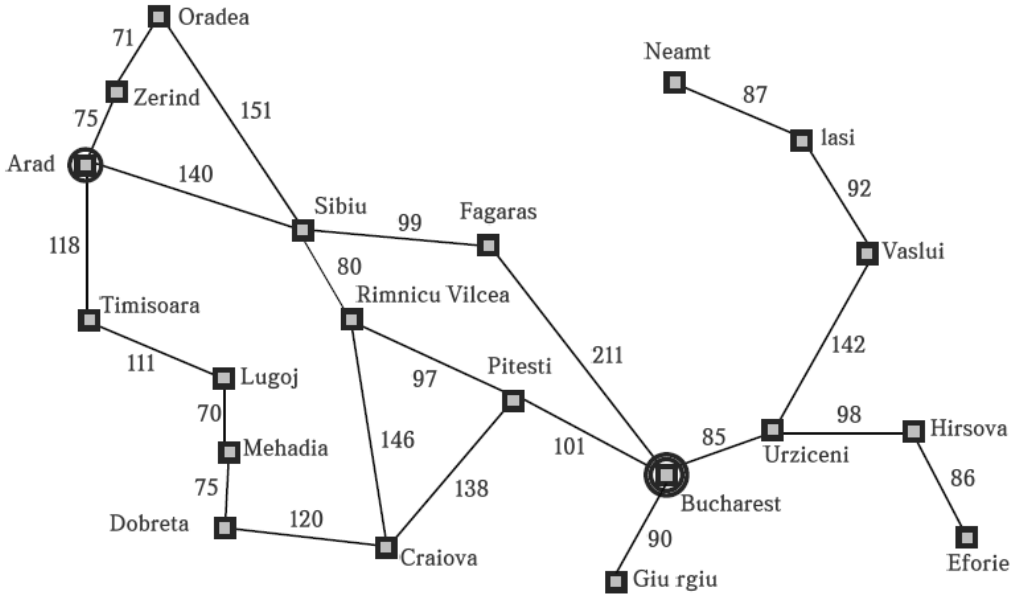
휴리스틱 탐색

휴리스틱

- 문제정의 외에 다른 추가적인 정보가 있을 때 어떻게 그것을 활용하여 탐색을 더 용이하게 하는지에 대한 부가적인 정보
- 노드를 펼쳐놓고 그 노드에 대해서 깊이 탐색을 하기 전에, 그 노드가 얼마나 우리 문제를 푸는데 있어서 좋은 노드인지, 바람직한 노드인지 판단해줄 수 있는 평가함수 정의
- 평가함수에 따라서 평가함수 값이 높은 노드부터 일단 더 깊이 펼치고 더 탐색을 해 나가자.
- 정의하기 어렵지만, 빨리 정확하게 최적값을 찾을 수 있다.(trade-off)

Greedy best-first search



Straight-line distance to Bucharest

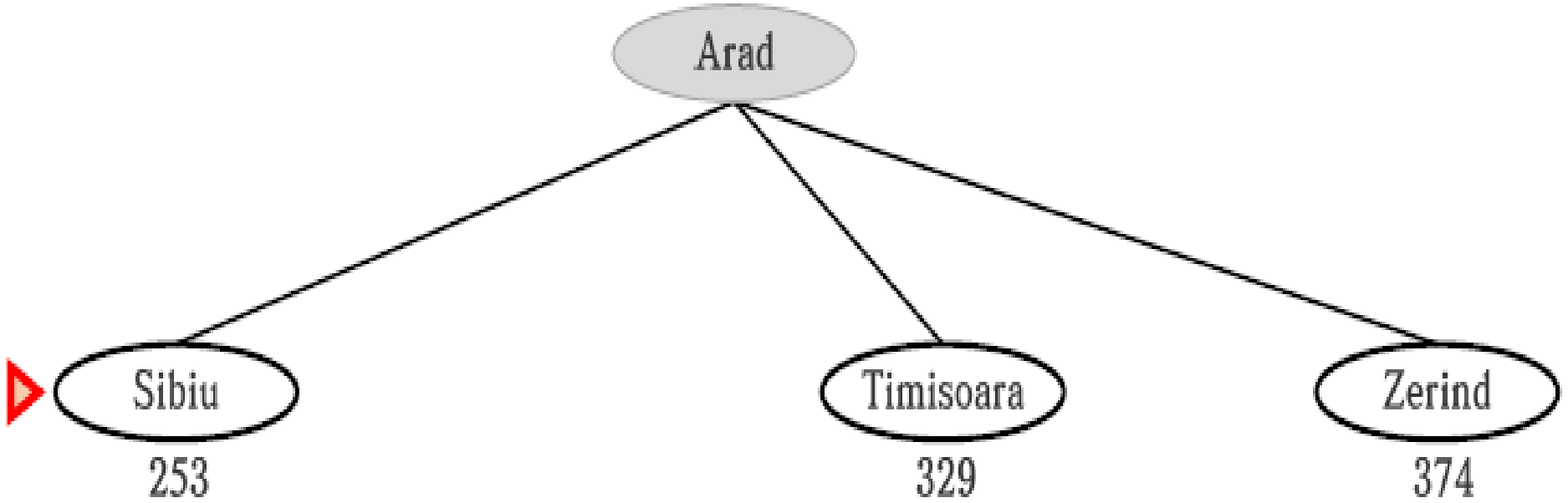
Arad	366	Fagaras	176	Mehadia	241	Sibiu	253
Bucharest	0	Giurgiu	77	Neamt	234	Timisoara	329
Craiova	160	Hirsova	151	Oradea	380	Urziceni	80
Dobreta	242	Iasi	226	Pitesti	10	Vaslui	199
Eforie	161	Lugoj	244	Rimnicu Vilcea	193	Zerind	374

추가정보 : 각 도시에서 Bucharest까지의 직선거리

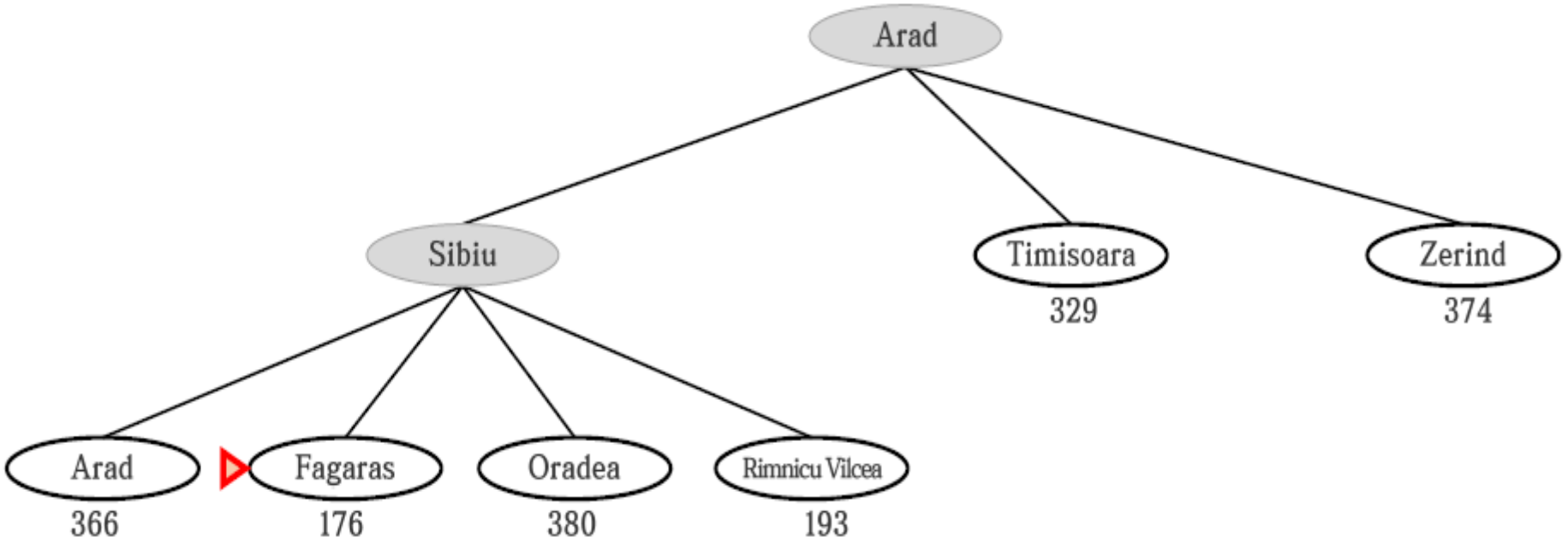
Evaluation function :
Heuristic function, 직선거리

F(n) = h(n)
각 노드에서 최단거리를 따라 목적지에 도달한다.

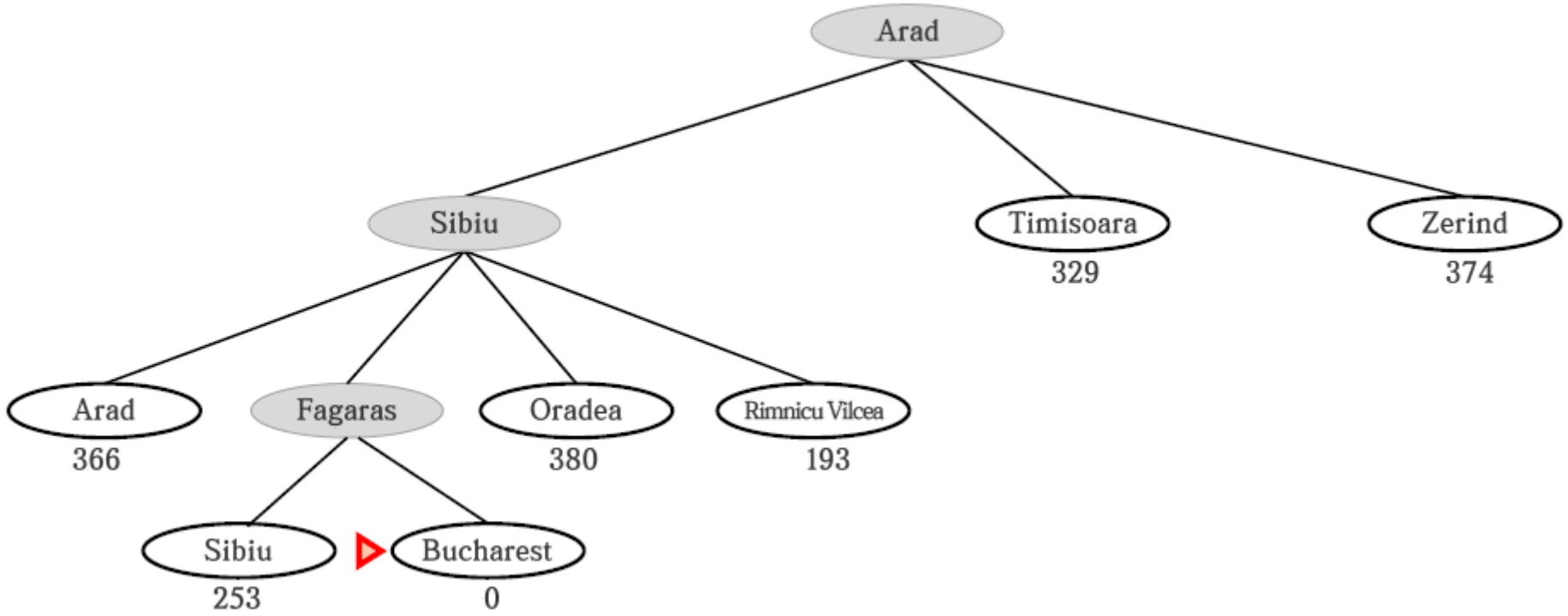
Greedy best-first search example



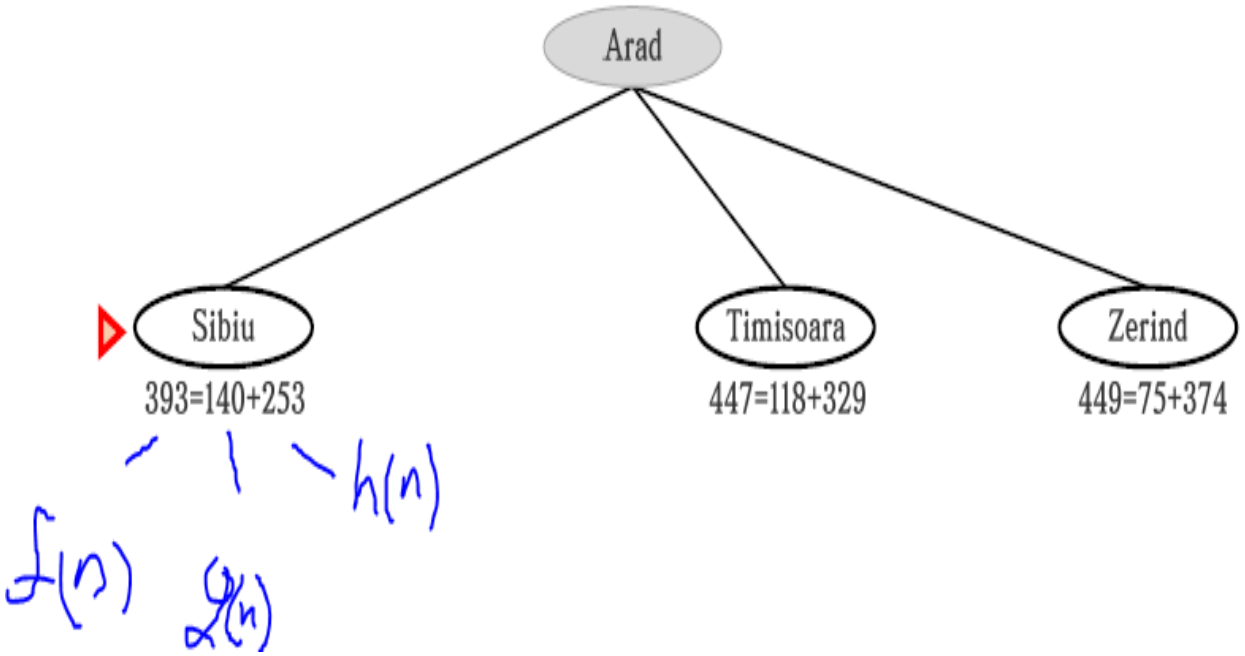
Greedy best-first search example



Greedy best-first search example

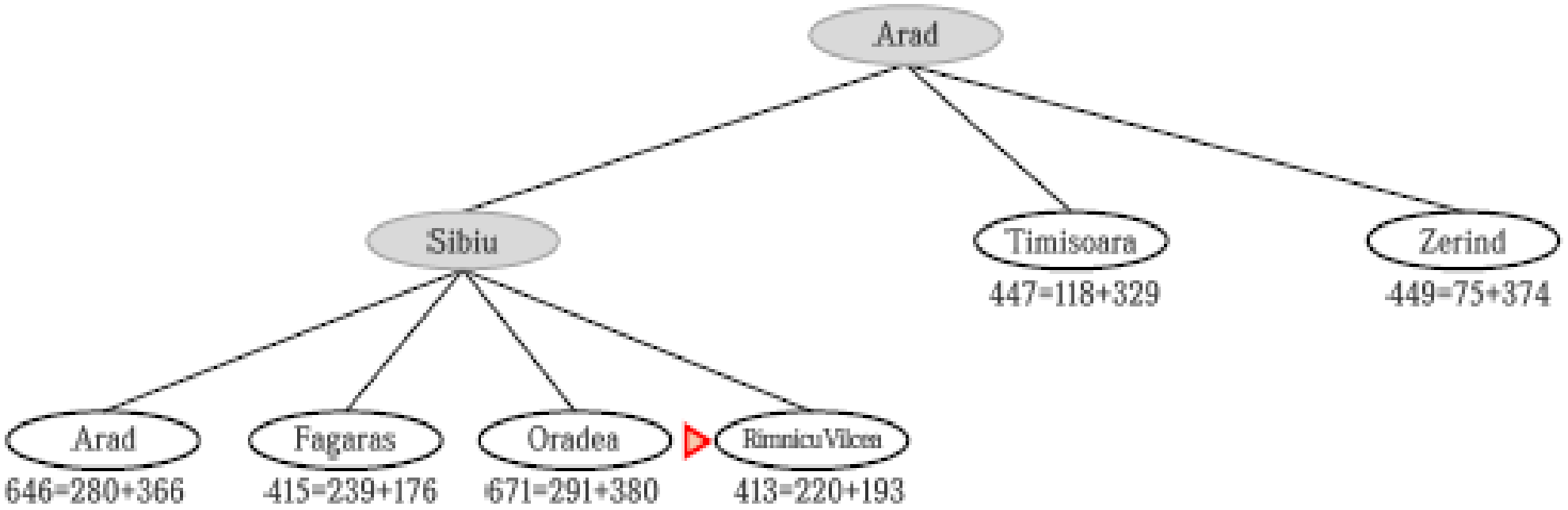


A* search example

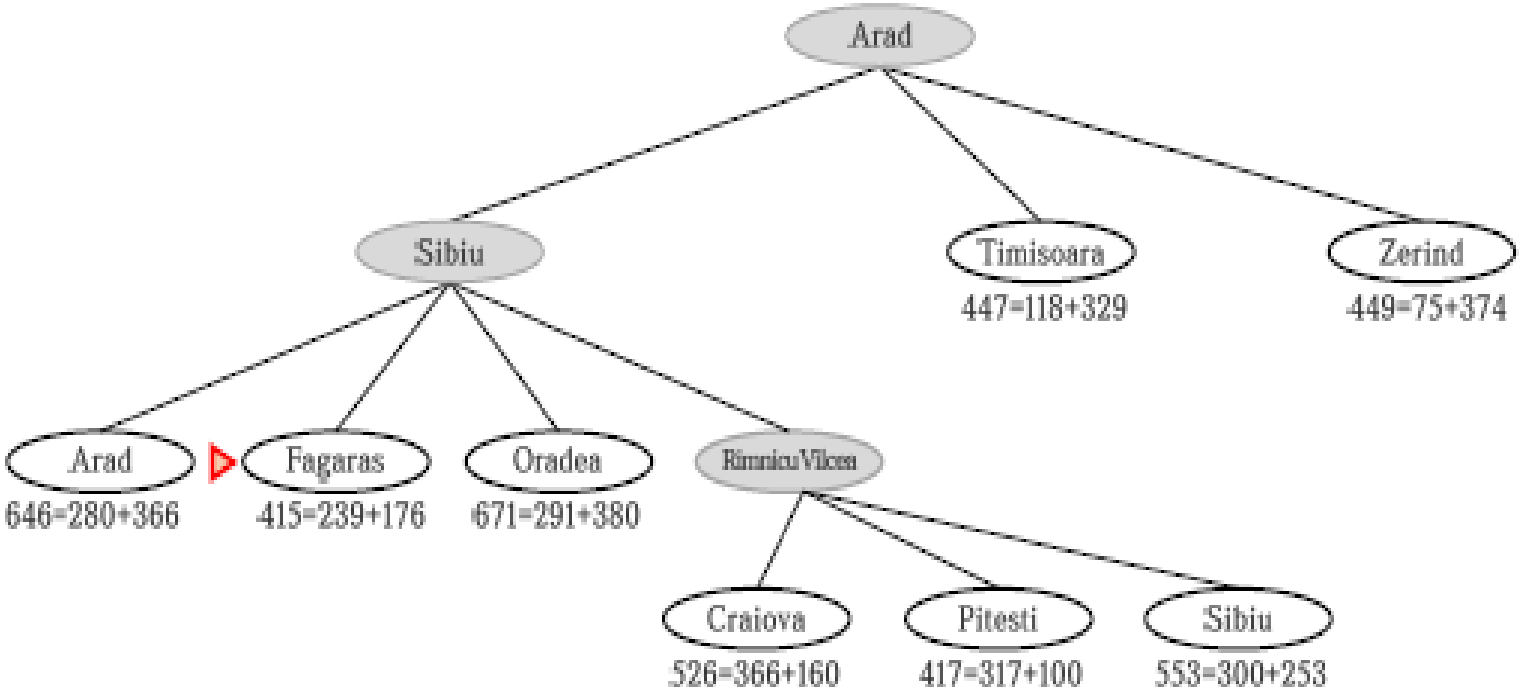


F(n) = h(n) + g(n)
g(n) : 그 도시까지 가는데 걸린 시간, cost.
h(n) : 그 노드에서 goal까지 가는 최단거리

A* search example

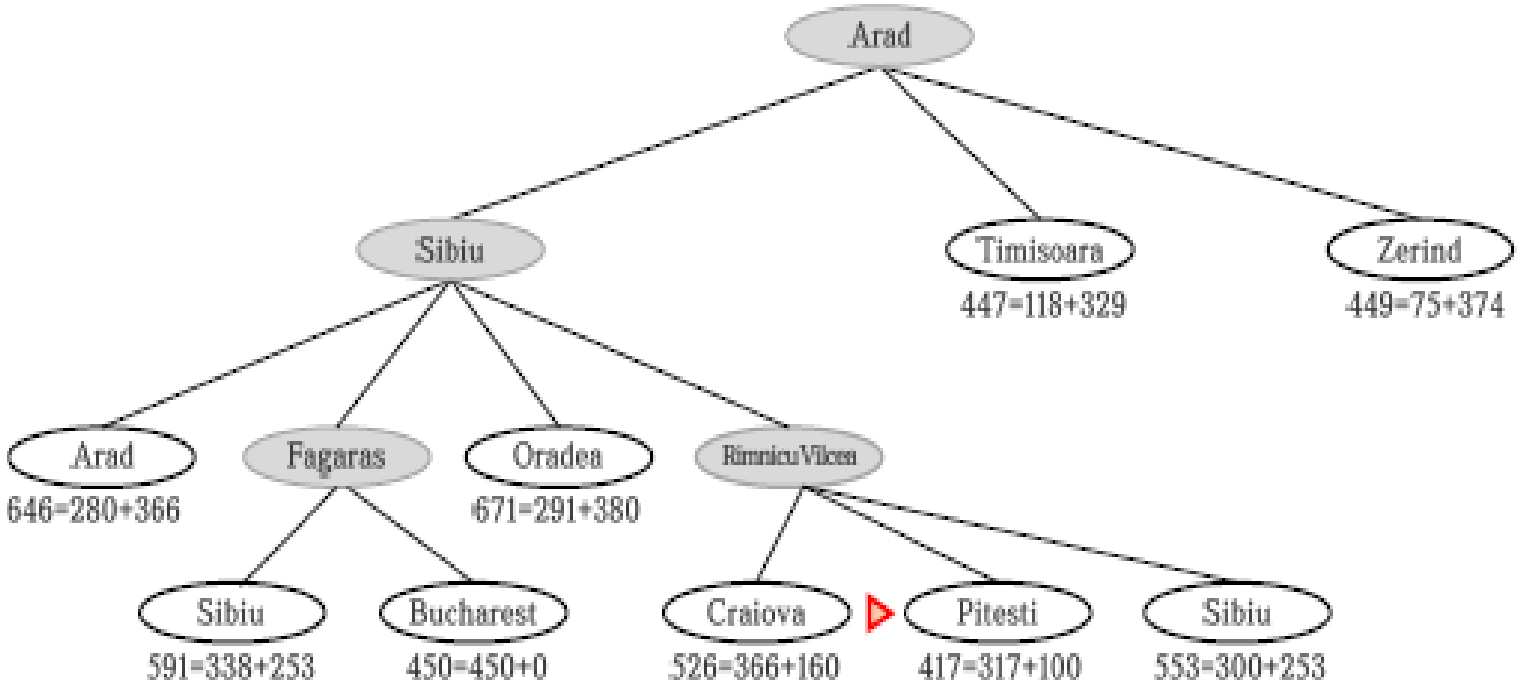


A* search example



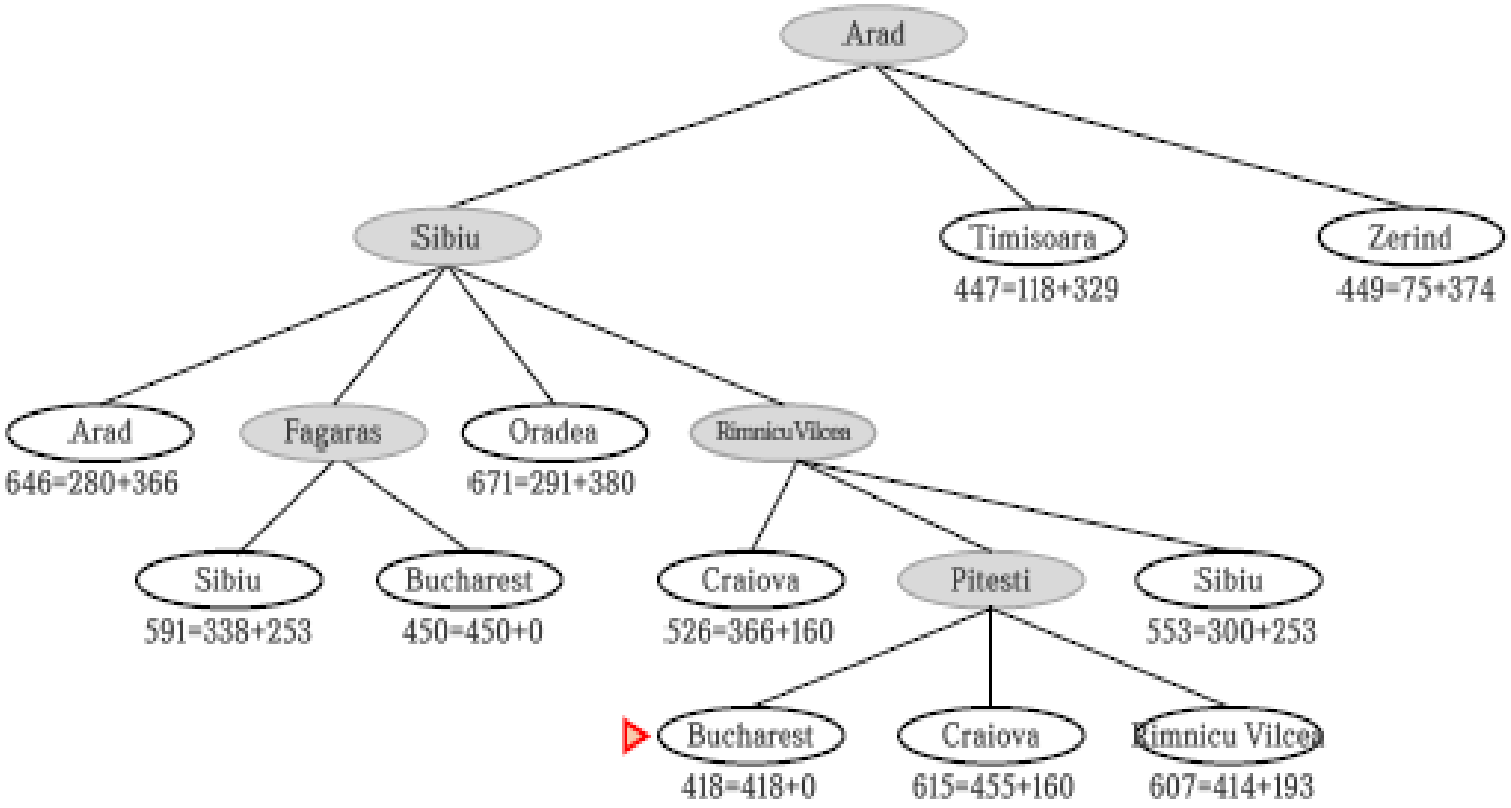
Vilcea를 선택한 뒤, F(n)값 중 가장 작은 값을 선택한다.
->Fagaras

A* search example



Fagaras를 선택한 뒤, F(n)값 중 가장 작은 값을 선택한다.
->Pitesti

A* search example



Pitesti를 선택한 뒤, F(n)값 중 가장 작은 값을 선택한다.
->Bucharest

Admissible 조건

- 우리가 사용하는 휴리스틱 함수 값이 실제 그 노드에서 goal까지의 true cost보다 항상 작거나 같은 경우를 의미함.
- 실제 값보다 항상 작거나 같게 예측을 하는 경우.
- 아주 객관적인 휴리스틱을 설정해서 누구나 인정할 수 있어야 한다.
- 앞의 예에서 어떤 도시와 goal까지의 직선거리를 활용함.
- 직선 거리는 실제 거리보다 항상 작다.(실제 거리가 항상 길음, 정해진 도로로 가야 하기 때문에)
- 이 경우, **A*** 알고리즘으로 언제나 최적 솔루션을 찾을 수 있다.