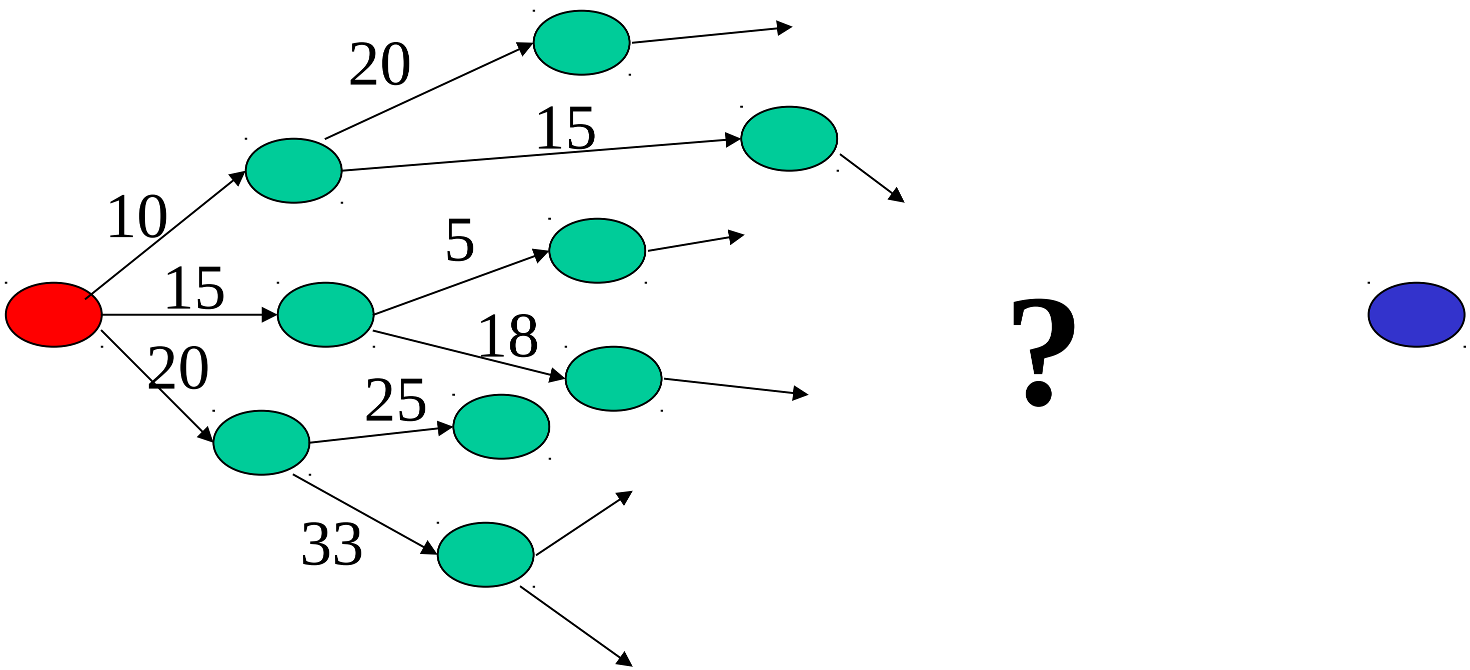
The A* Algorithm

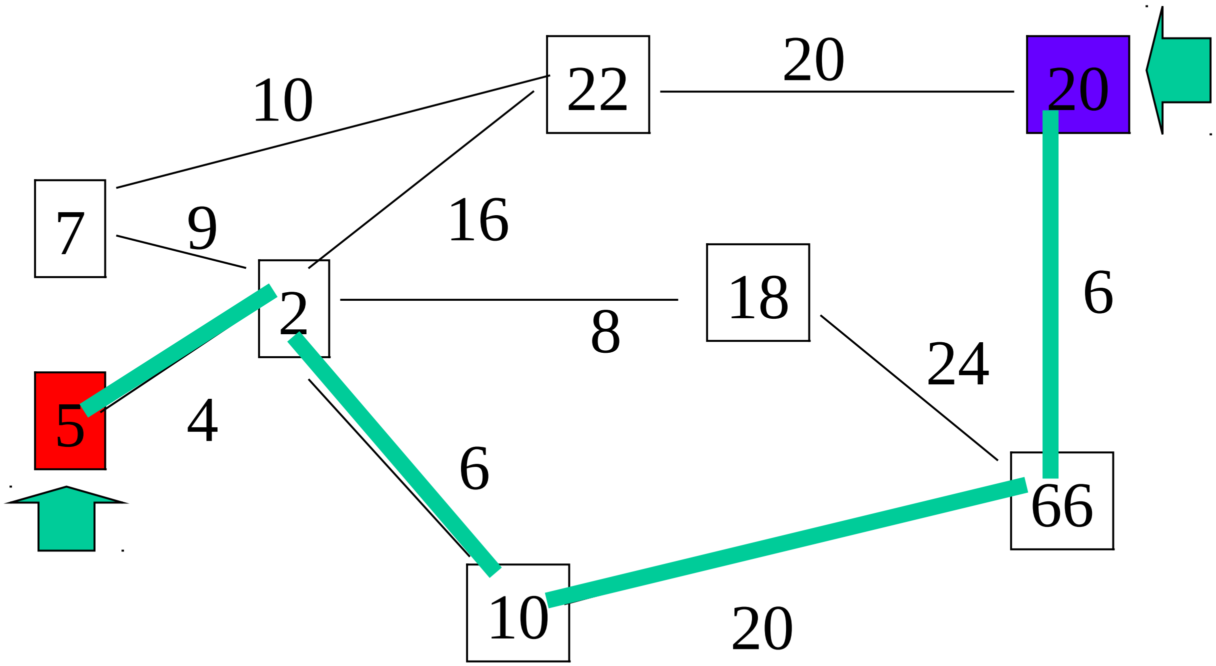
The Search Problem

Starting from a node n find the shortest path to a goal node g



Shortest Path

Consider the following weighted undirected graph:

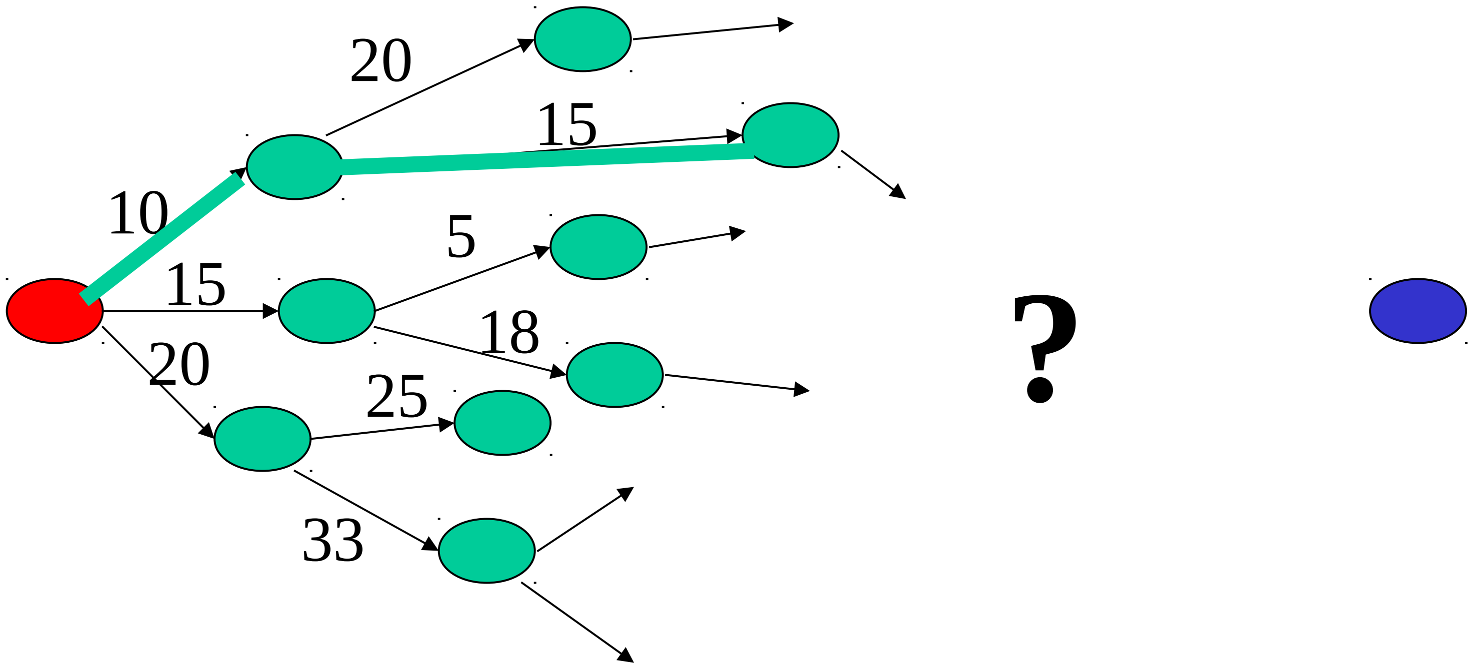


We want: A path 5 → v1 → v2 → … → 20

Such that g(20) = cost(5→v1) + cost(v1→ v2) + … + cost (→20) is minimum

Djikstra Algorithm

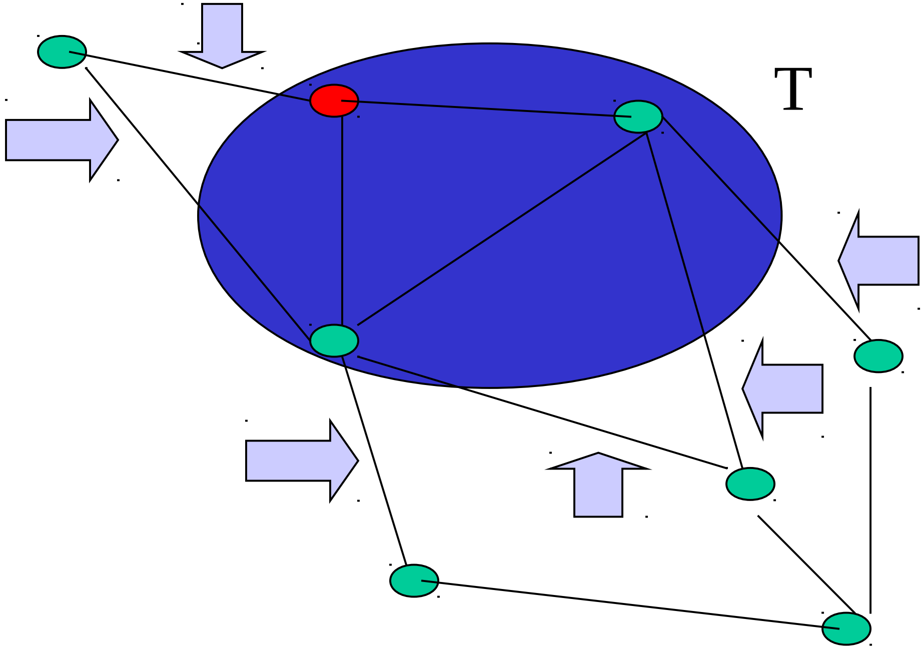
Greedy algorithm: from the candidate nodes select the one that has a path with minimum cost from the starting node

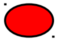


Djikstra Algorithm

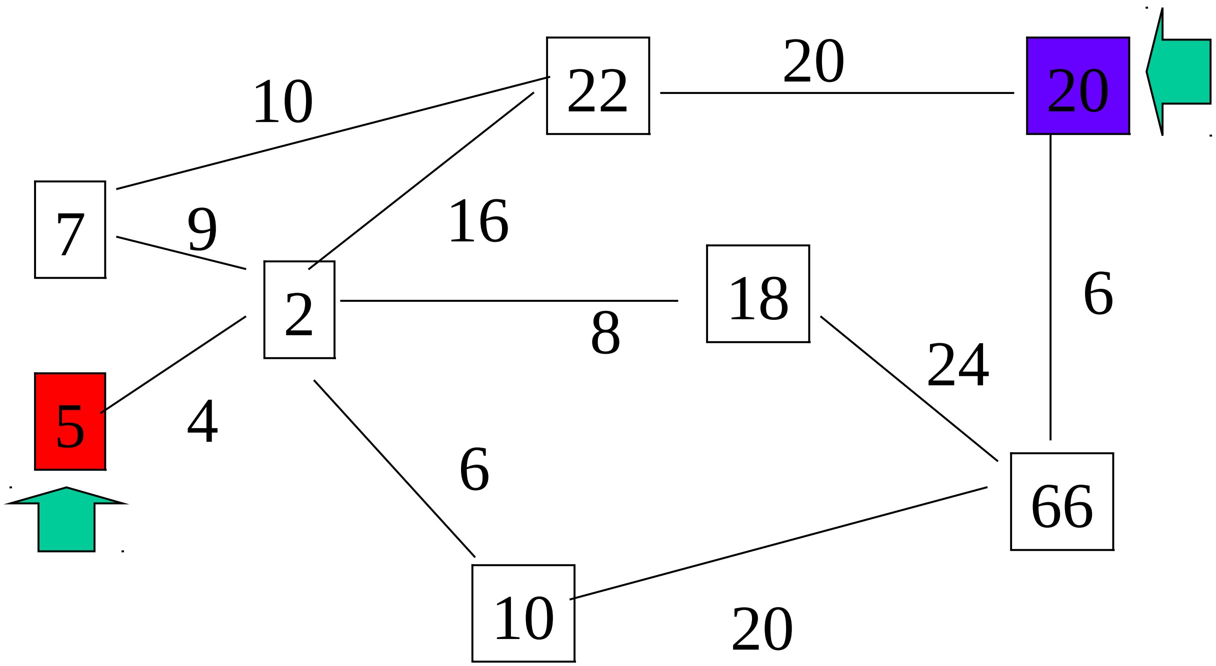
Given a Graph G = (V,E) and T a subset of V, the fringe of T, is defined as:

$$\text{Fringe}(T) = \{ (w,x) \text{ in } E : w \in T \text{ and } x \in V - T\}$$



Djikstra's algorithm pick the edge v in Fringe(T) that has minimum distance to the starting node  g(v) is minimum

Example

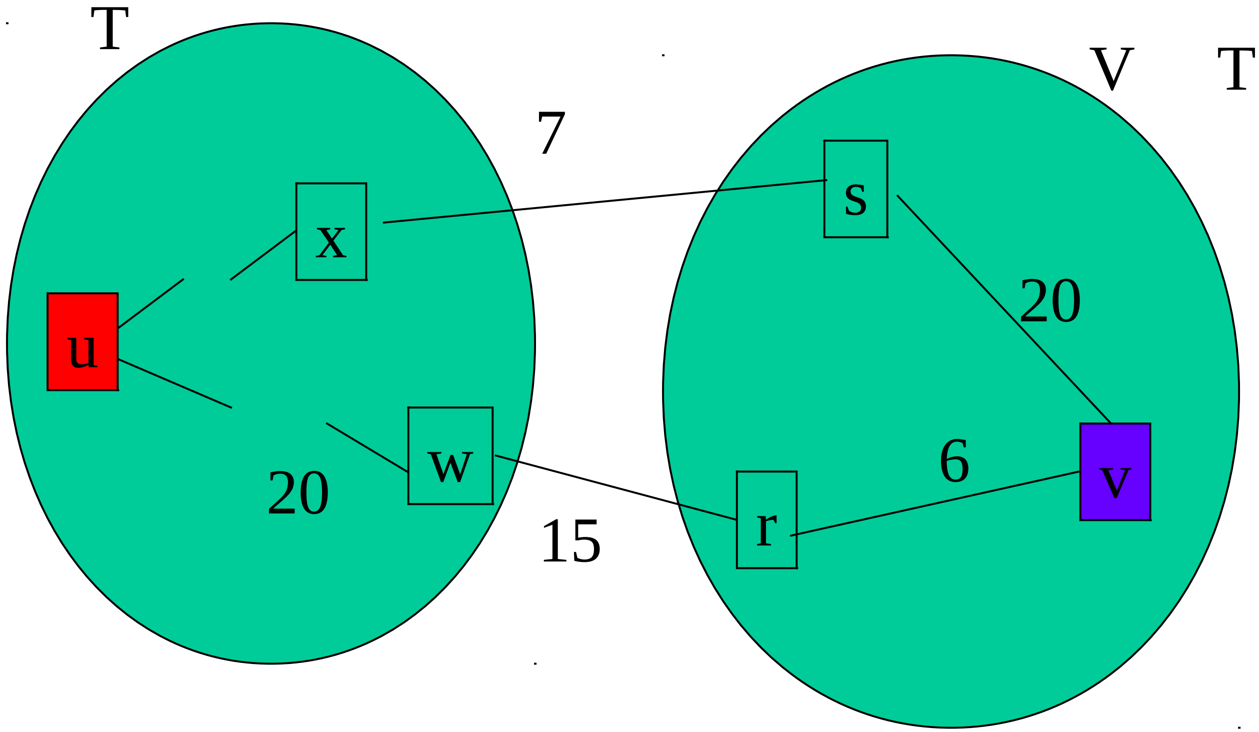


Properties

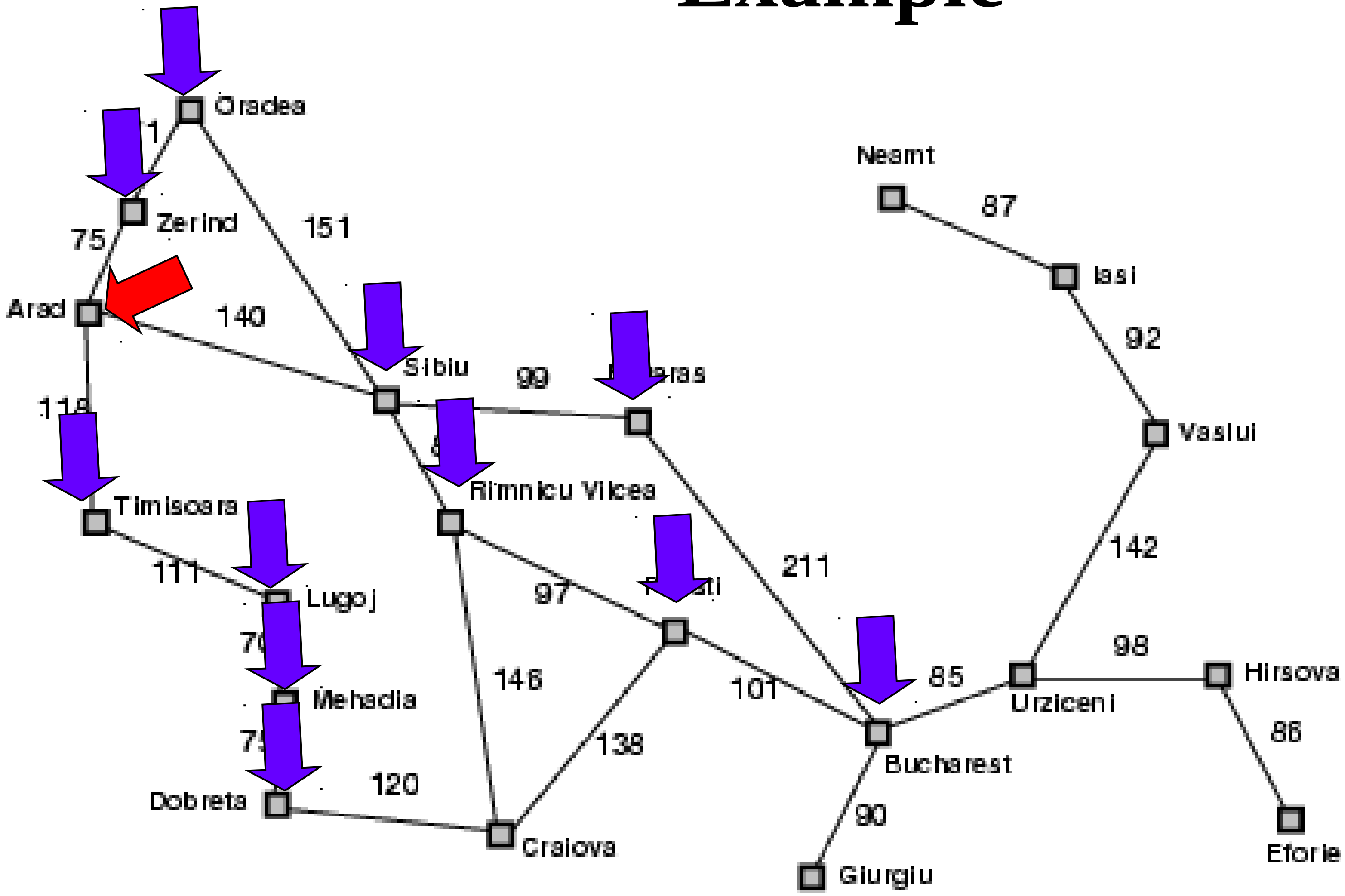
Dijkstra's is a greedy algorithm

Why Dijkstra's Algorithm works?

The path from u to every node in T is the minimum path



Example



Straight–line distance to Bucharest

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

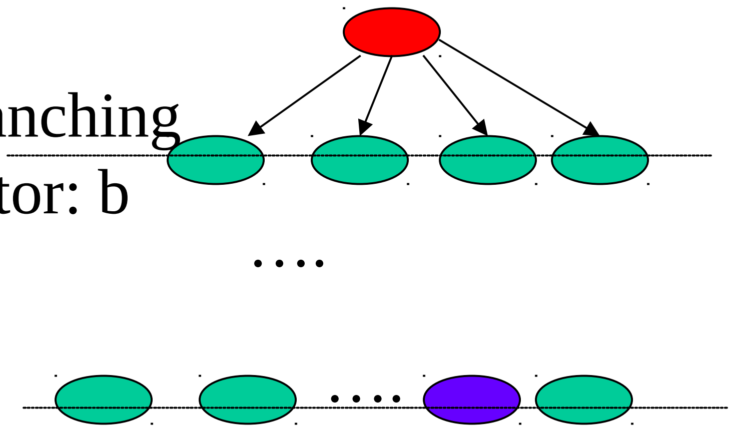
What does Djikstra’s algorithm will do? (minimizing g(n))

Problem: Visit too many nodes, some ***clearly*** out of the question

Complexity

- Actual complexity is $O(|E|\log_2 |E|)$
- Is this good?
  Actually it is bad for very large graphs!

Branching factor: b



nodes = $b^{(\# \text{ levels})}$

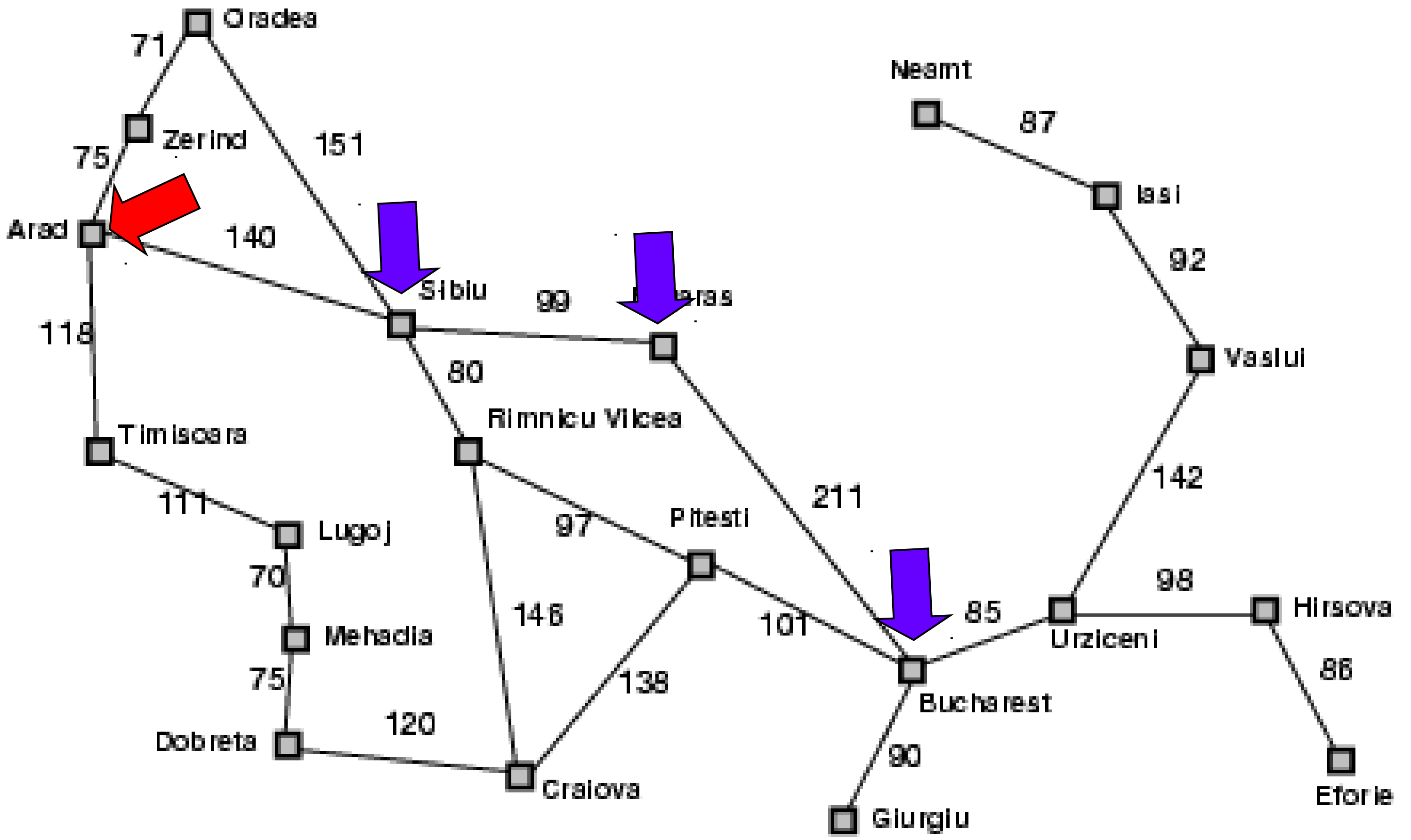
Another Example: think of the search space in chess

Better Solution: Make a 'hunch"!

- Use ***heuristics*** to guide the search
  - **Heuristic**: estimation or "hunch" of how to search for a solution
- We define a heuristic function:

  h(n) = "estimate of the cost of the cheapest path from the starting node to the goal node"

Lets Try A Heuristic



Straight–line distance to Bucharest

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

Heuristic: minimize h(n) = “Euclidean distance to destination”

Problem: not optimal (through Rimmici Viicea and Pitesti is shorter)

The A* Search

- **Difficulty**: we want to still be able to generate the path with minimum cost

- A* is an algorithm that:
  - Uses heuristic to guide search
  - While ensuring that it will compute a path with minimum cost

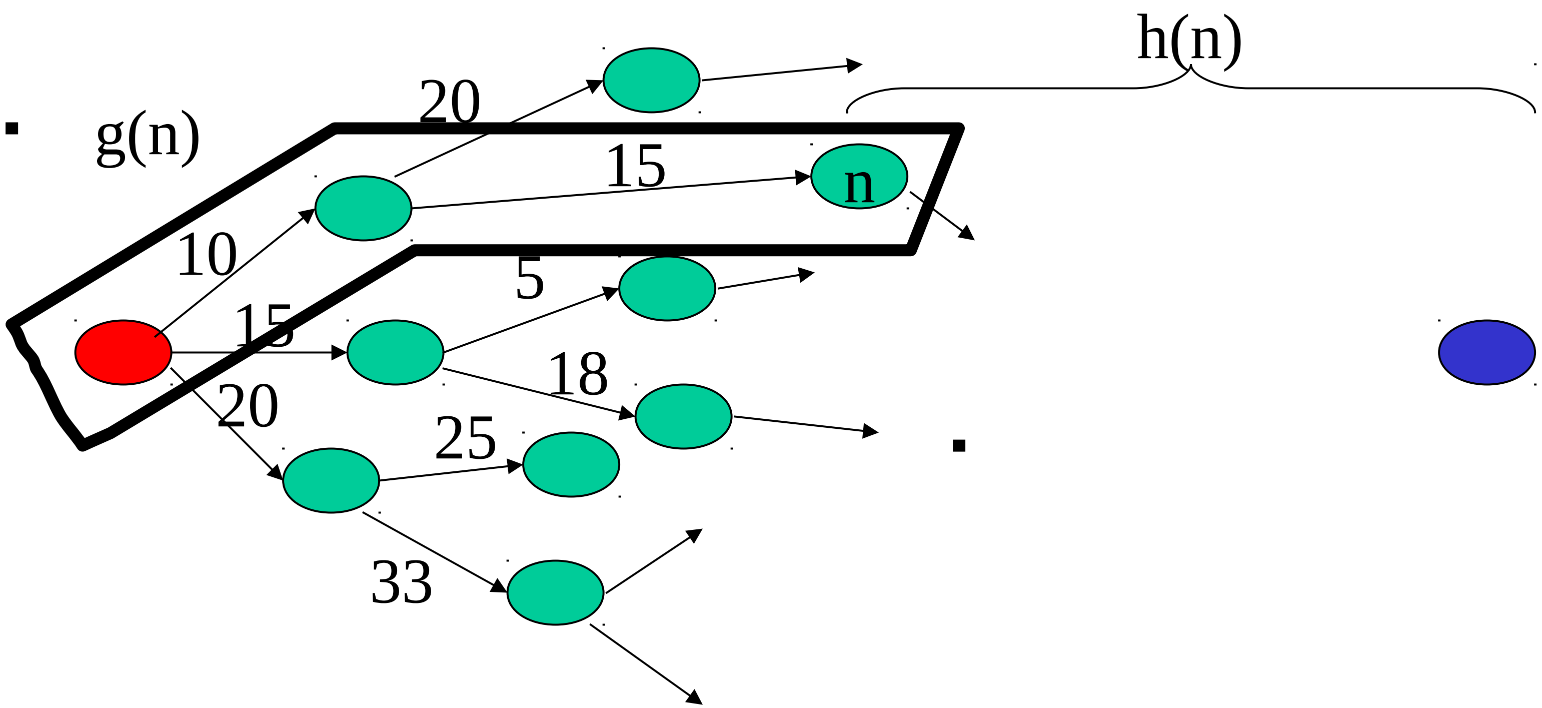
“estimated cost”

- A* computes the function f(n) = g(n) + h(n)

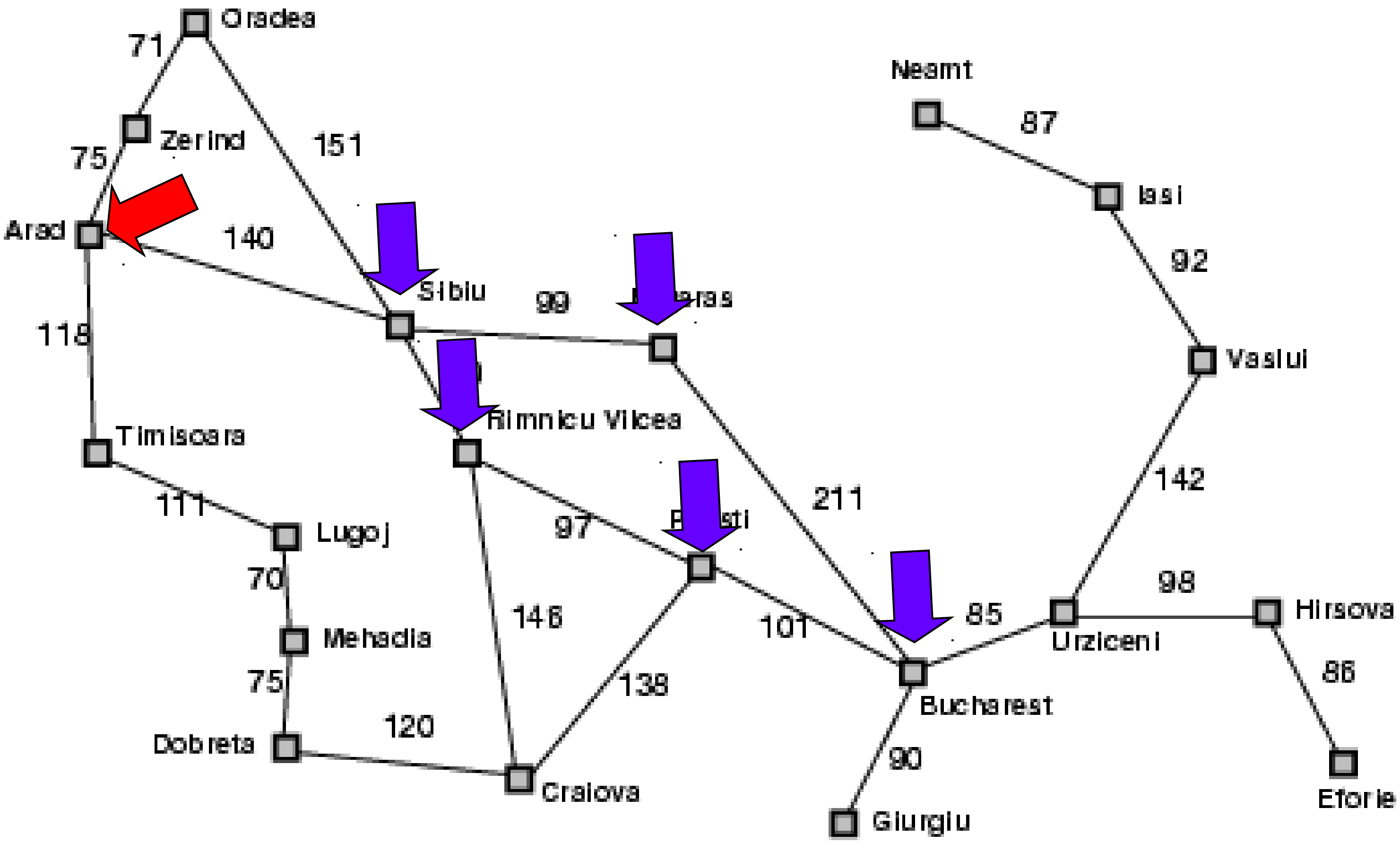
“actual cost”

A*

- f(n) = g(n) + h(n)
  - g(n) = “cost from the starting node to reach n”
  - h(n) = “estimate of the cost of the cheapest path from n to the goal node”



Example



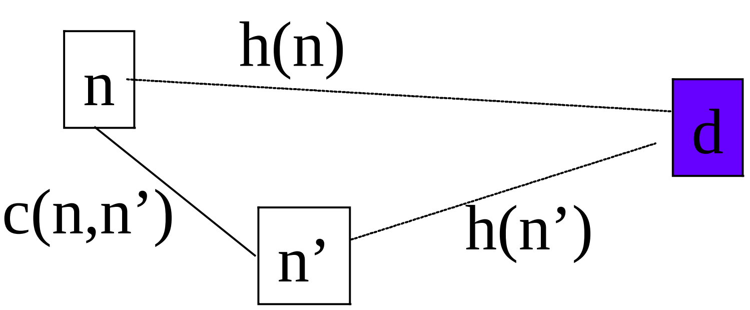
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A*: minimize f(n) = g(n) + h(n)

Properties of A*

- A* generates an optimal solution if h(n) is an admissible heuristic and the search space is a tree:
  - h(n) is **admissible** if it never overestimates the cost to reach the destination node
- A* generates an optimal solution if h(n) is a consistent heuristic and the search space is a graph:
  - h(n) is **consistent** if for every node n and for every successor node n' of n:

$$h(n) \leq c(n,n') + h(n')$$



- If h(n) is consistent then h(n) is admissible
- Frequently when h(n) is admissible, it is also consistent

Admissible Heuristics

- A heuristic is admissible if it is too optimistic, estimating the cost to be smaller than it actually is.

- Example:

  In the road map domain,

  h(n) = "Euclidean distance to destination"

  is admissible as normally cities are not connected by roads that make straight lines

How to Create Admissible Heuristics

- Relax the conditions of the problem
  - This will result in admissible heuristics!

- Lets look at an 8-puzzle game:

  http://www.permadi.com/java/puzzle8/

- Possible heuristics?

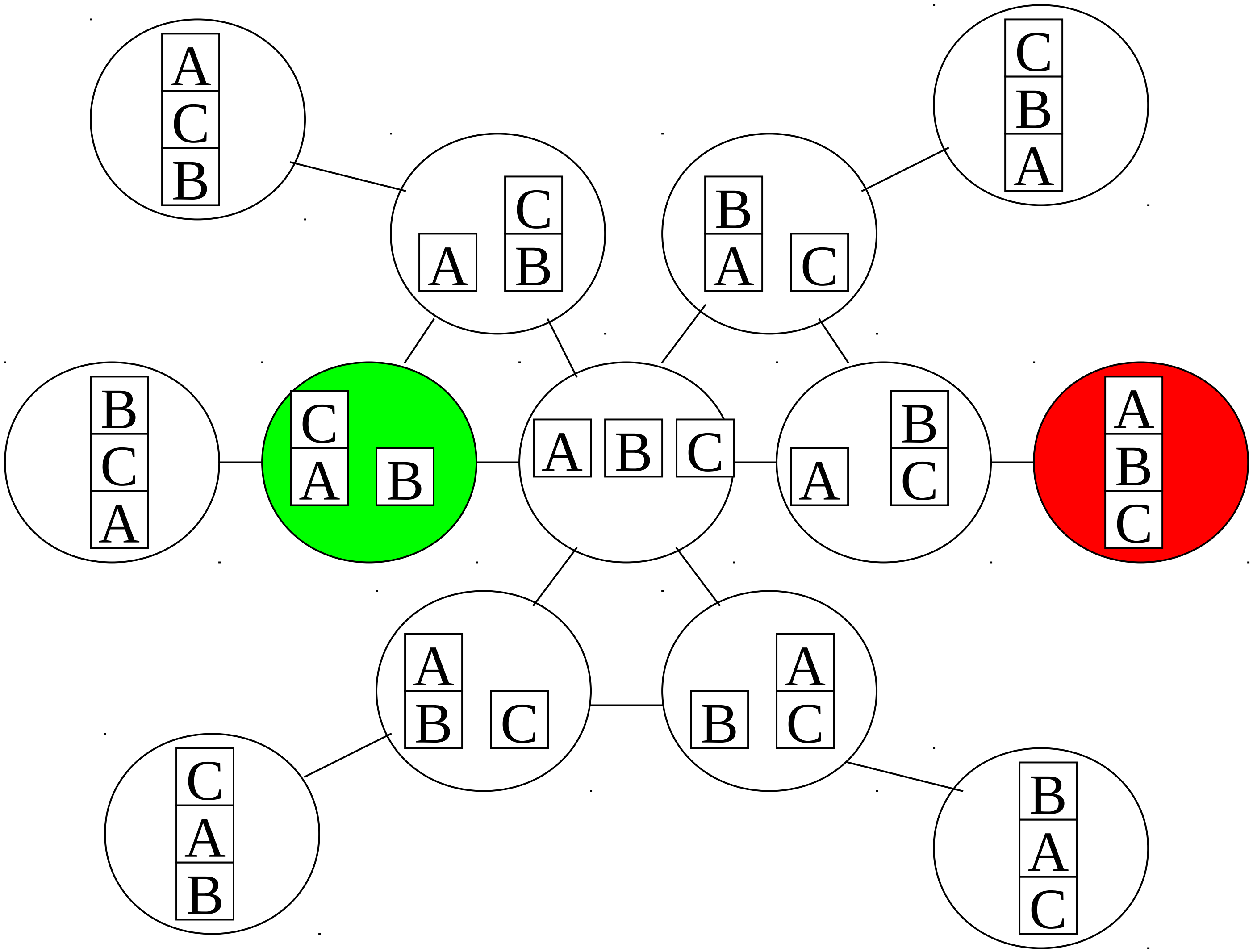
Example: Admissible Heuristics in 8-Puzzle Game

- Heuristic: a tile A can be moved to any tile B
  - H1(n) = “number of misplaced tiles in board n”

- Heuristic: a tile A can be moved to a tile B if B is adjacent to A
  - H2(n) = “sum of distances of misplaced tiles to goal positions in board n”

- Some experimental results reported in Russell & Norvig (2002):
  - A* with h2 performs up to 10 times better than A* with h1
  - A* with h2 performs up to 36,000 times better than a classical uninformed search algorithm (iterative deepening)

Using A* in Planning



h(n) = “# of goals remaining to be satisfied” g(n) = “# of steps so far”

A* in Games

- Path finding
- Improvement
  - We will see that sometimes even A* speed improvements are not sufficient
  - Additional improvements are required

- A* can be used for planning moves computer-controlled player (e.g., chess)