A* (A-Star)

AI+X
INSTITUTE FOR ARTIFICIAL INTELLIGENCE

A* Search

- Combination of uniform cost search and greedy best-first search
- Assume:
  - **g(n)** is the path cost from the initial state to node **n**
  - **h(n)** is the estimated cost of the shortest path from **n** to a **goal state** (heuristic function)
- Expand the leaf node that minimizes:

$$\mathbf{f(n) = g(n) + h(n)}$$

- **f(n)** finds the most promising paths from the initial state to a goal

Traveling from Arad to Bucharest

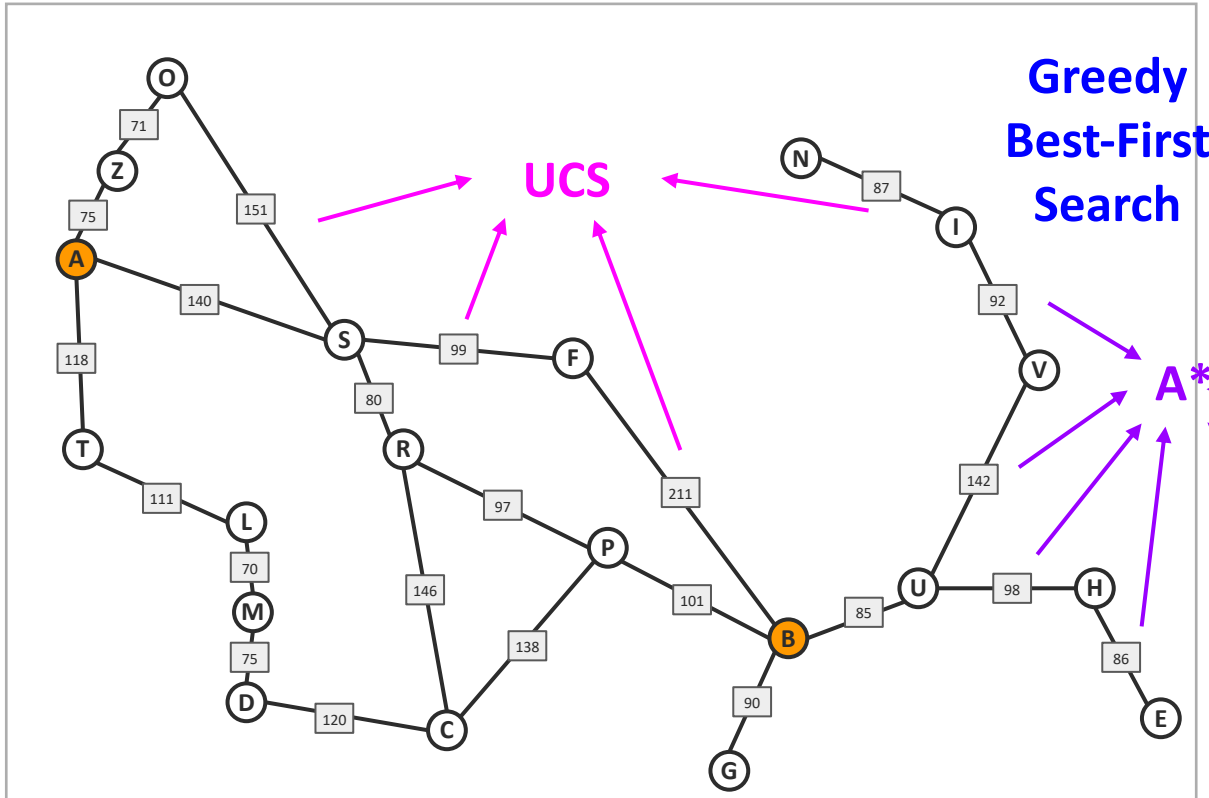

Figure 1. A simplified road map of part of Romania, with road distances in miles.

Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Table 1. Heuristic function: straight-line distances to Bucharest.

A* search

- Strategy:
  - Expand the node with the lowest **f(n)**
  - $\mathbf{f(n) = g(n) + h(n)}$
- Implementation
  - Fringe (set of leaf nodes) is a priority queue
    - Priority is given to the nodes with lowest **f(n)**

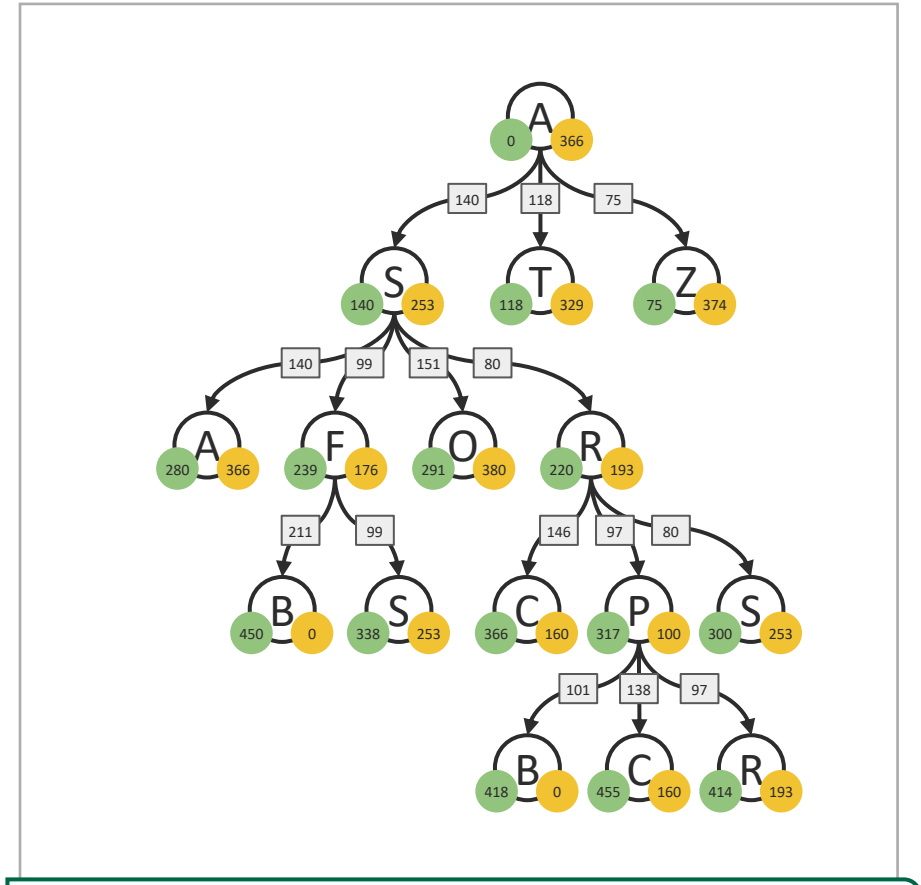

Figure 2. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

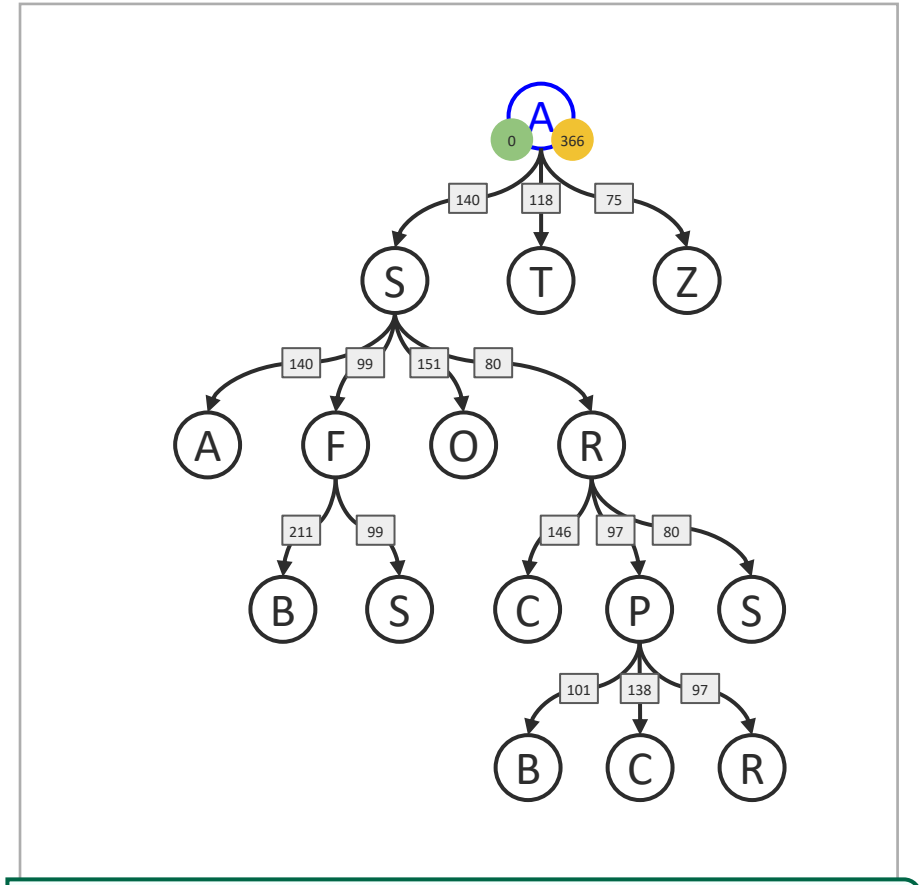

Figure 3. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

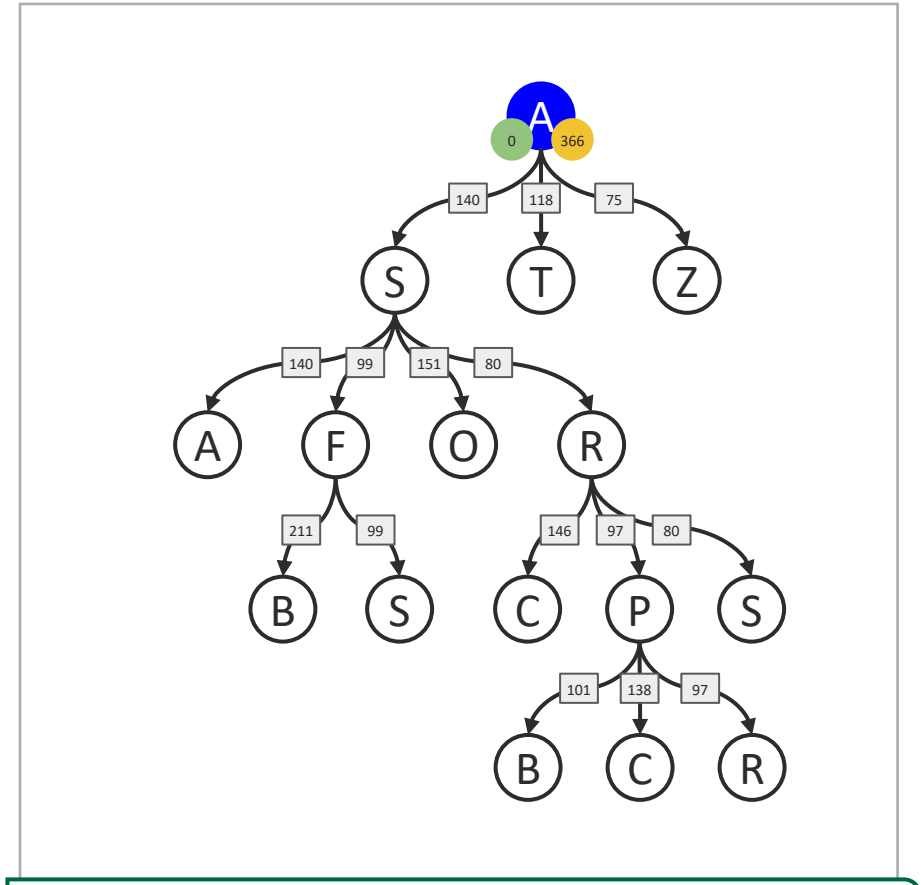

Figure 3. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

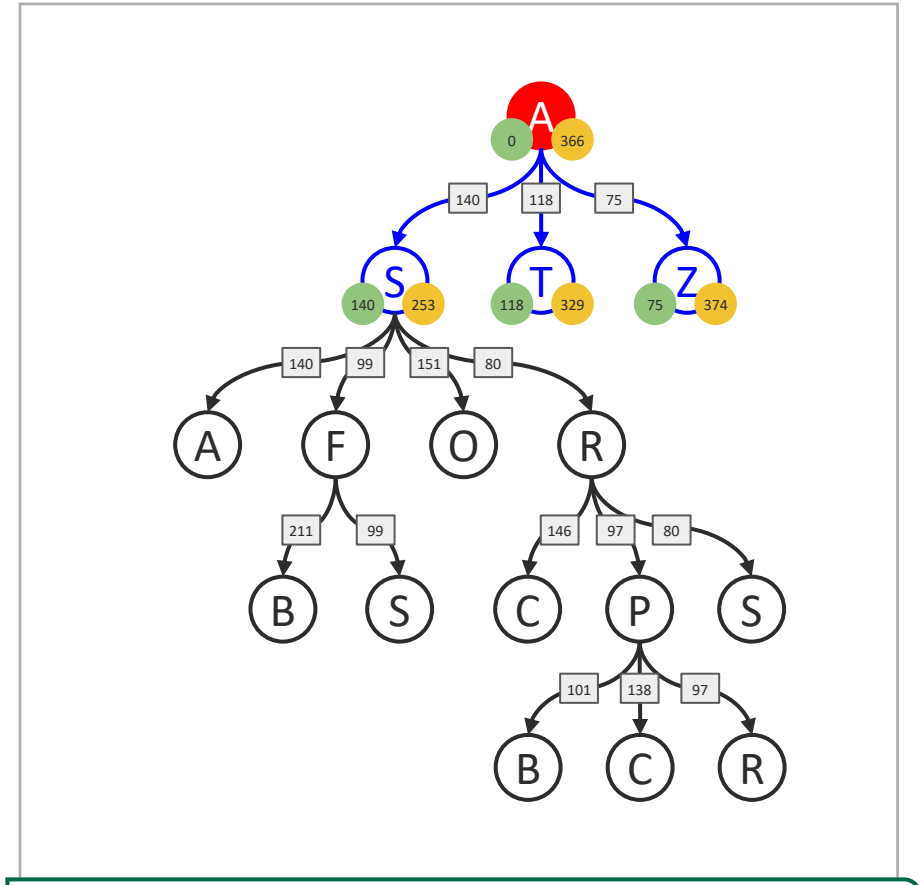

Figure 3. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

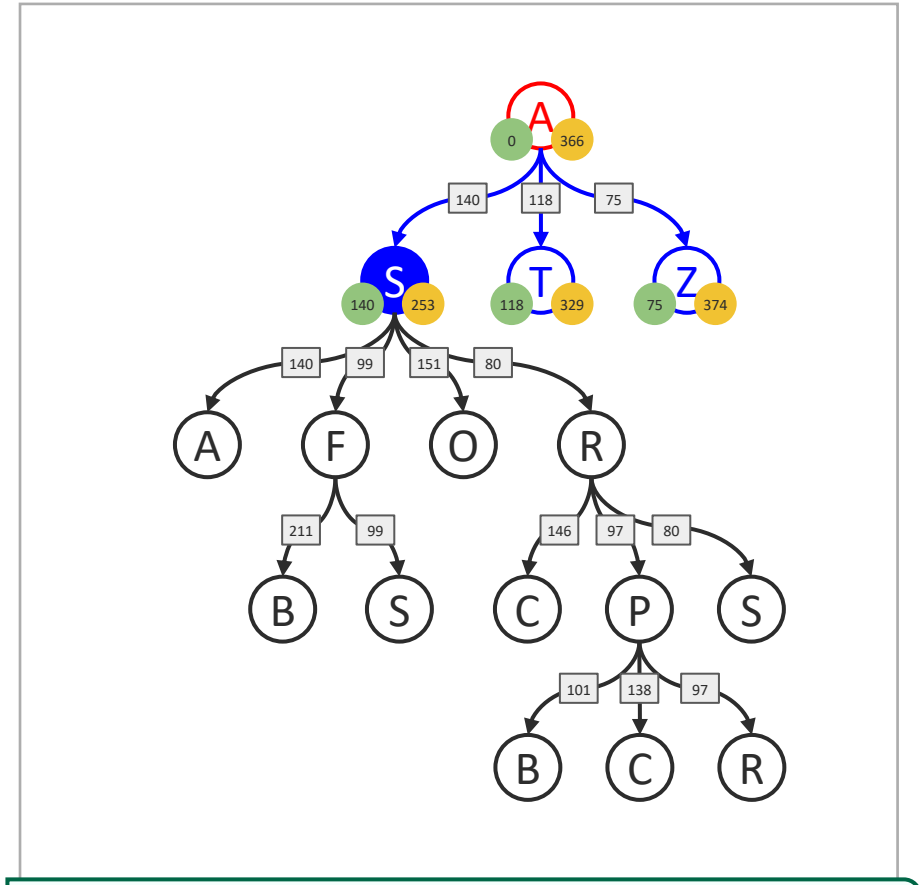

Figure 3. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

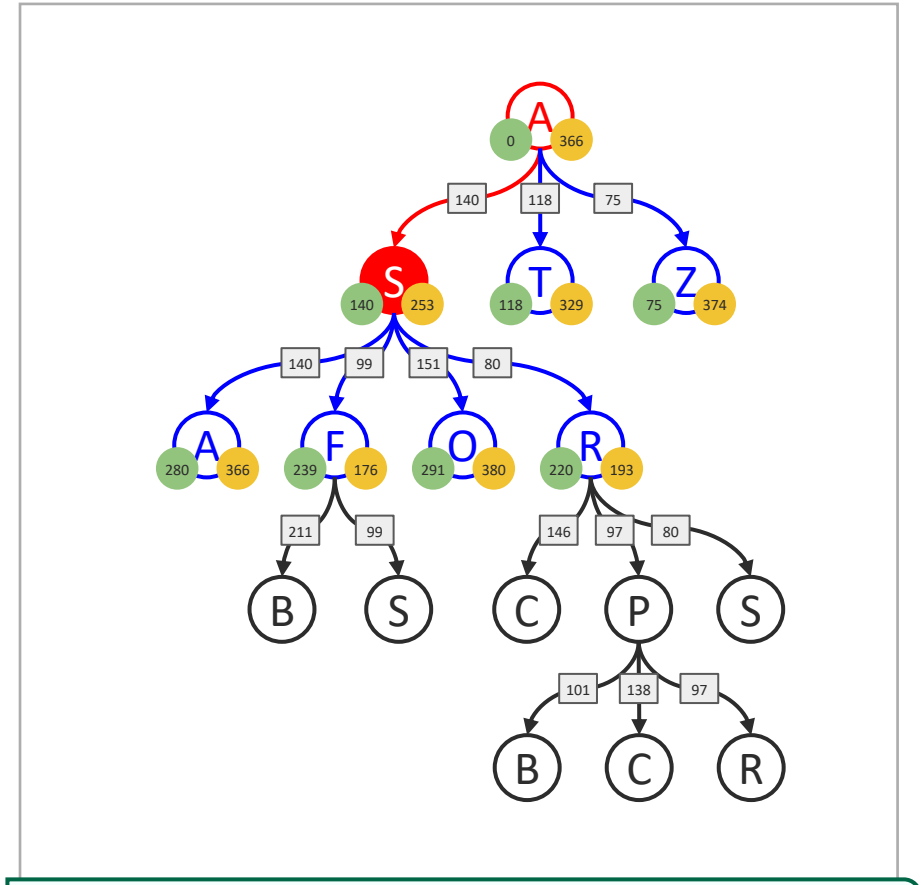

Figure 3. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

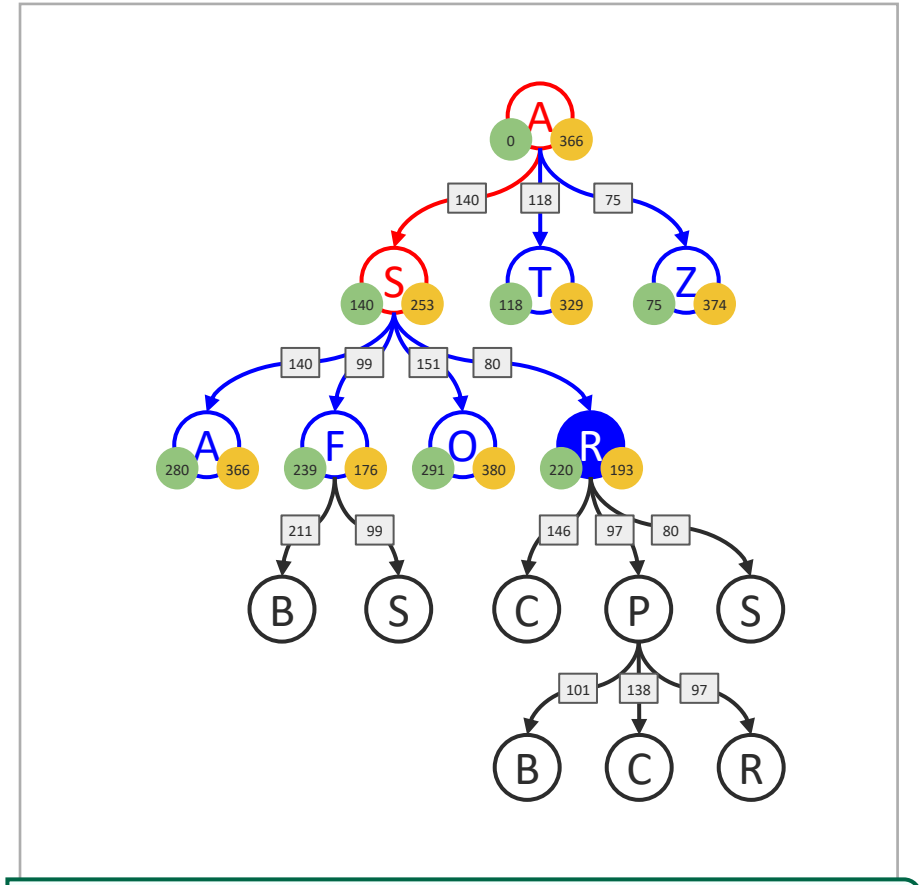

Figure 3. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

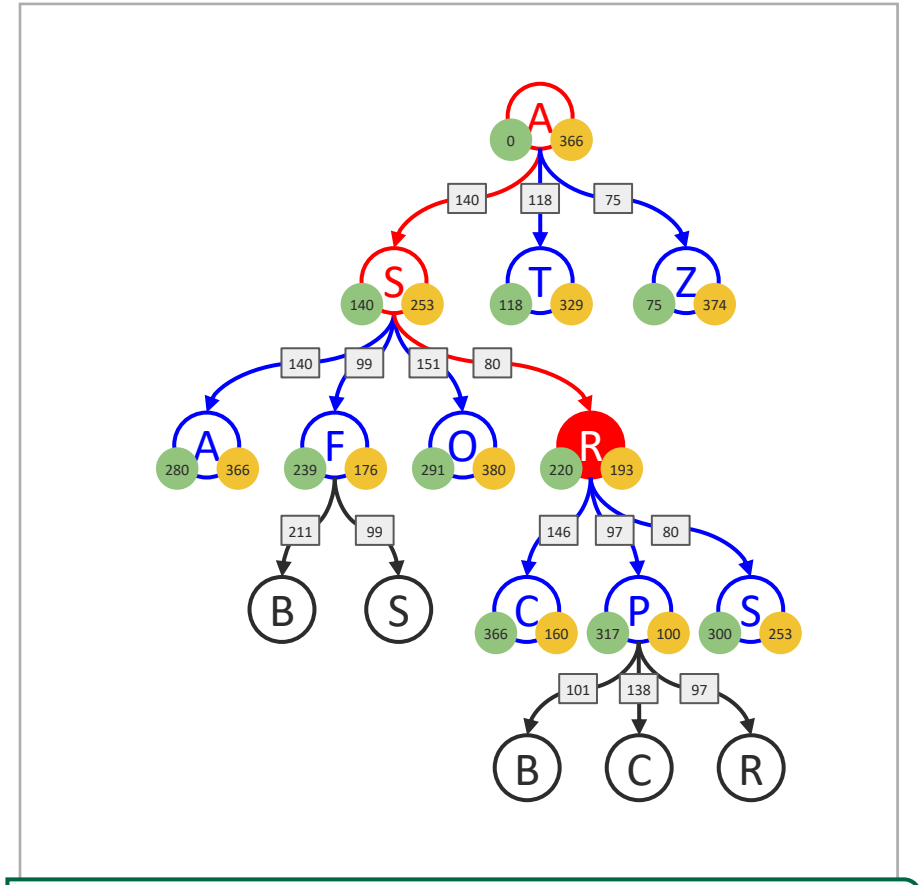

Figure 3. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

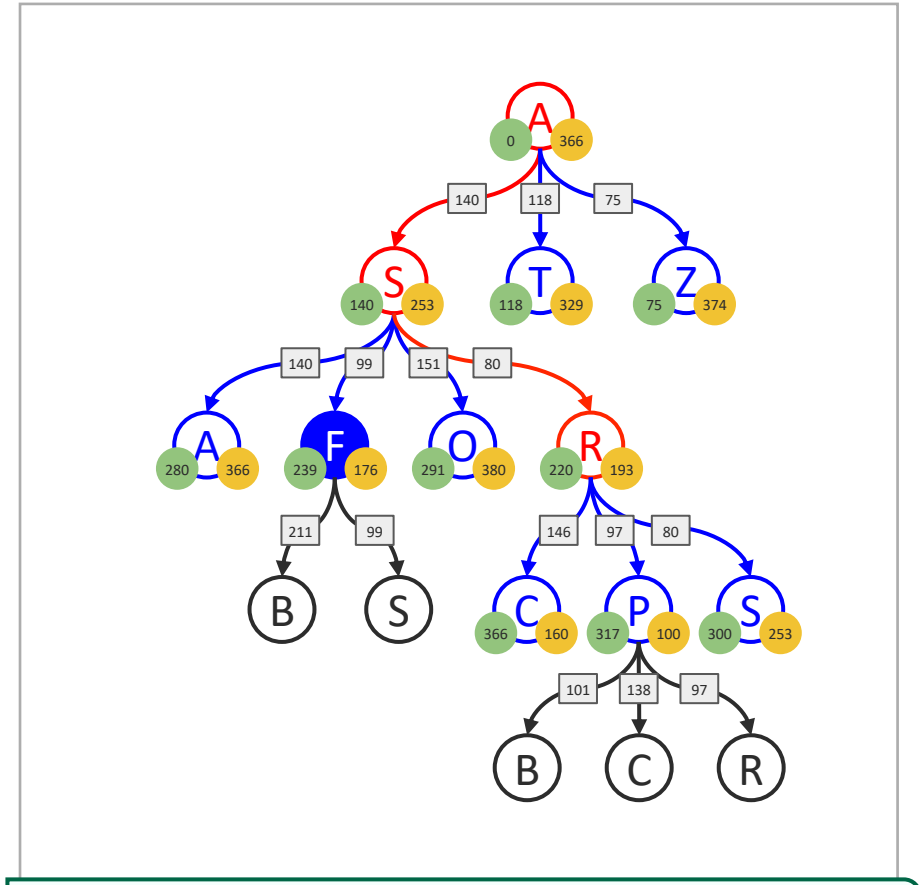

Figure 3. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

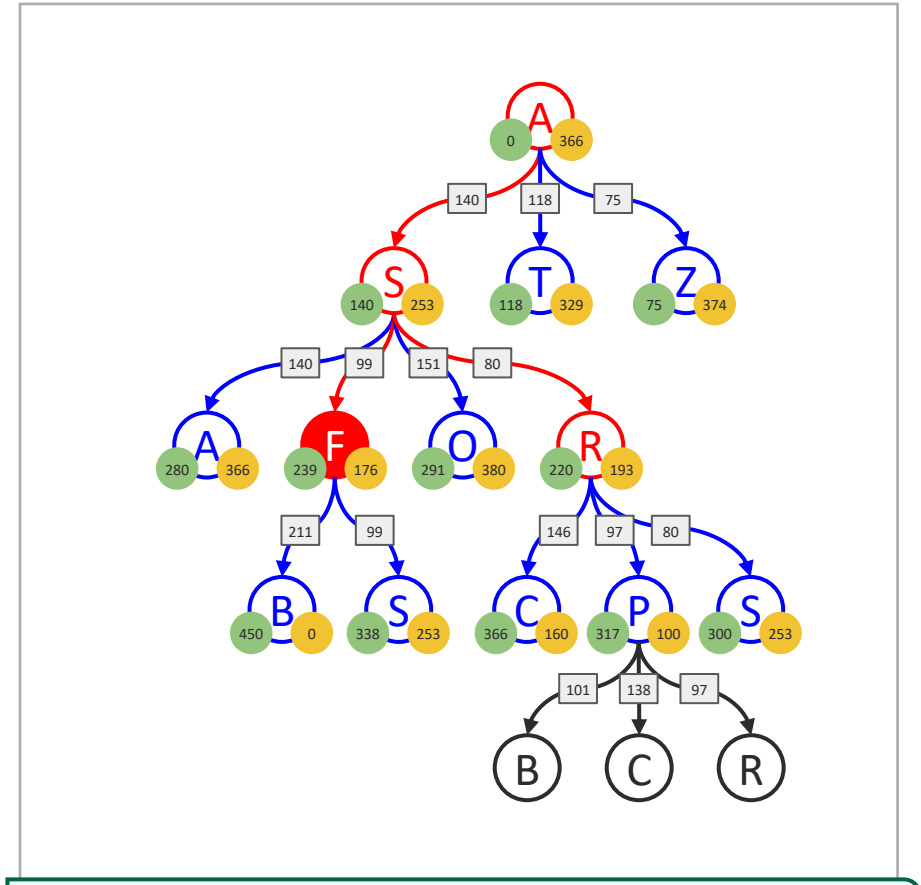

Figure 3. Search tree.

Knowledge Check 1

Can we stop A* as soon as a goal state is added to the tree?

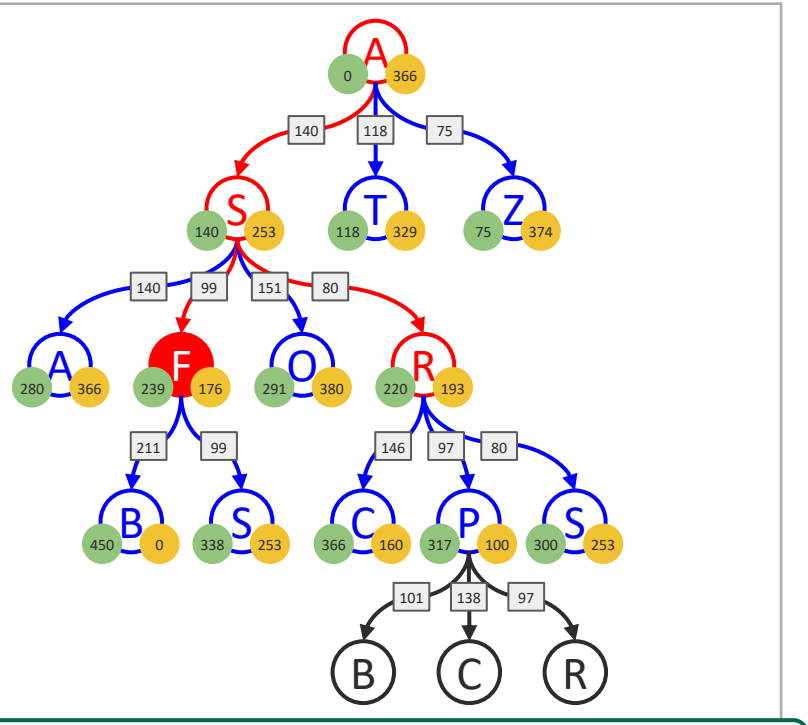

Figure 4. Search tree

A. Yes

B. No

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

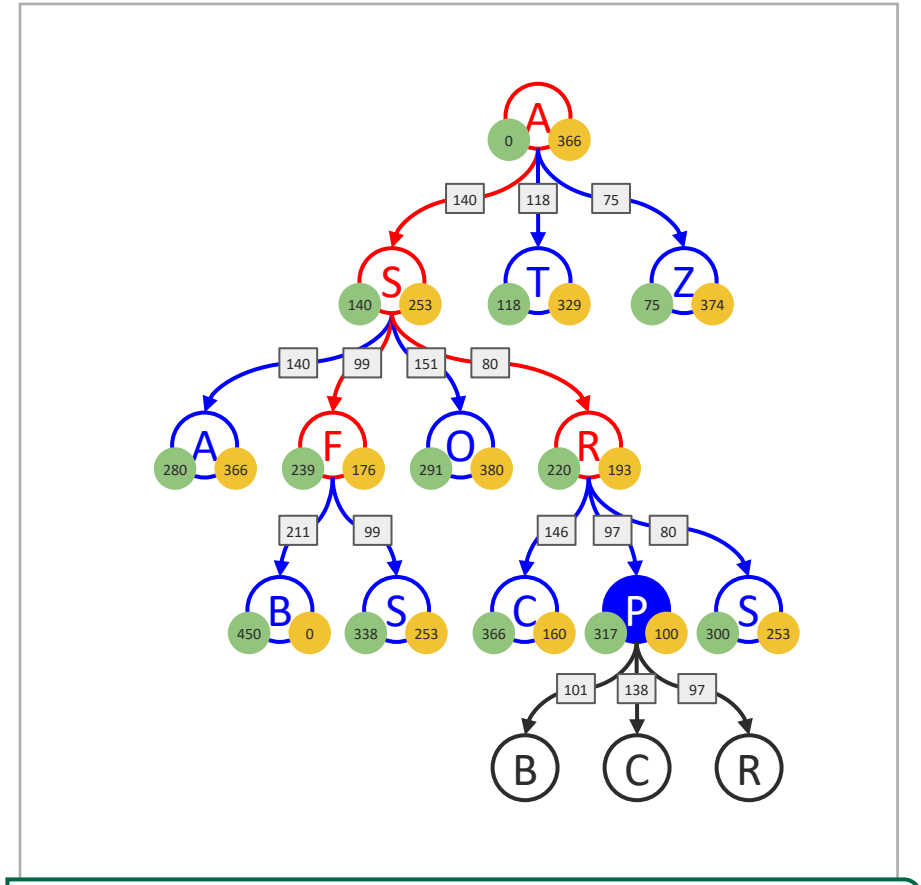

Figure 5. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

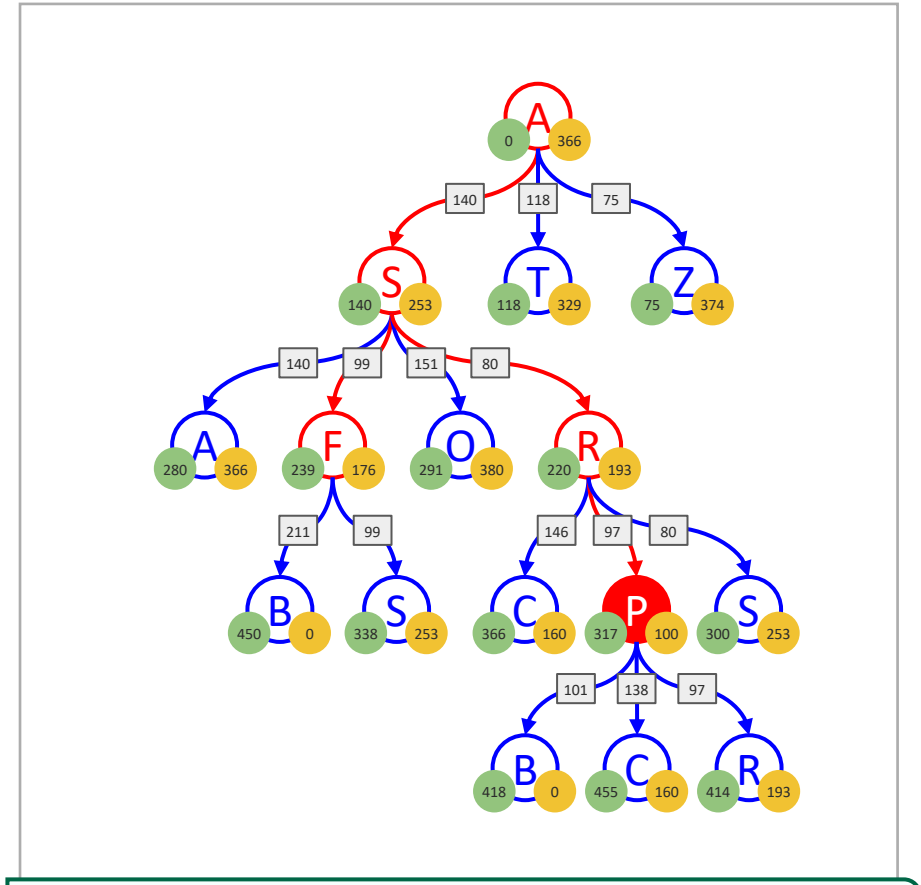

Figure 5. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

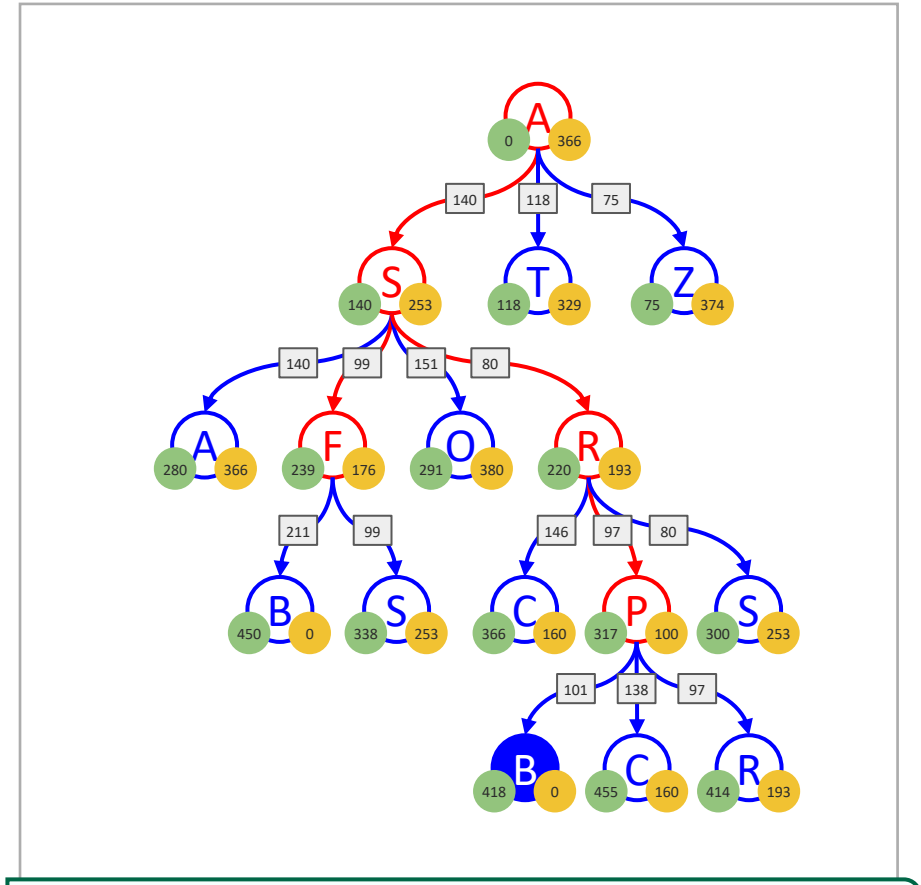

Figure 5. Search tree.

A* search

```
set initial state of problem as the tree root
while True:
  choose node with lowest f(n) value for expansion
  if there are no candidates for expansion
    return failure
  if chosen node is a goal state
    return path from root to node
  else
    expand node and add neighbors to the tree
```

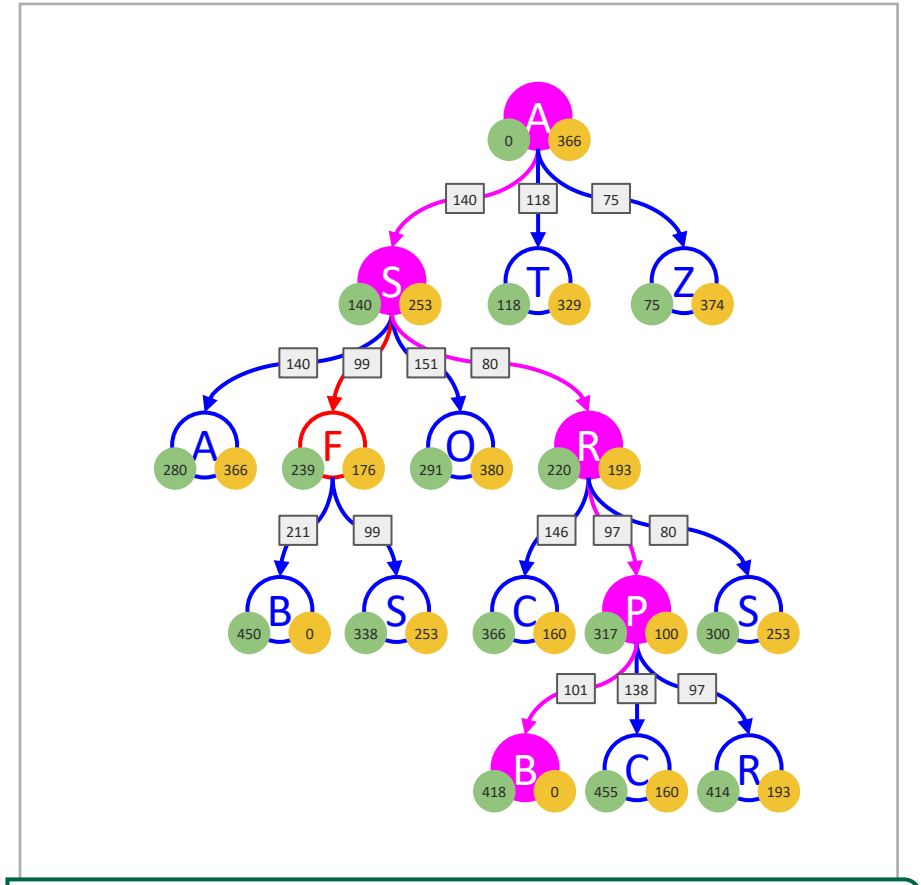

Figure 5. Search tree.

Traveling from Arad to Bucharest

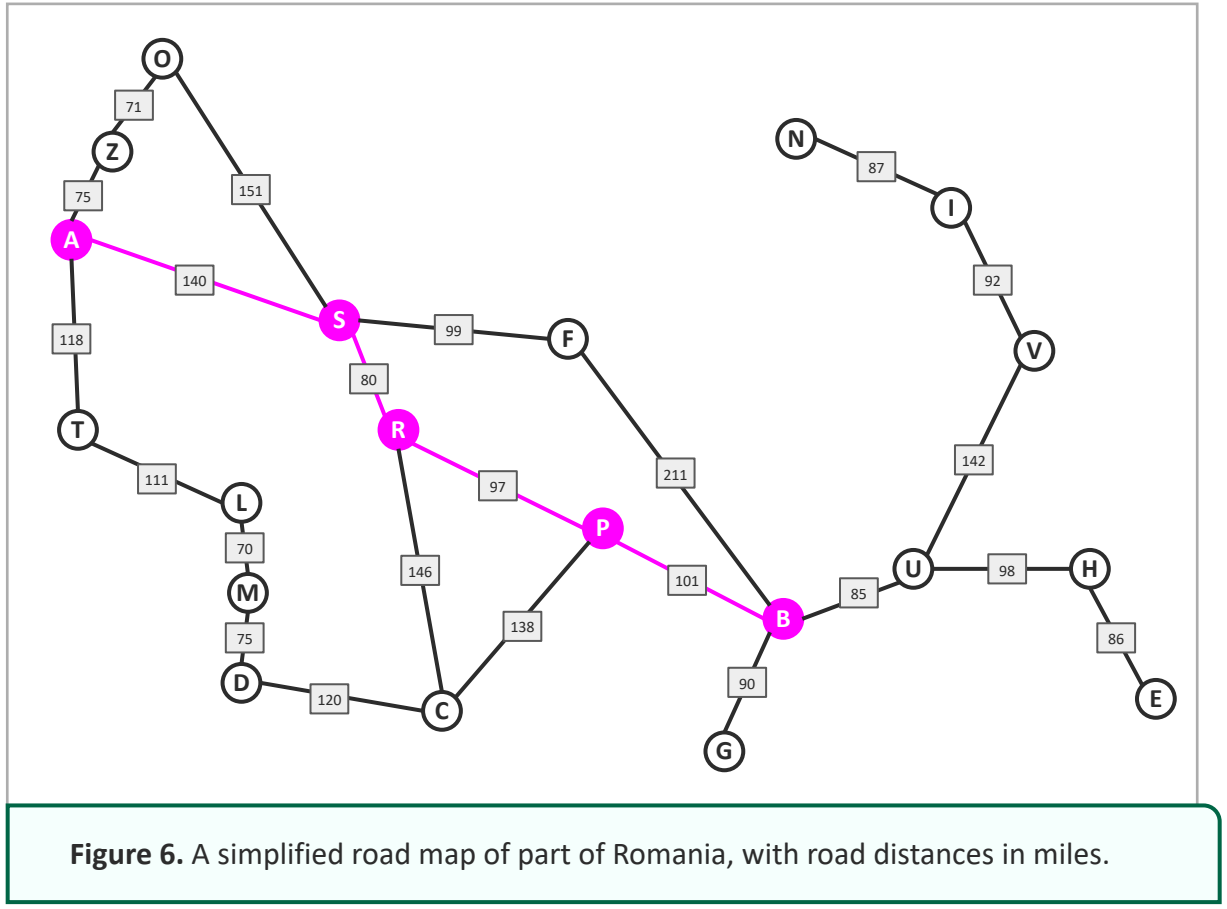

Figure 6. A simplified road map of part of Romania, with road distances in miles.

Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Table 1. Heuristic function: straight-line distances to Bucharest.

Knowledge Check 2

Is A* complete?

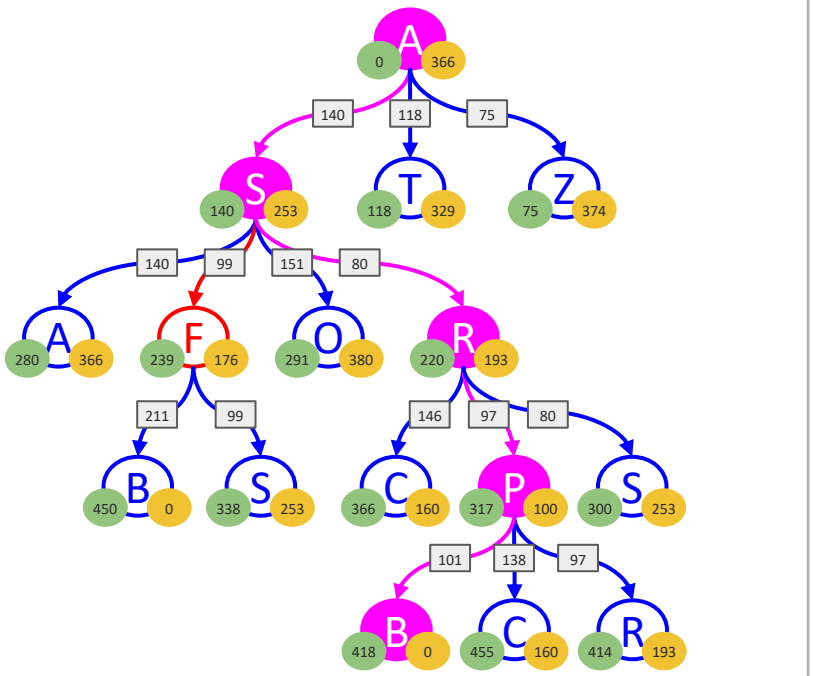

Figure 7. Search tree

A Yes

B No

Admissible heuristics

- A* being cost-optimal depends on certain properties of the heuristic
- The key property is admissibility
  - An admissible heuristic is one that never overestimates the lowest cost to reach a goal
  - An admissible heuristic is therefore optimistic
- With an admissible heuristic, A* is cost-optimal

Knowledge Check 3

The heuristic **h(n) = 0** for all nodes is admissible, although not very useful. If we use this heuristic function, A* will behave like which search algorithm?

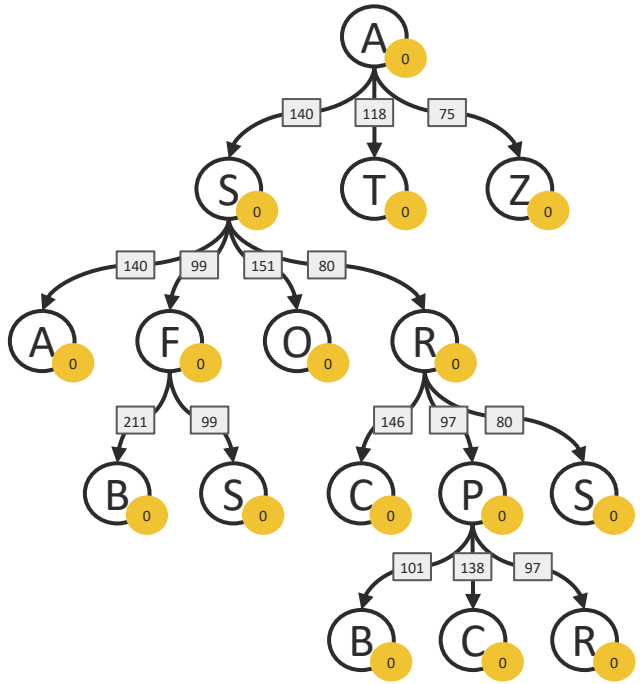

Figure 8. Search tree.

A DFS

B BFS

C UCS

D Iterative Deepening

Consistent heuristics

- A heuristic **h(n)** is consistent if, for every node **n** and every successor **n′** of **n** generated by an action **a**, we have:

$$\mathbf{h(n) \leq c(n, a, n') + h(n')}$$

- Every consistent heuristic is admissible (but not vice versa)
- With a consistent heuristic, the first time we reach a state it will be on an optimal path (we can stop early)

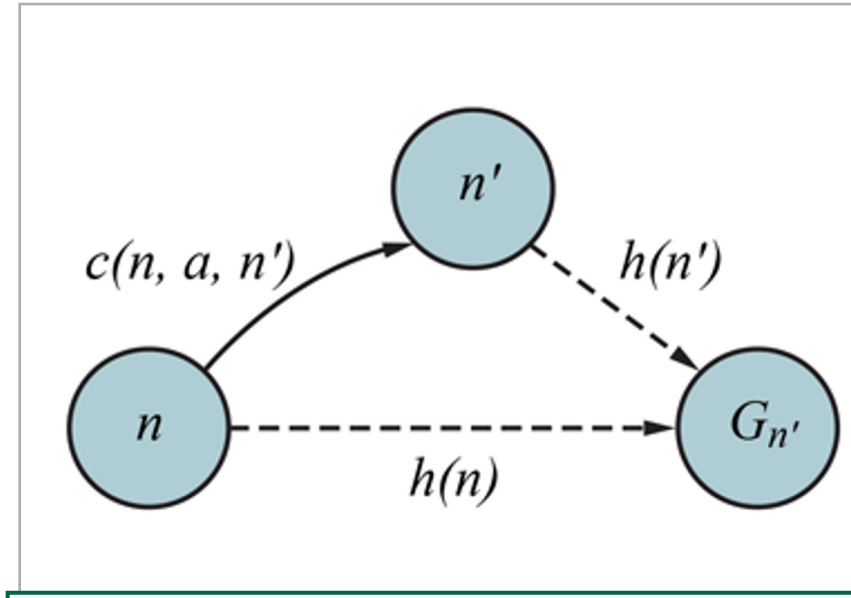

Figure 9. Triangle inequality.

Inadmissible heuristics

- With an inadmissible heuristic, A* may be cost-optimal
  - If there is even one cost-optimal path on which **h(n)** is admissible for all nodes **n** on the path
  - If the optimal solution has cost $\mathbf{C^*}$, and the second-best has cost $\mathbf{C_2}$, and if **h(n)** overestimates some costs, but never by more than $\mathbf{C_2 - C^*}$

Weighted A*

- A* search has many good qualities, but it expands a lot of nodes
- We can explore fewer nodes if we are willing to accept solutions that are “good enough”
- If we use an inadmissible heuristic, we risk missing the optimal solution, but potentially reduce the number of nodes expanded
- We can weight the heuristic value more heavily

$$\mathbf{f(n) = g(n) + W \times h(n)}\text{, for some } \mathbf{W > 1}$$

A* Analysis

Is it complete? Is it guaranteed to find a solution if one exists?

What is its time complexity?

Is it optimal? Is it guaranteed to find the least cost path?

What is its space complexity?

A* Analysis

Is it complete? Is it guaranteed to find a solution if one exists?

- Yes, as long as all action costs are greater than zero and the state space either has a solution or is finite

Is it optimal? Is it guaranteed to find the least cost path?

What is its time complexity?

What is its space complexity?

A* Analysis

Is it complete? Is it guaranteed to find a solution if one exists?

- Yes, as long as all action costs are greater than zero and the state space either has a solution or is finite

Is it optimal? Is it guaranteed to find the least cost path?

- Yes, as long as the heuristic function is admissible

What is its time complexity?

What is its space complexity?

A* Analysis

Is it complete? Is it guaranteed to find a solution if one exists?

- Yes, as long as all action costs are greater than zero and the state space either has a solution or is finite

Is it optimal? Is it guaranteed to find the least cost path?

- Yes, as long as the heuristic function is admissible

What is its time complexity?

- In the worst case, is equivalent to UCS - $O(b^{C/\varepsilon})$

What is its space complexity?

A* Analysis

Is it complete? Is it guaranteed to find a solution if one exists?

- Yes, as long as all action costs are greater than zero and the state space either has a solution or is finite

Is it optimal? Is it guaranteed to find the least cost path?

- Yes, as long as the heuristic function is admissible

What is its time complexity?

- In the worst case, is equivalent to UCS - $O(b^{C/\varepsilon})$

What is its space complexity?

- In the worst case, is equivalent to UCS - $O(b^{C/\varepsilon})$

You have reached the end of the lecture.

Image/Figure References

Figure 1. A simplified road map of part of Romania, with road distances in miles. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Table 1. Heuristic function: straight-line distances to Bucharest. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 2. Search tree. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 3. Search tree. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 4. Search tree. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 5. Search tree. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 6. A simplified road map of part of Romania, with road distances in miles. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 7. Search tree. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 8. Search tree. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.
Figure 9. Triangle inequality. Source: Russell & Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.