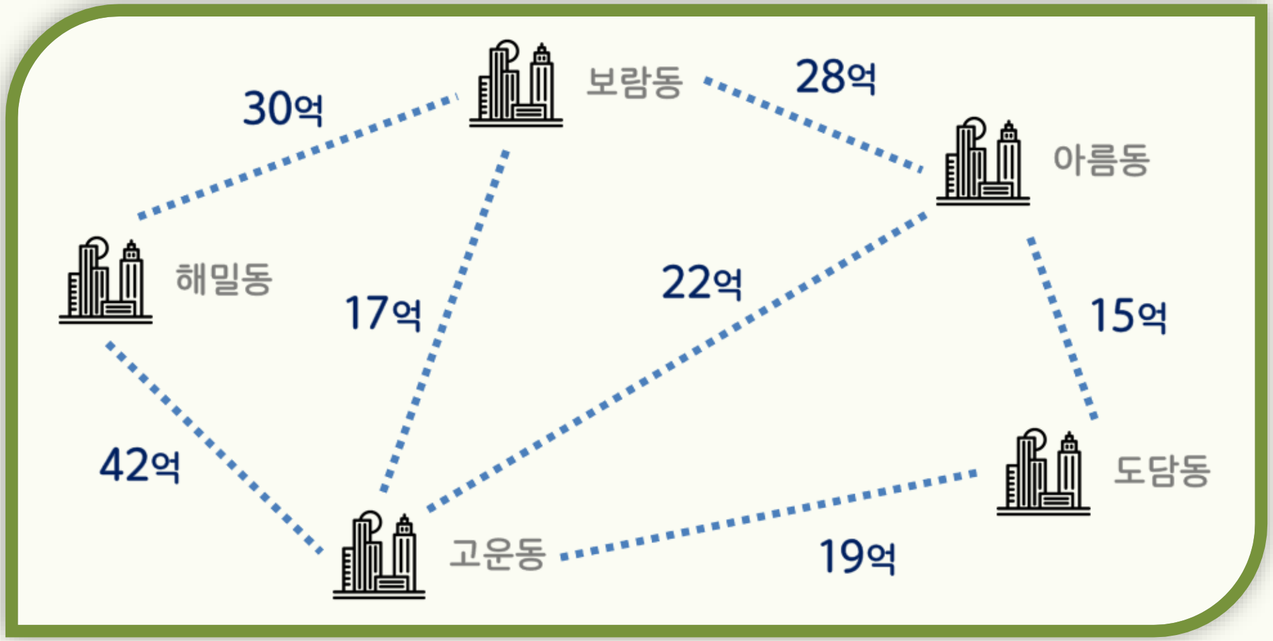
NOW

04 Kruskal Algorithm의 이해

05 Kruskal Algorithm의 구현

04 Kruskal Algorithm의 이해

정점 말고, 간선에 집중하면 안 되나? Kruskal Algorithm의 시작



- ✓ 왼쪽 가중 Graph에 존재하는 모든 간선들을 가중치 순으로 나열하면 15억 → 17억 → … → 30억 → 42억
- ✓ 다섯 개의 동이 모두 이어질 때까지 가중치가 낮은 간선 순서로 택하면?

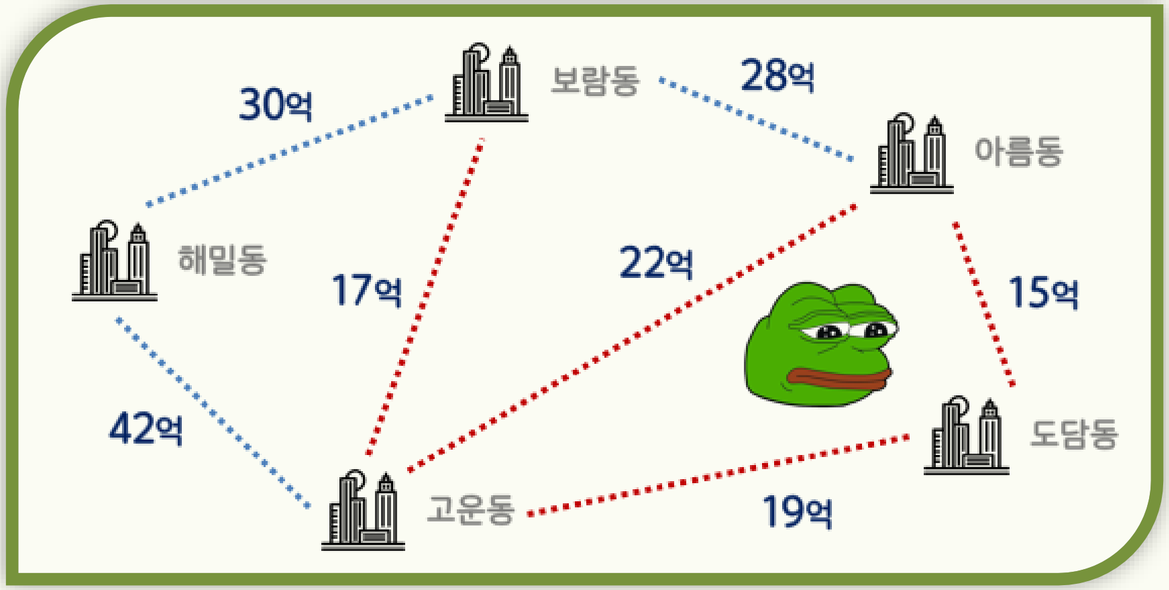
04 Kruskal Algorithm의 이해

Cycle만 이루어지지 않게끔
가중치가 낮은 간선부터, Kruskal Algorithm

Kruskal Algorithm

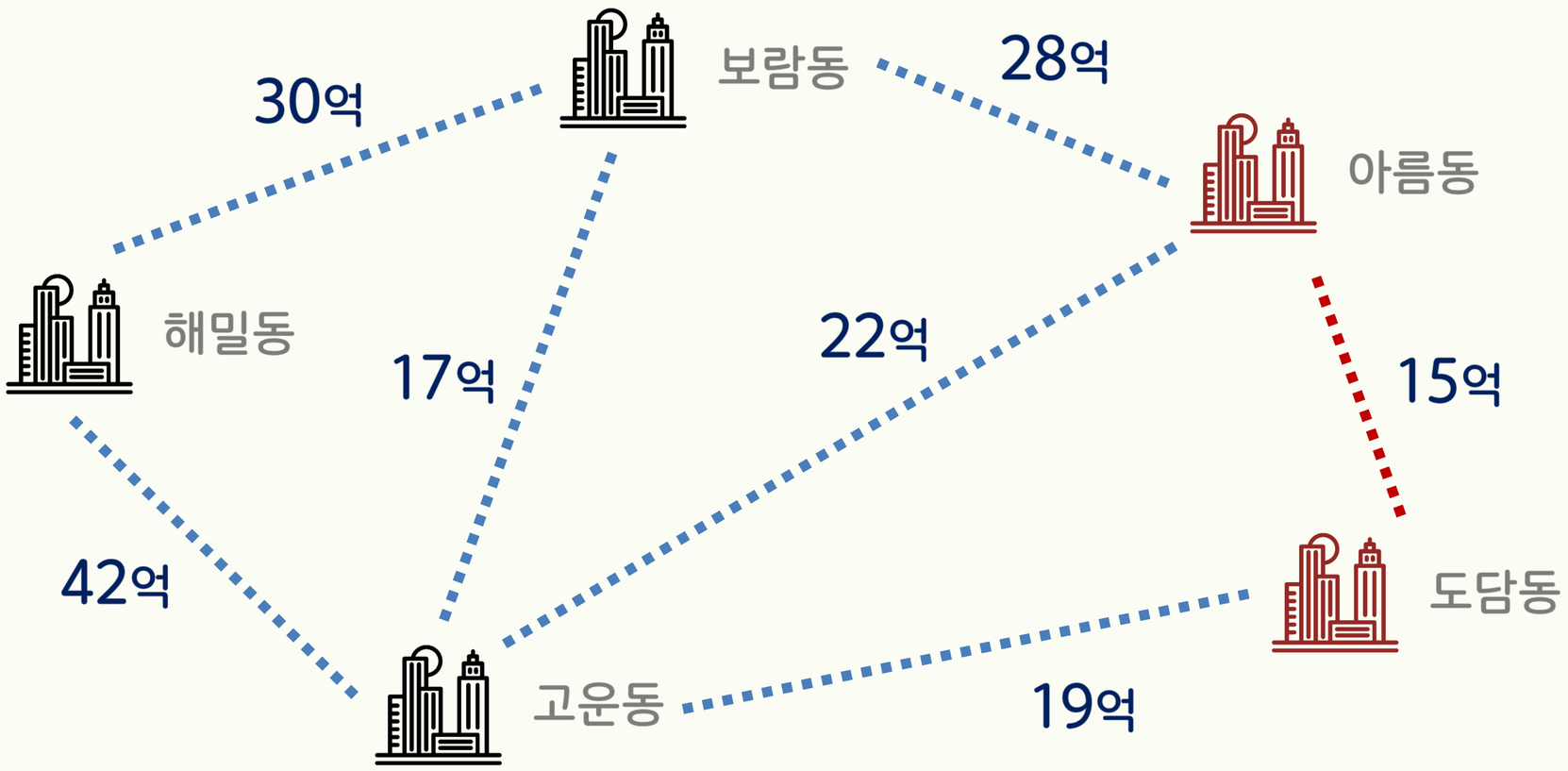
✓ 주어진 가중 Graph 내에 존재하는 모든 간선을
가중치 순으로 나열함으로써 최소 신장 Tree 구성

✓ 가중치가 낮은 간선부터 순차적으로 택하되,
우측과 같이 Cycle이 이루어지는 경우는 배제함



04 Kruskal Algorithm의 이해

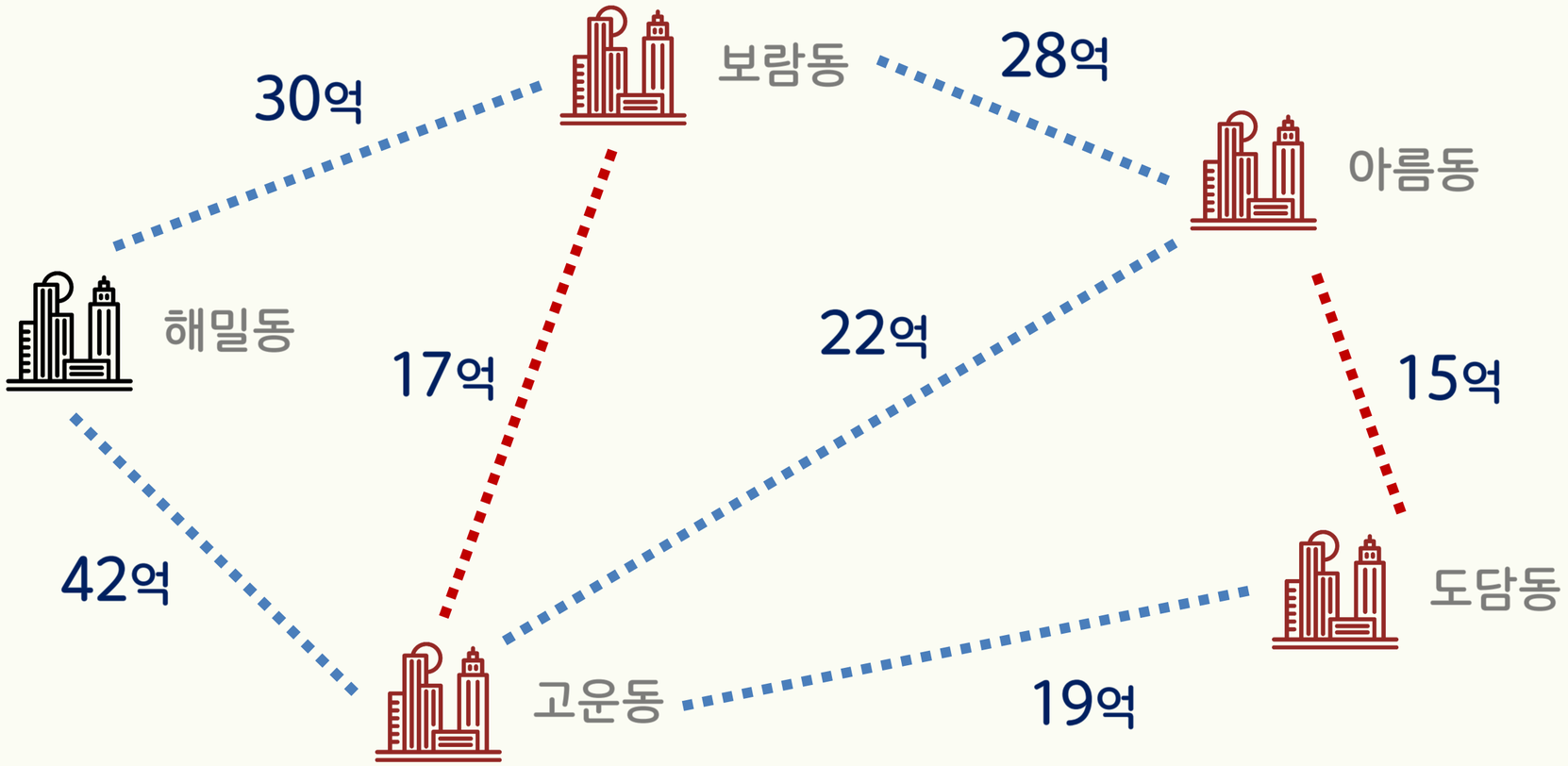
Kruskal Algorithm 실전, 어떻게 최소 신장 Tree가 구성되는가?



- 15 / 아름동 / 도담동
- 17 / 보람동 / 고운동
- 19 / 고운동 / 도담동
- 22 / 아름동 / 고운동
- 28 / 보람동 / 아름동
- 30 / 보람동 / 해밀동
- 42 / 해밀동 / 고운동

04 Kruskal Algorithm의 이해

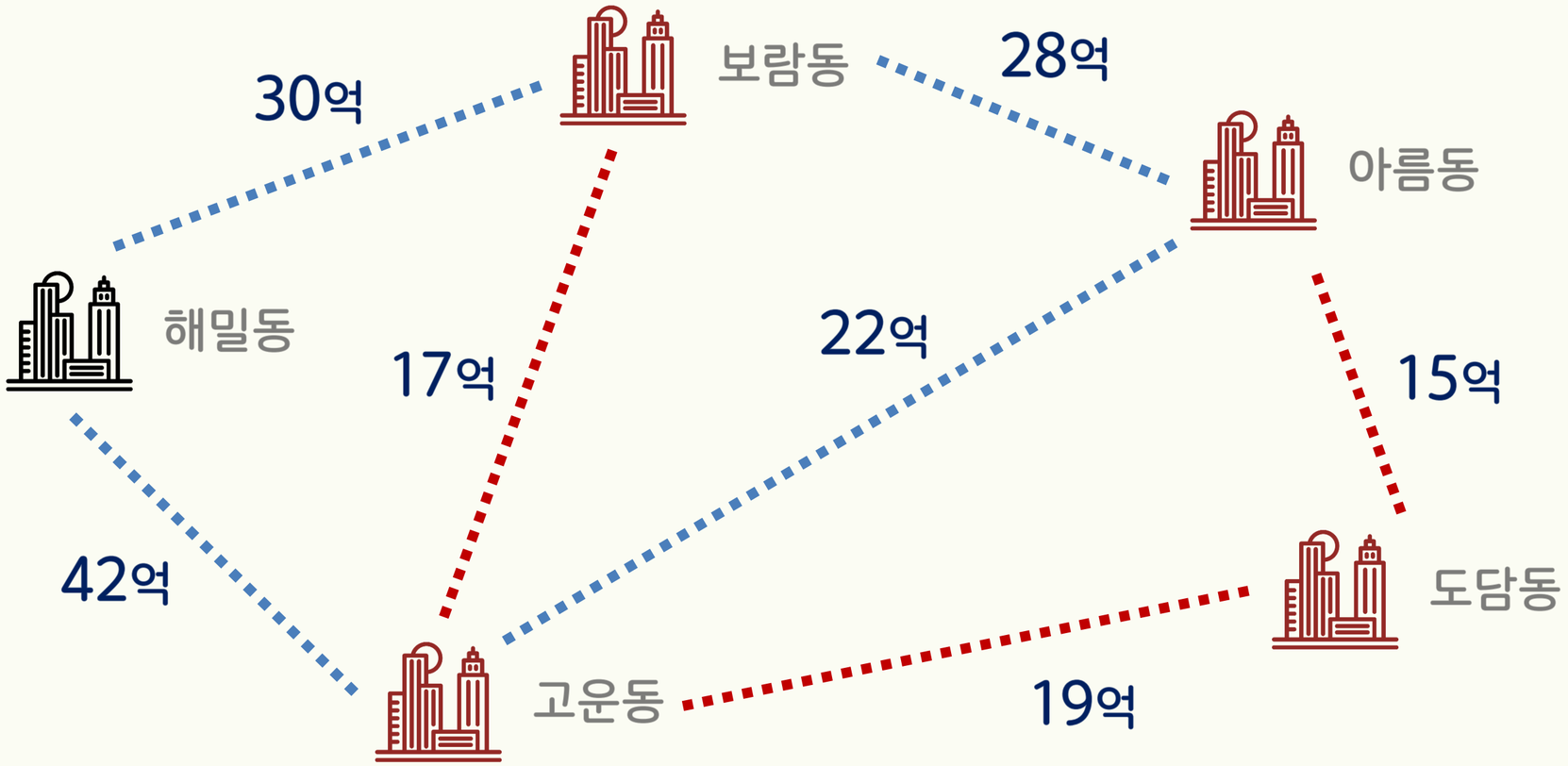
Kruskal Algorithm 실전, 어떻게 최소 신장 Tree가 구성되는가?



- 17 / 보람동 / 고운동
- 19 / 고운동 / 도담동
- 22 / 아름동 / 고운동
- 28 / 보람동 / 아름동
- 30 / 보람동 / 해밀동
- 42 / 해밀동 / 고운동

04 Kruskal Algorithm의 이해

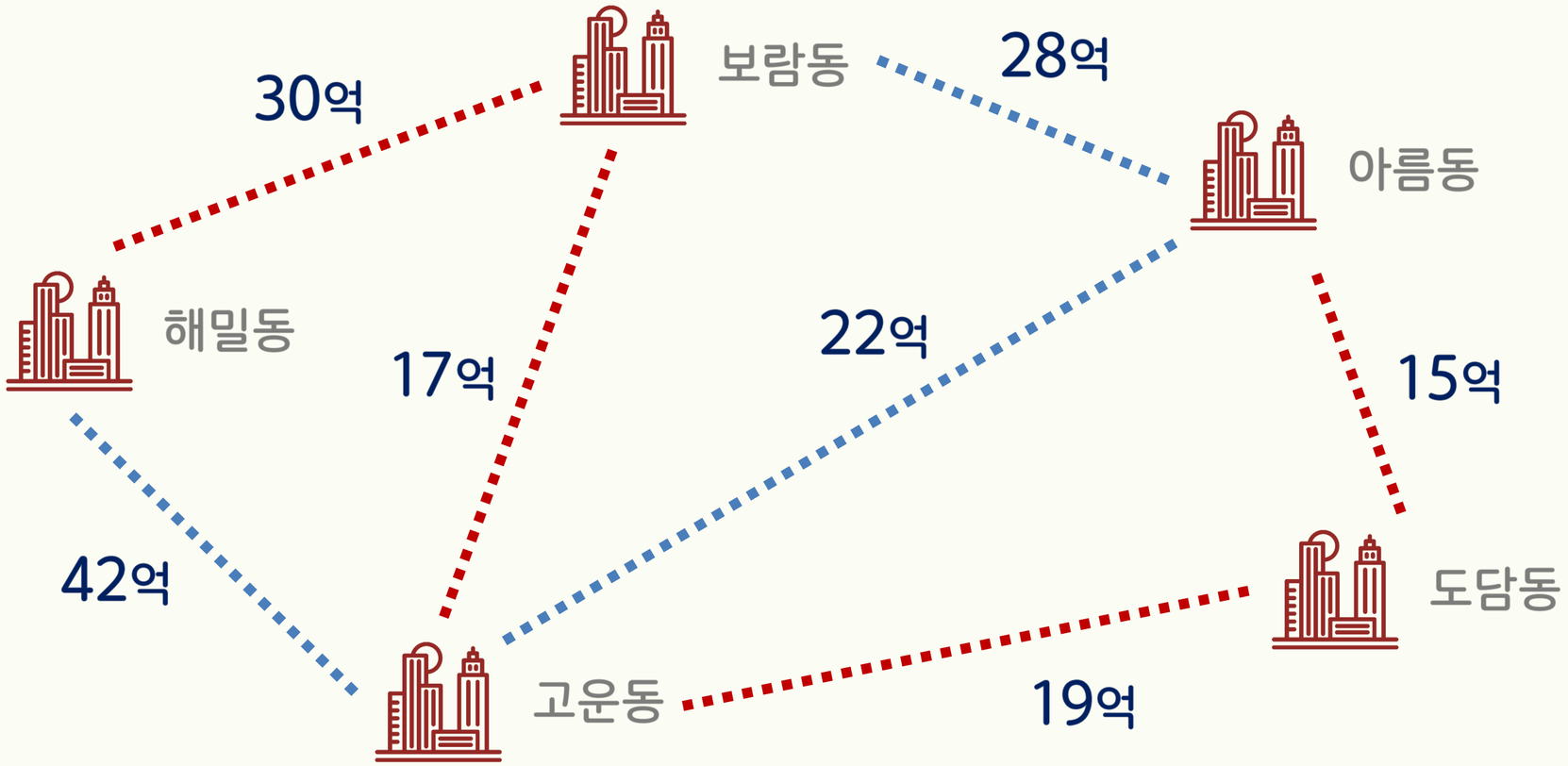
Kruskal Algorithm 실전, 어떻게 최소 신장 Tree가 구성되는가?



- 19 / 고운동 / 도담동
- 22 / 아름동 / 고운동
- 28 / 보람동 / 아름동
- 30 / 보람동 / 해밀동
- 42 / 해밀동 / 고운동

04 Kruskal Algorithm의 이해

Kruskal Algorithm 실전, 어떻게 최소 신장 Tree가 구성되는가?

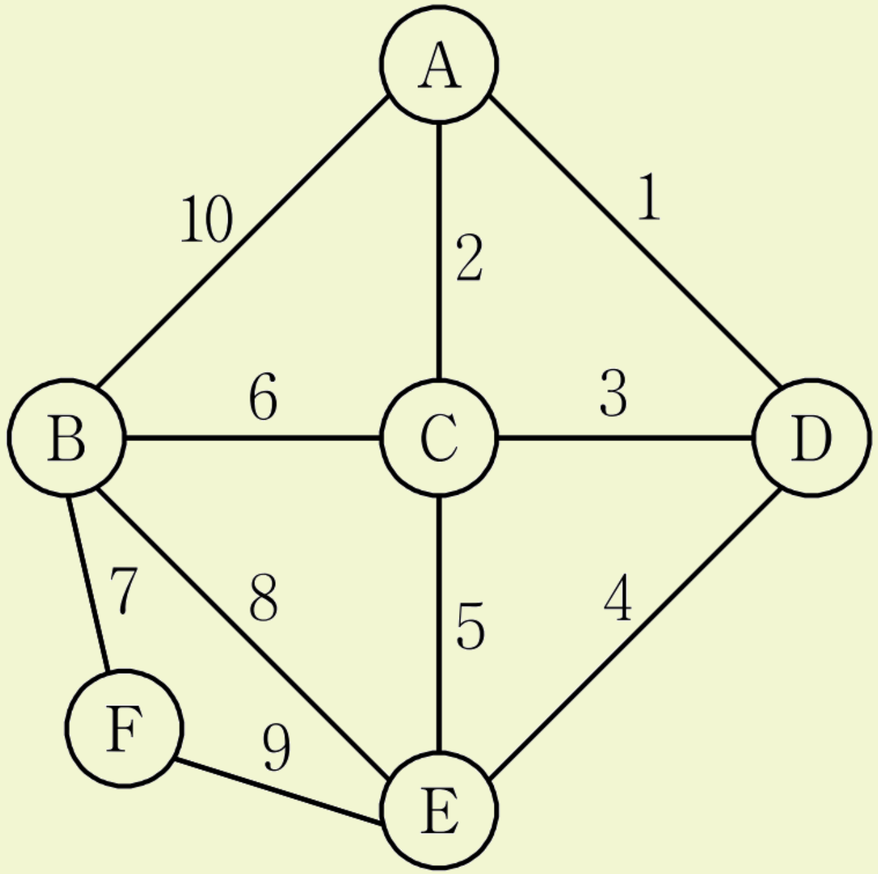


- ~~22 / 아름동 / 고운동~~
- ~~28 / 보람동 / 아름동~~
- 30 / 보람동 / 해밀동
- 42 / 해밀동 / 고운동

EXERCISE 3

오른쪽 그래프에 대하여 최소 신장 트리를 구성한다고 할 때,
선택되지 않는 간선은?

① (A, C)
② (B, C)
③ (B, F)
④ (C, D)



WEEK 12 / SPRING 2023

Graph에 기반한 최적 Tree 산출

THEME TODAY

01 최소 신장 Tree 문제

02 Prim Algorithm의 실제

03 Prim Algorithm의 개선

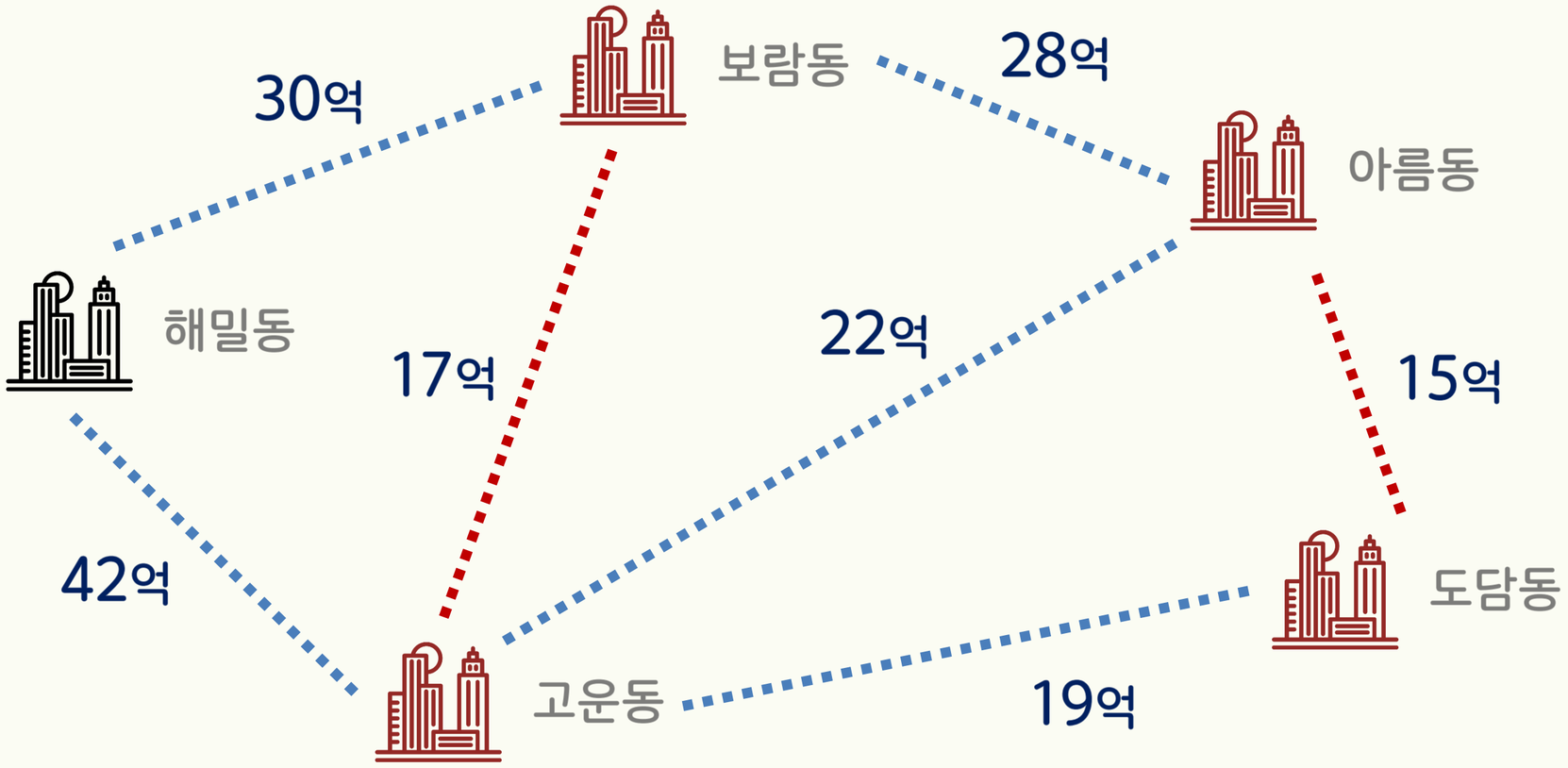
04 Kruskal Algorithm의 이해

NOW

05 Kruskal Algorithm의 구현

05 Kruskal Algorithm의 구현

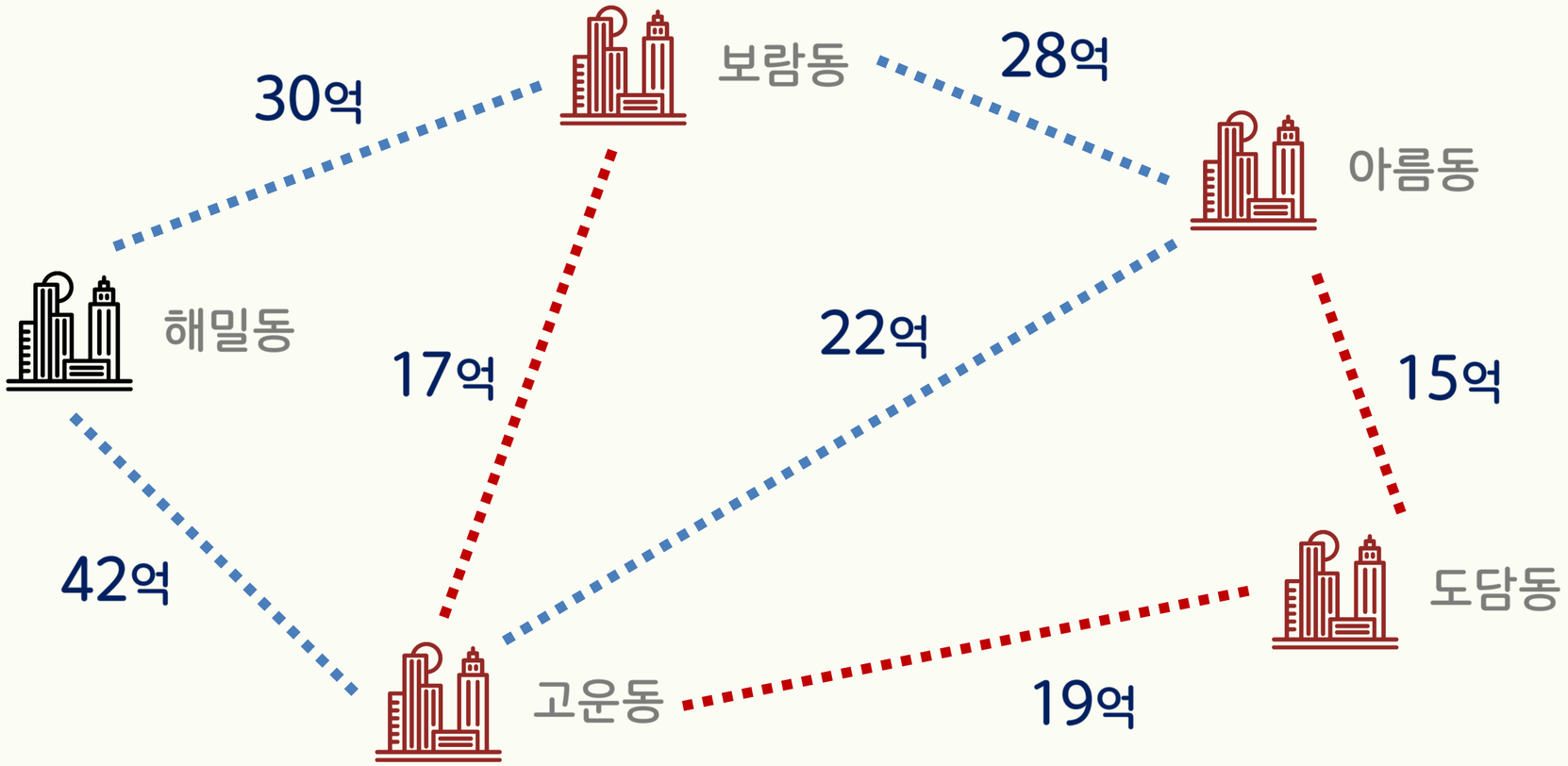
가중치 순서로 삽입하는 것은 무난한데…
대체 어떻게 Cycle 여부를 판단한다는 건지?



- {해밀동} / {보람동, 고운동} / {도담동, 아름동}
- 고운동과 도담동을 잇는 것은 서로 다른 집합끼리 잇는 것

05 Kruskal Algorithm의 구현

가중치 순서로 삽입하는 것은 무난한데…
대체 어떻게 Cycle 여부를 판단한다는 건지?



- {해밀동} / {보람동, 고운동, 도담동, 아름동}
- 고운동과 아름동을 잇는 것은 단일한 집합 내부를 잇는 것

Union의 원리로 단숨에 끝낸다, Kruskal Algorithm의 실제적 구현

Kruskal Algorithm을 이용한 최소 신장 Tree 산출 Union 최초 정의 및 간선 정렬

```
def Kruskal(edges, vertexes) :

  edge_count = 0
  total_weight = 0

  union = dict()
  for each_vertex in vertexes :
    union[each_vertex] = each_vertex

  edges.sort()
```

05 Kruskal Algorithm의 구현

Union의 원리로 단숨에 끝낸다, Kruskal Algorithm의 실제적 구현

Kruskal Algorithm을 이용한 최소 신장 Tree 산출 Cycle을 구성하지 않도록 간선을 반복 선택

```
for each_edge in edges :

  if (union[each_edge[1]] != union[each_edge[2]]) :
    total_weight = total_weight + each_edge[0]
    edge_count = edge_count + 1
    print(each_edge[1], '과', each_edge[2], '이 연결되었습니다.')
    print('누적 가중치 합은', total_weight, '입니다.')
    new_value = union[each_edge[1]]
    old_value = union[each_edge[2]]
    for each_vertex in vertexes :
      if (union[each_vertex] == old_value) :
        union[each_vertex] = new_value

  if edge_count >= (len(vertexes) - 1) :
    break
```

05 Kruskal Algorithm의 구현

Union의 원리로 단숨에 끝낸다, Kruskal Algorithm의 실제적 구현

Graph와 Node를 정의하고 Kruskal 함수를 호출하기

```
E = [(30, '보람동', '해밀동'), (28, '보람동', '아름동'),
     (17, '보람동', '고운동'), (22, '아름동', '고운동'),
     (42, '해밀동', '고운동'), (19, '고운동', '도담동'),
     (15, '아름동', '도담동')]

V = ['보람동', '해밀동', '아름동', '고운동', '도담동']

Kruskal(E, V)
```

Any Questions ?

유용재 | 정보대학 컴퓨터학과 | yooy@korea.ac.kr