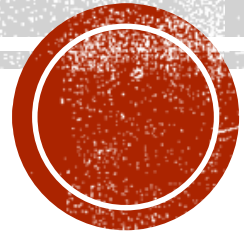
Lect07. A* Algorithm



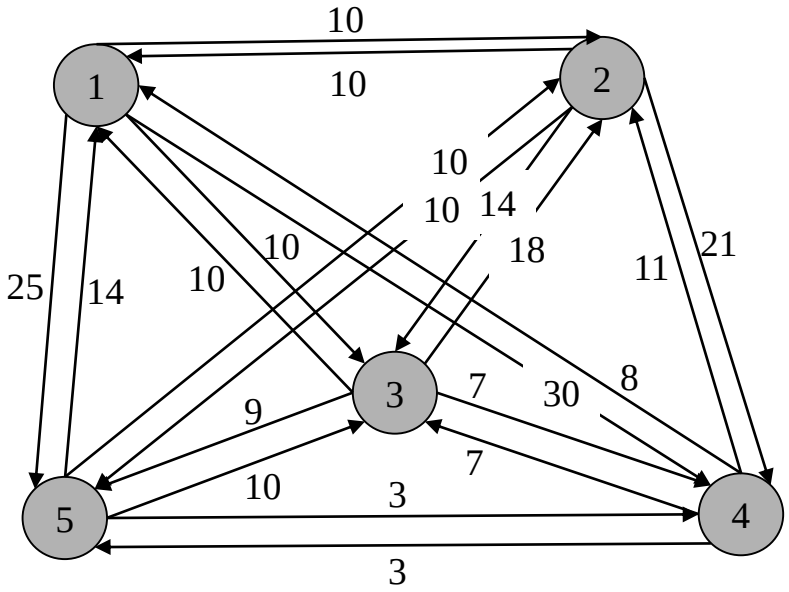
State-Space Tree

- **State-space tree** (상태공간트리)
  - 문제 해결 과정의 중간 상태를 각각 한 노드로 나타낸 트리
- 세가지 상태공간 탐색 기법
  - Backtracking
  - Branch-and-bound
  - A* algorithm

Branch-and-Bound (review)

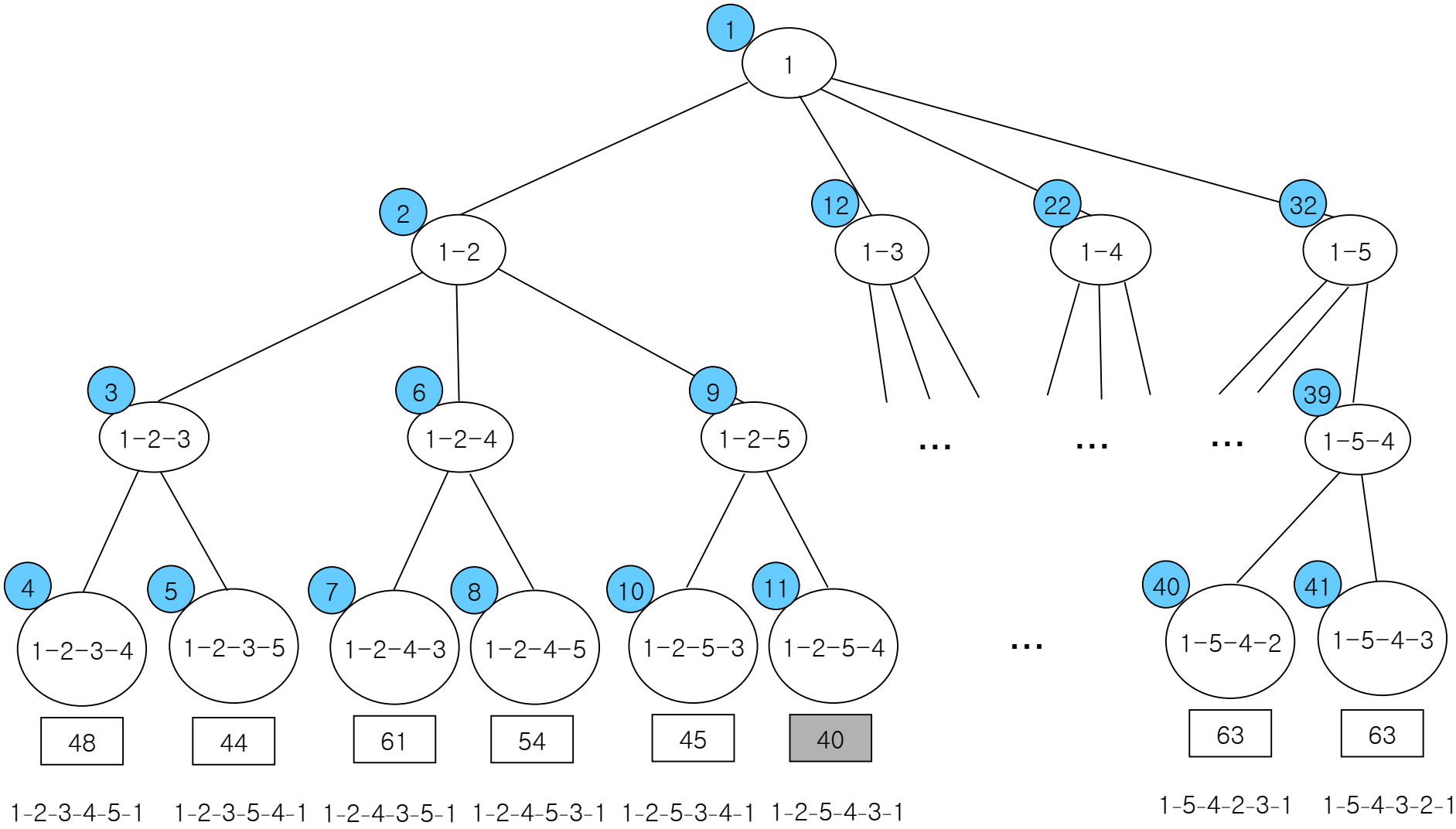
- 분기branch와 한정bound의 결합
  - 분기를 한정시켜 쓸데없는 시간 낭비를 줄이는 방법
- **Backtracking**과 공통점, 차이점
  - 공통점
    - 경우들을 차례로 나열하는 방법 필요
  - 차이점
    - **Backtracking** – 가보고 더 이상 진행이 되지 않으면 돌아온다
    - **Branch-&-Bound** – 최적해를 찾을 가능성이 없으면 분기는 하지 않는다

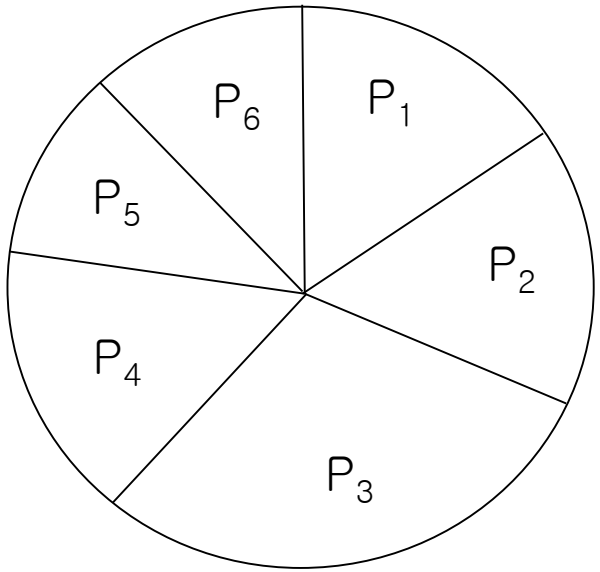
Backtracking vs Branch&Bound (TSP)



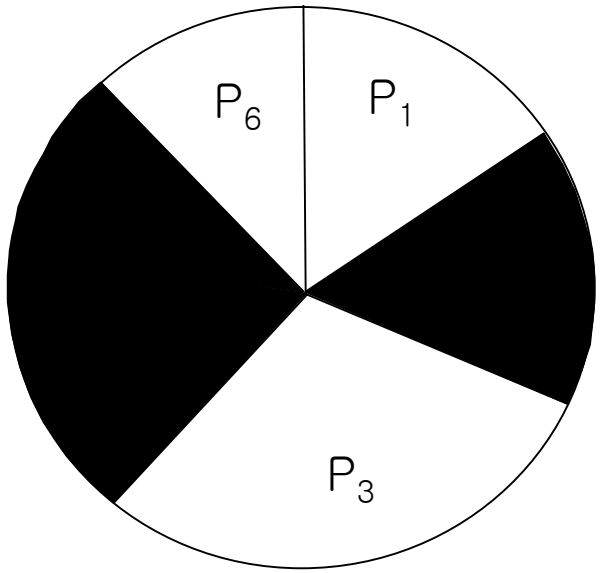
	1	2	3	4	5
1	0	10	10	30	25
2	10	0	14	21	10
3	10	18	0	7	9
4	8	11	7	0	3
5	14	10	10	3	0

State-Space Tree for TSP



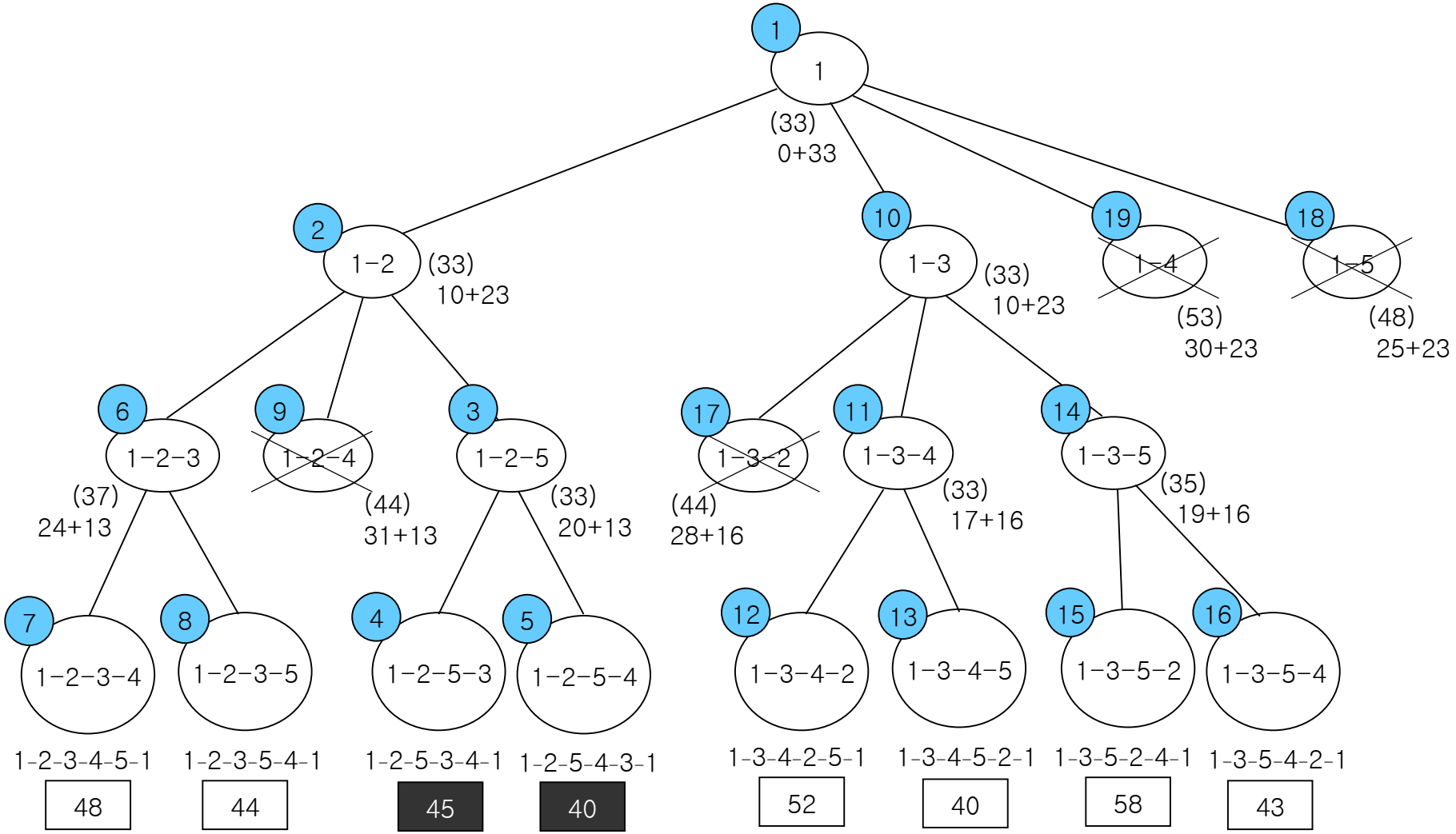


어느 시점에 가능한 선택들



최적해를 포함하지 않아 제외하는 선택들

TSP 예제를 대상으로 한 Branch-and-Bound 탐색의 예 (State-Space Tree)



A* Algorithm

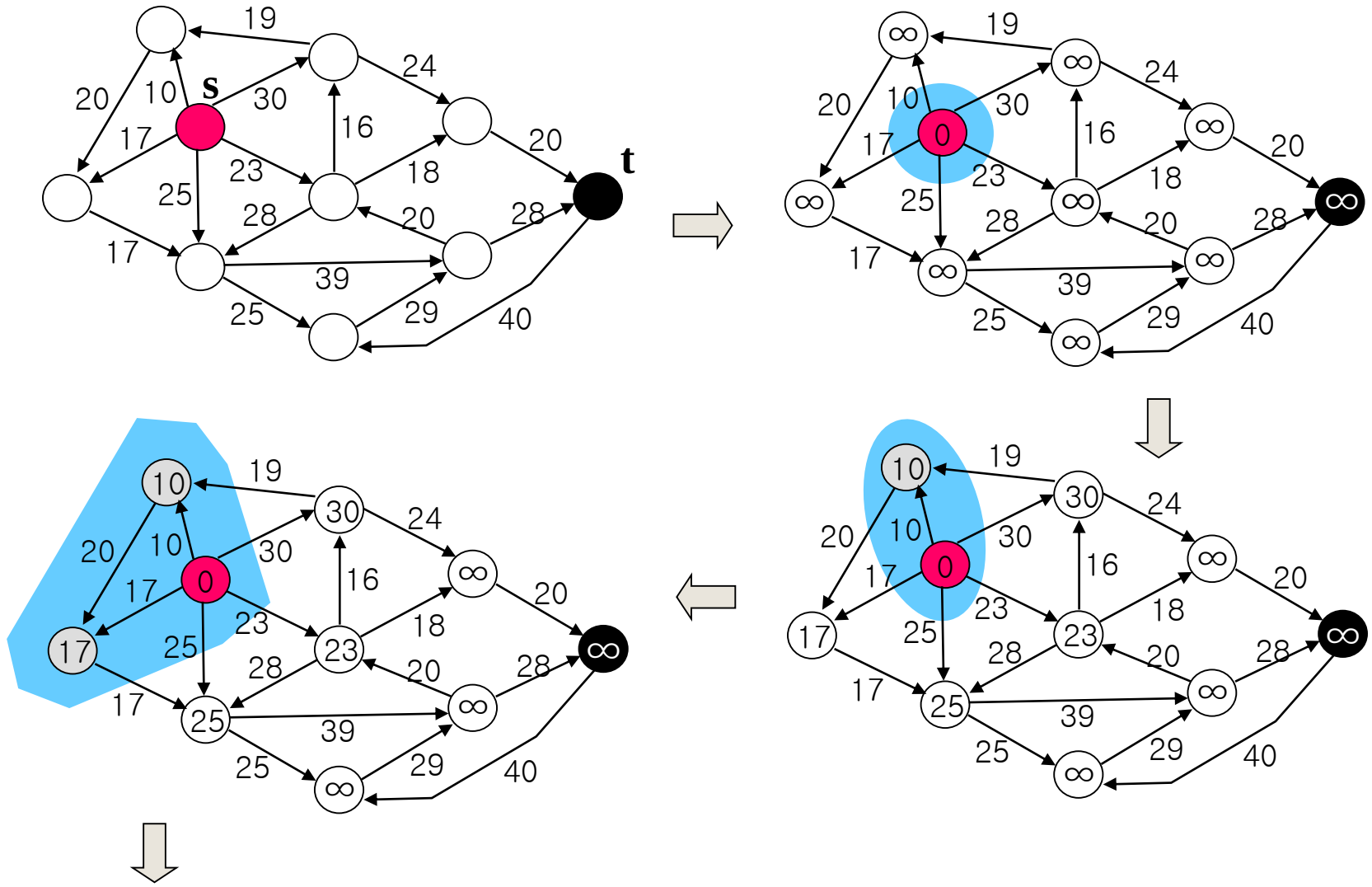
- **Best-First-Search**
  - 각 정점이 매력함수값 **g(*x*)**를 갖고 있다
  - 방문하지 않은 정점들 중 **g(*x*)** 값이 가장 매력적인 것부터 방문한다

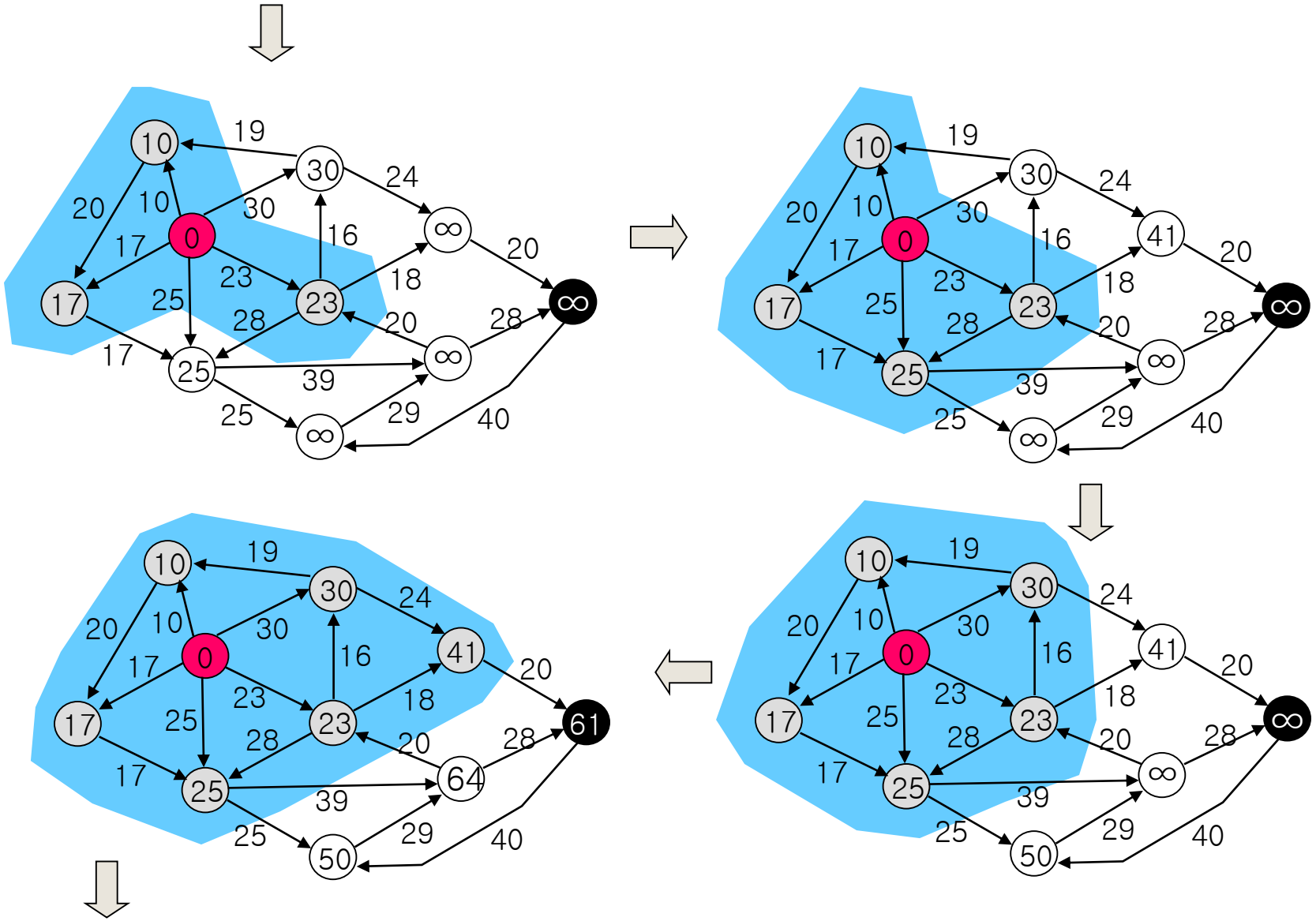
- **A* algorithm**은 **best-first search**에 목적점에 이르는 잔여추정거리를 고려하는 알고리즘이다
  - **Vertex *x***로부터 목적점에 이르는 잔여거리의 추정치 **h(*x*)**는 실제치보다 크면 안된다

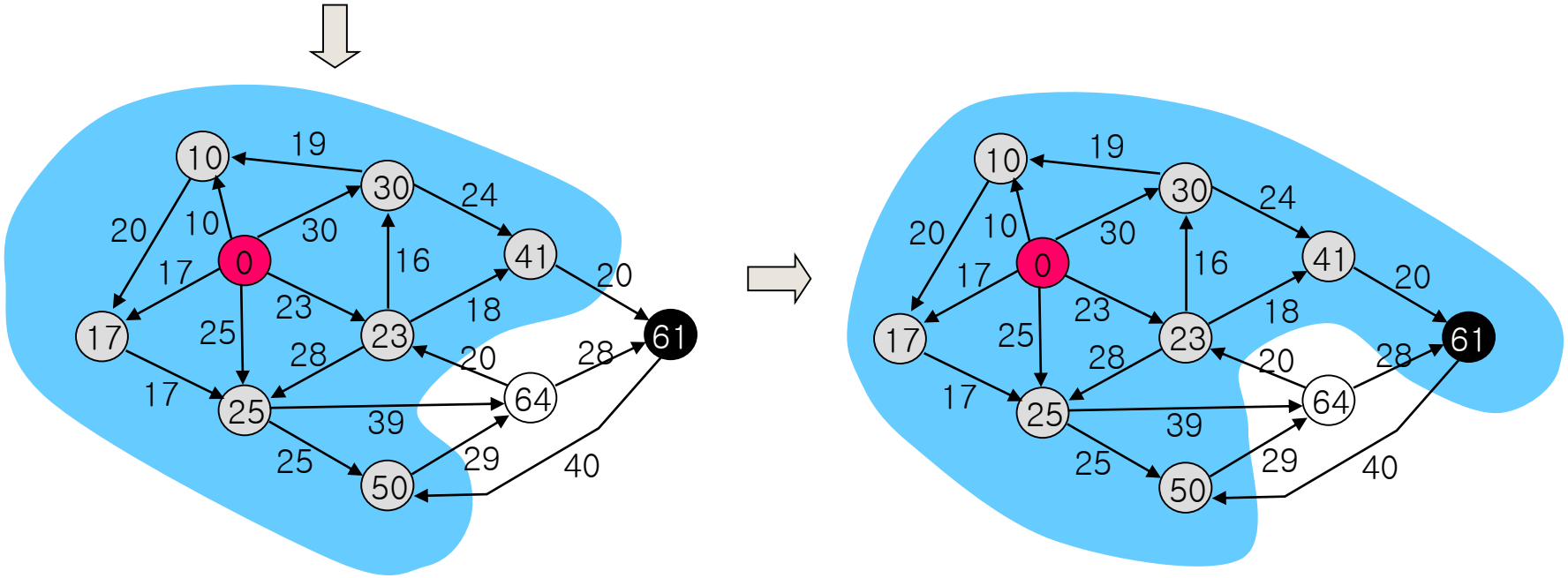
Shortest Path 문제

- **Remind: Dijkstra algorithm**
  - 시작점은 하나
  - 시작점으로부터 다른 모든 **vertex**에 이르는 최단경로를 구한다 (목적점이 하나가 아니다)
- **A* algorithm**에서는 목적점이 하나다

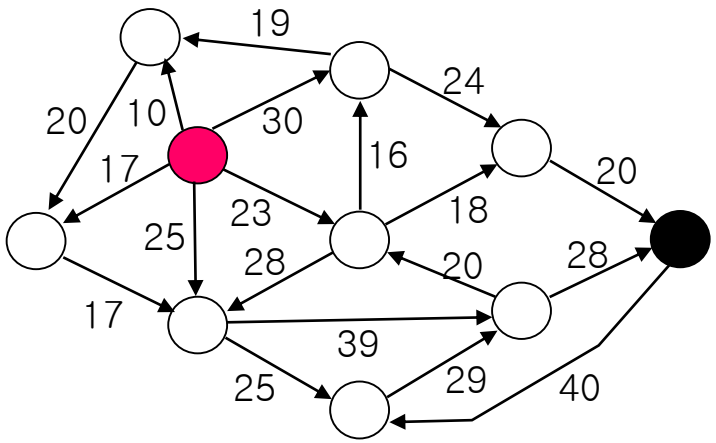
Dijkstra Algorithm의 작동 예

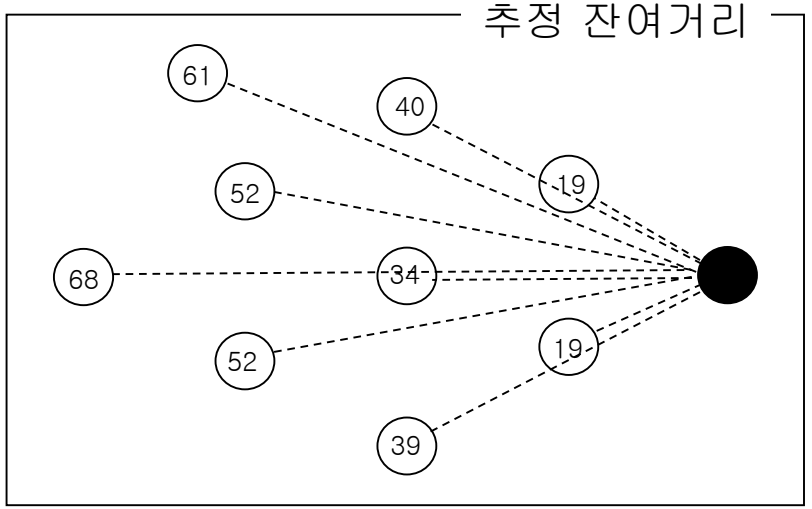


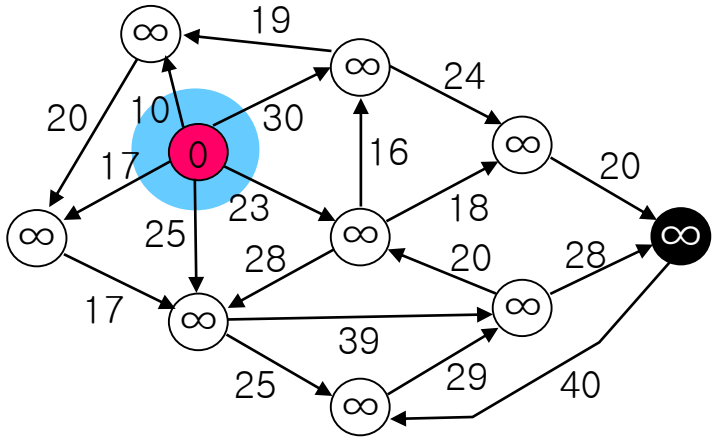

10
19
30
24
20
10
30
0
16
∞
20
17
23
18
17
25
28
23
20
28
17
∞
25
39
25
∞
29
40
10
19
30
24
20
10
30
0
16
41
20
17
23
18
17
25
28
23
20
28
17
∞
25
39
25
∞
29
40
10
19
30
24
20
10
30
0
16
41
20
17
23
18
17
25
28
23
20
28
61
17
25
64
25
39
50
29
40

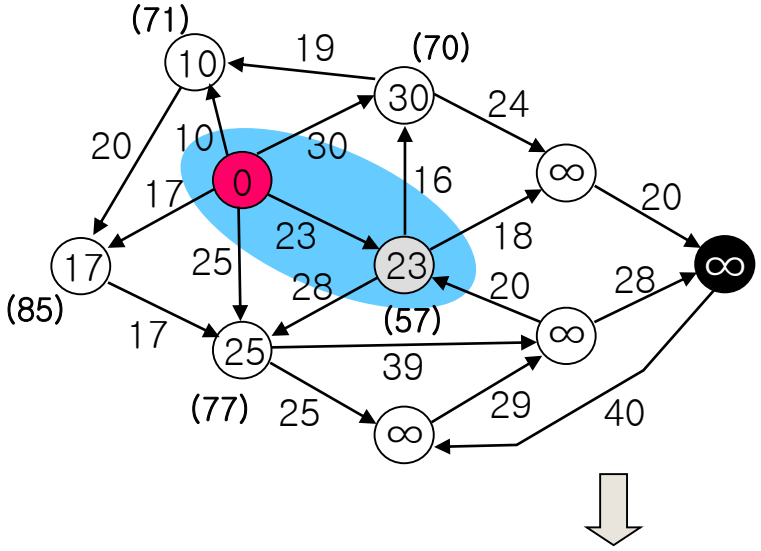

10
19
30
24
20
10
30
41
0
16
20
17
23
18
61
17
25
28
23
20
28
64
17
25
39
25
29
40
50

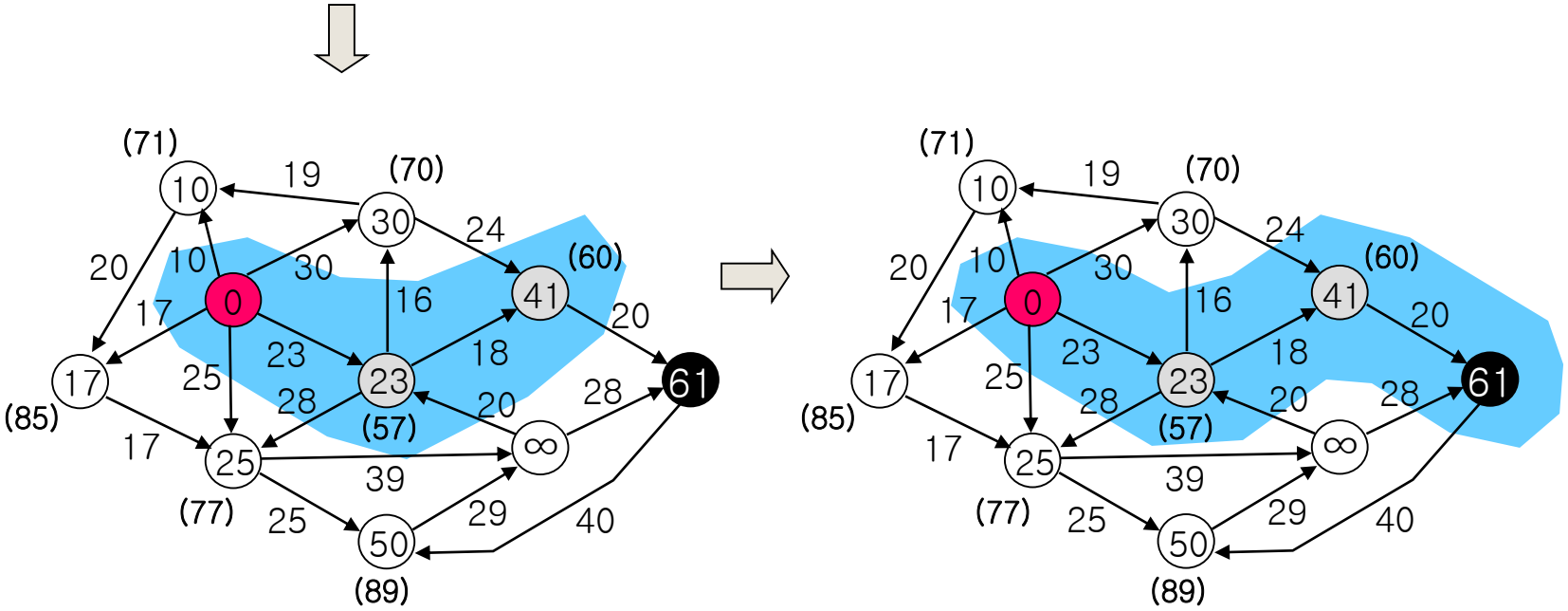
A* Algorithm의 작동 예





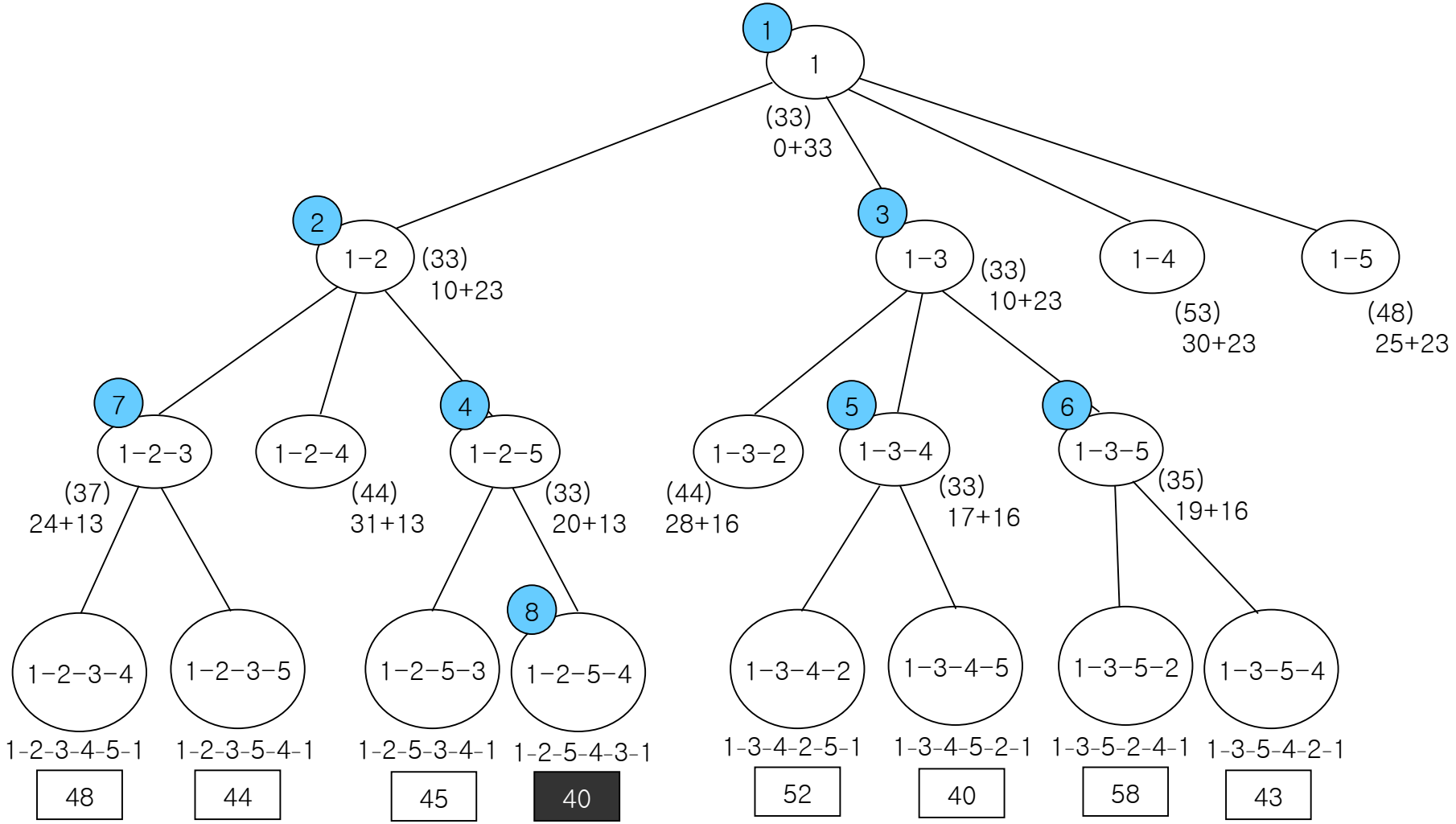






✓ 추정잔여거리를 사용함으로써 탐색의 단계가 현저히 줄었다

TSP 예제를 대상으로 한 A* Algorithm 탐색의 예 (State-Space Tree)



A* Algorithm이 첫 Leaf Node를 방문하는 순간 종료되는 이유

영역과 영역의 **leaf node**들이 모두 40 보다 커질 수 없는 이유를 이해할 것