Introduction to Computational Intelligence

Lecture 6

This Session

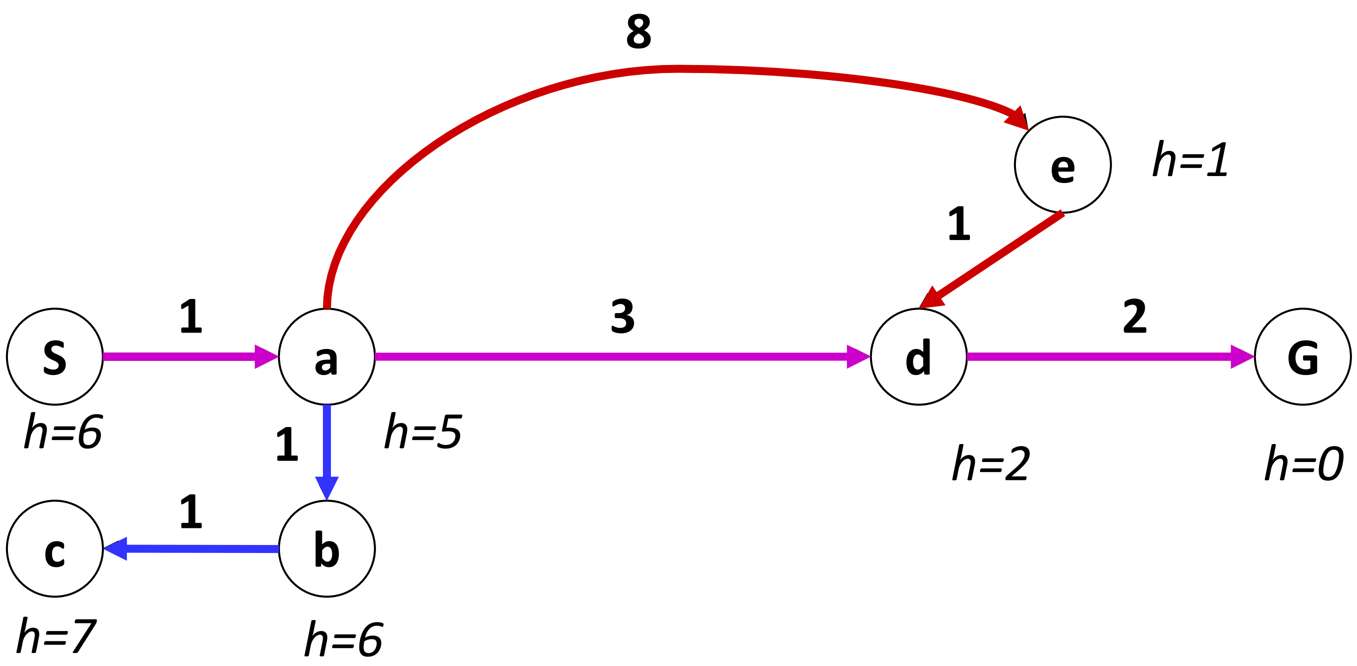
- Informed search strategies
  - Greedy best-first search
  - A* search

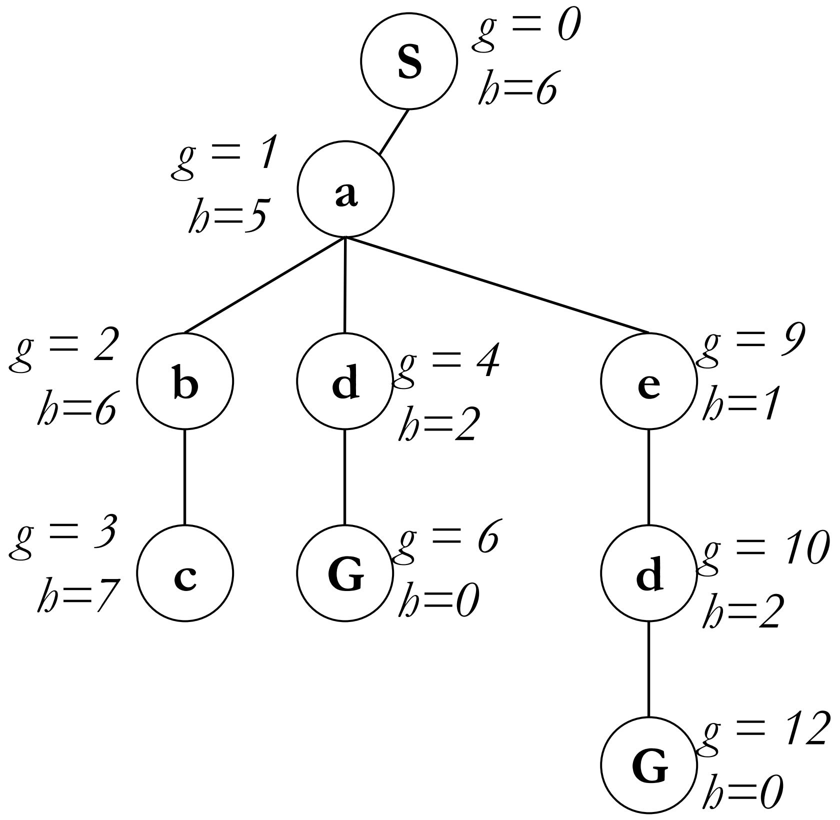
A* search - Recap

- **Idea:** avoid expanding paths that are already expensive based on a cost function.

- **Cost function:** $f(n) = g(n) + h(n)$

A* Search – Example



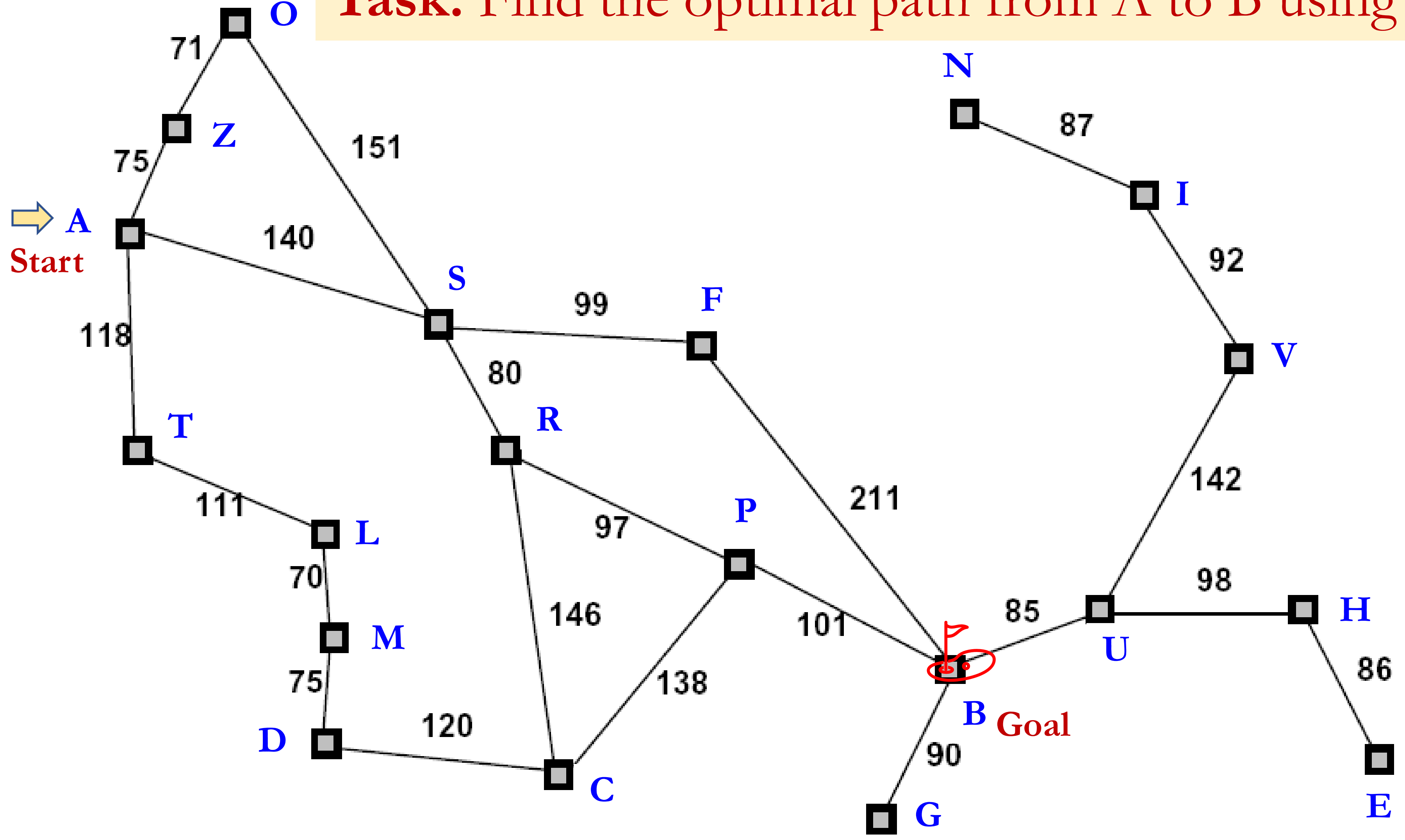


- Uniform-cost orders by backward path cost g(n)
- Greedy orders by forward path cost h(n)
- A* Search orders by total path cost $f(n) = g(n) + h(n)$

Example by: Teg Grenager

A* Search – Working Example

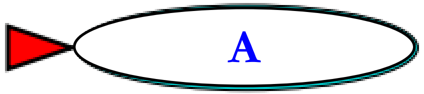
Task: Find the optimal path from A to B using A* search.



$h(n)$

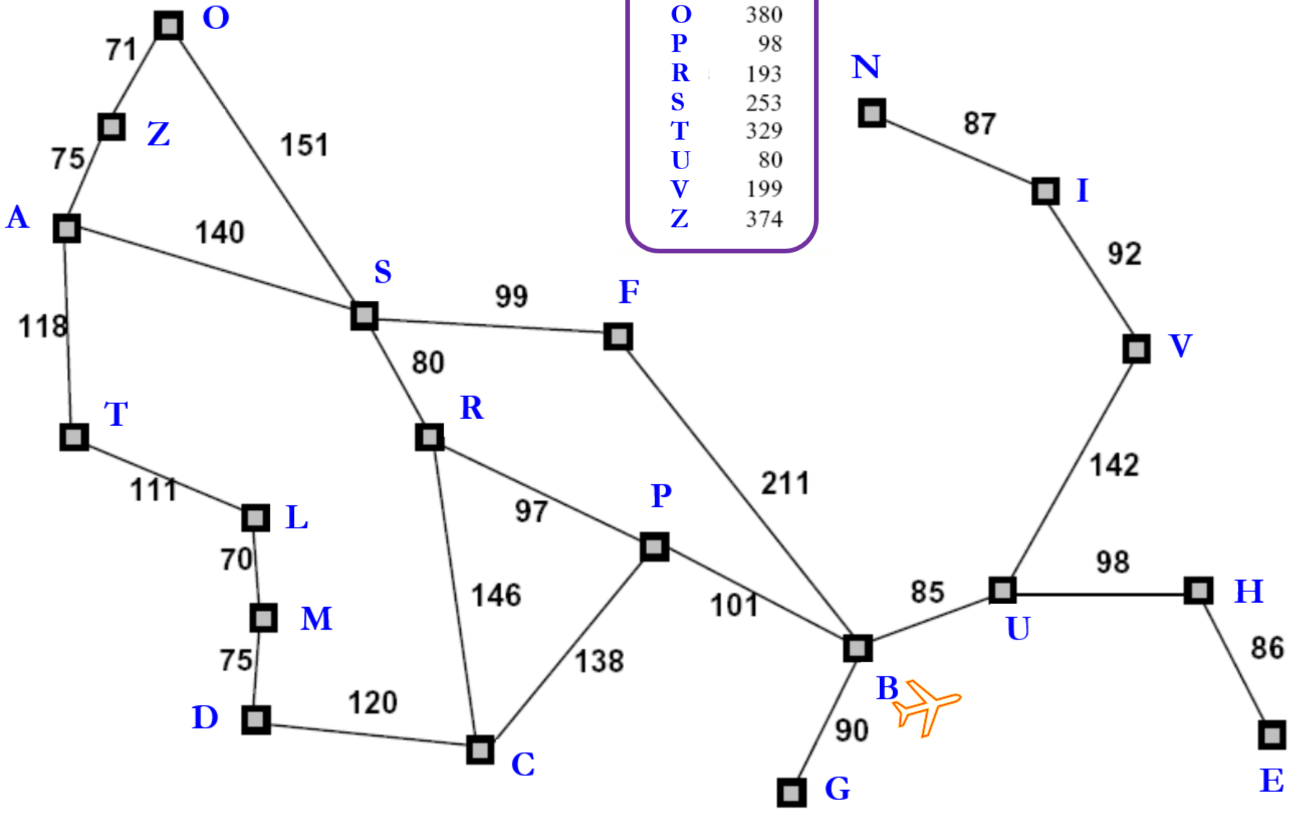
A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374

A* Search – Work Example Cont.

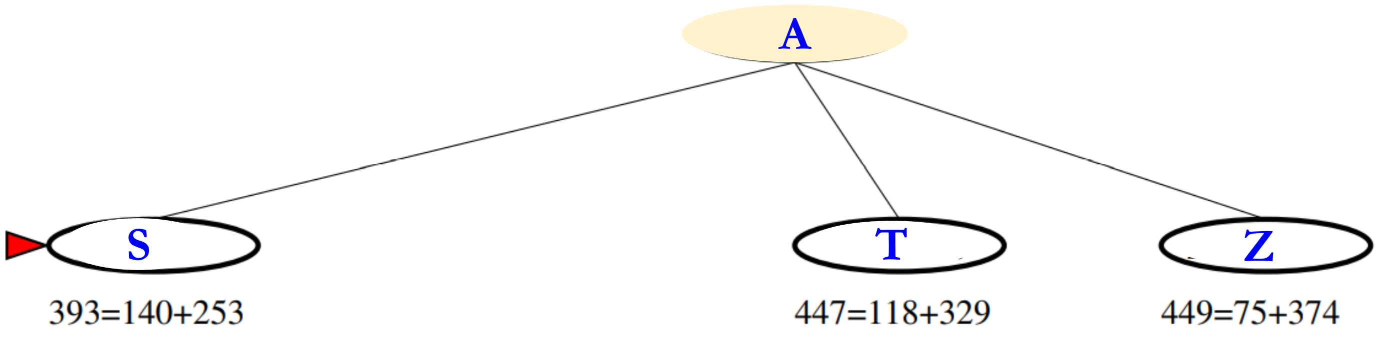


366=0+366

$h(n)$	
A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374

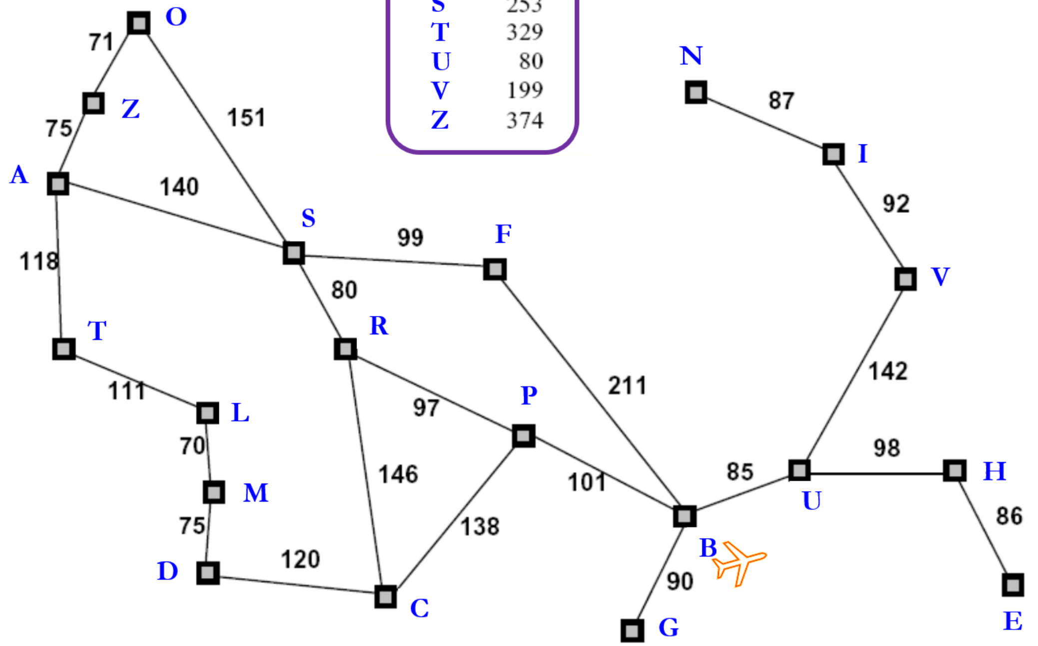


A* Search – Working Example: After *A* Expanded

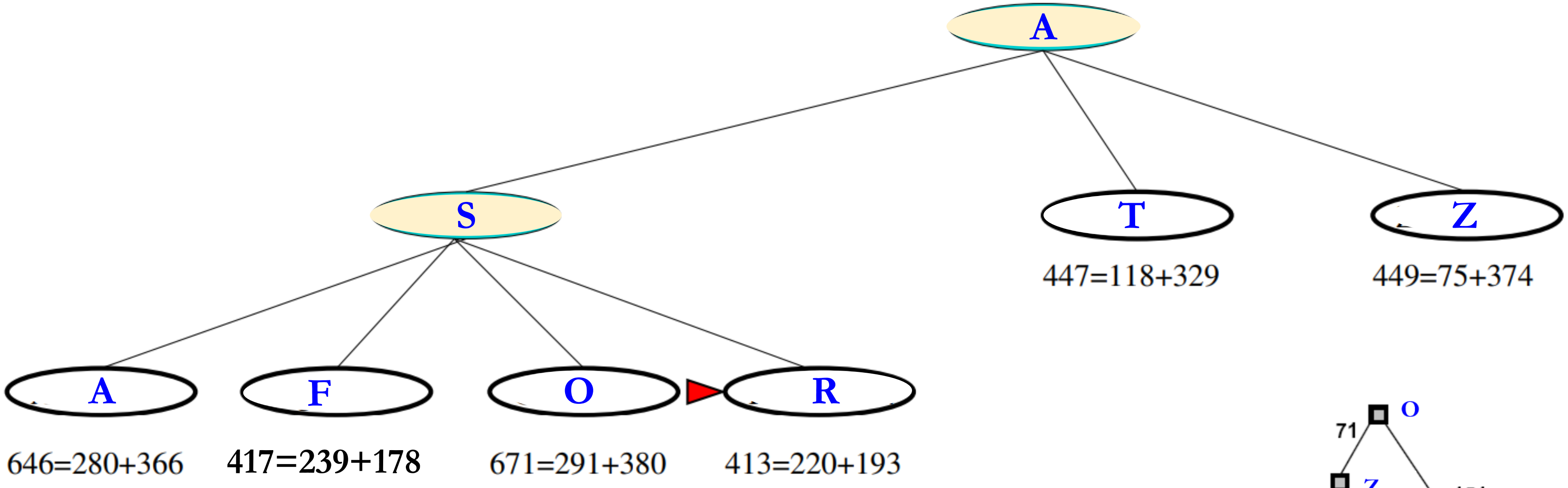


$h(n)$

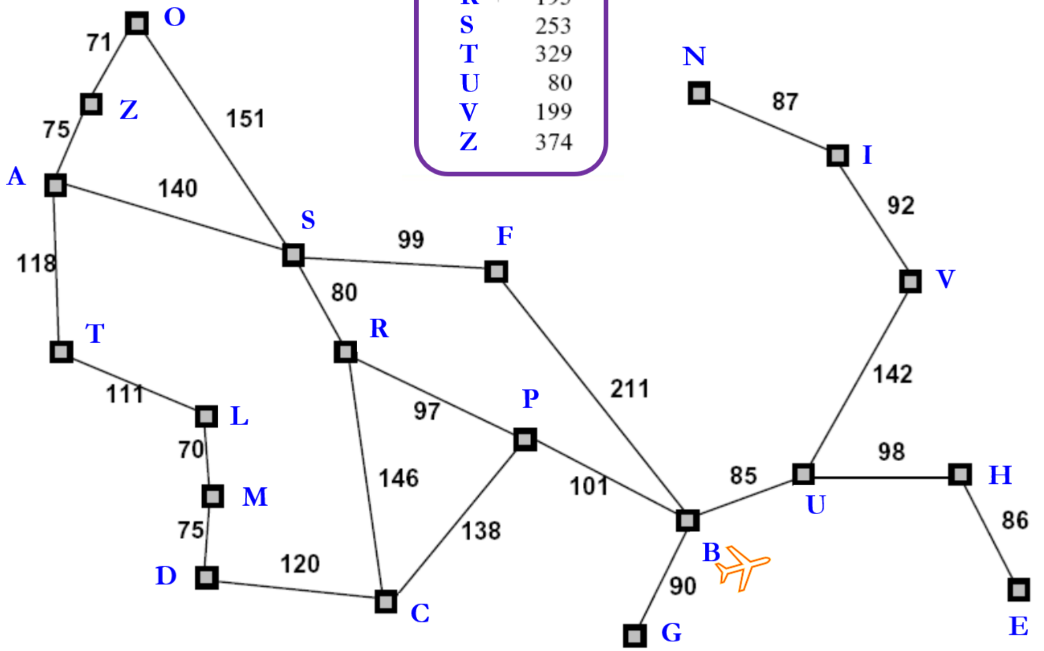
A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374



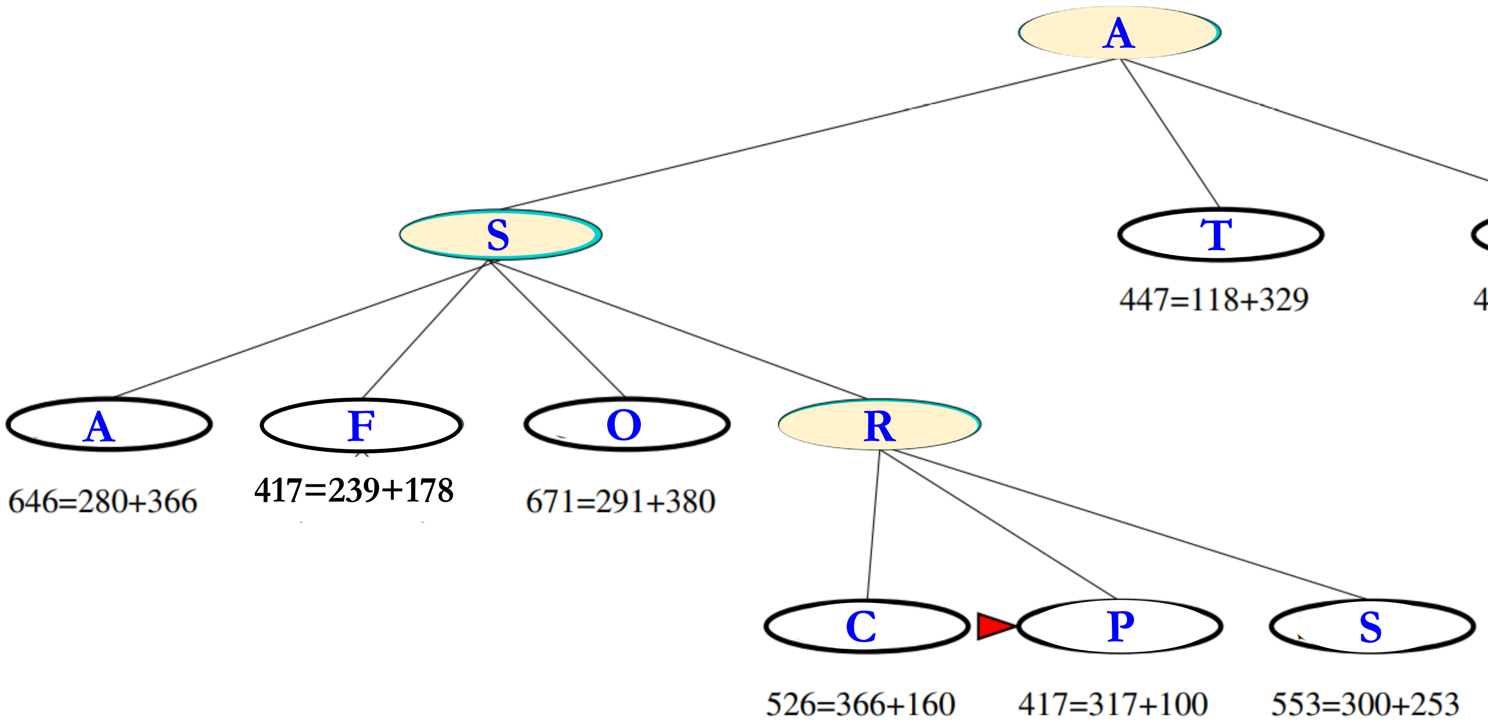
A* Search – Working Example: After *S* Expanded



$h(n)$	
A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374

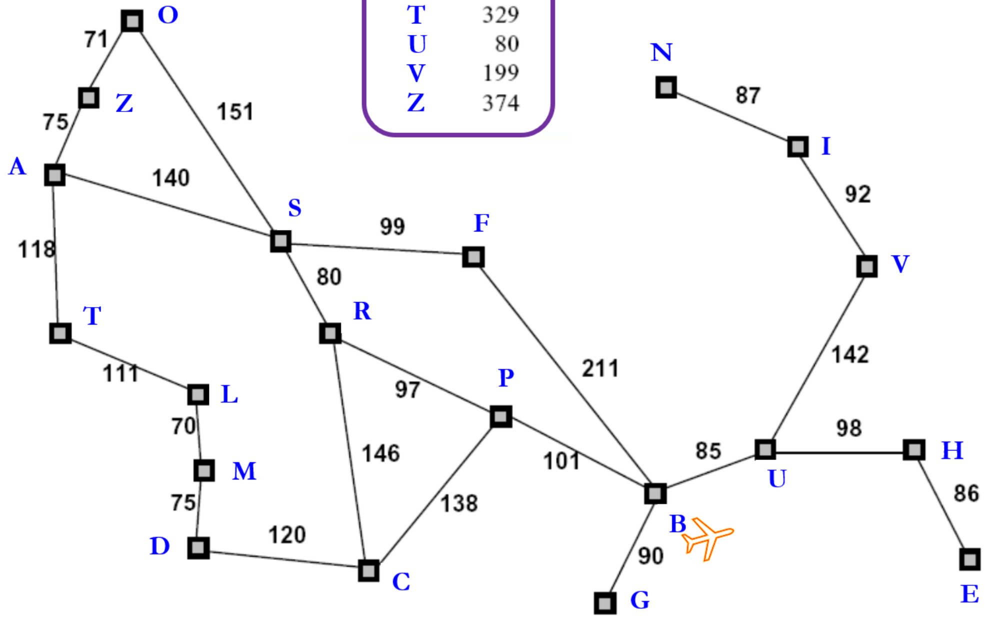


A^* Search – Working Example: After R Expanded

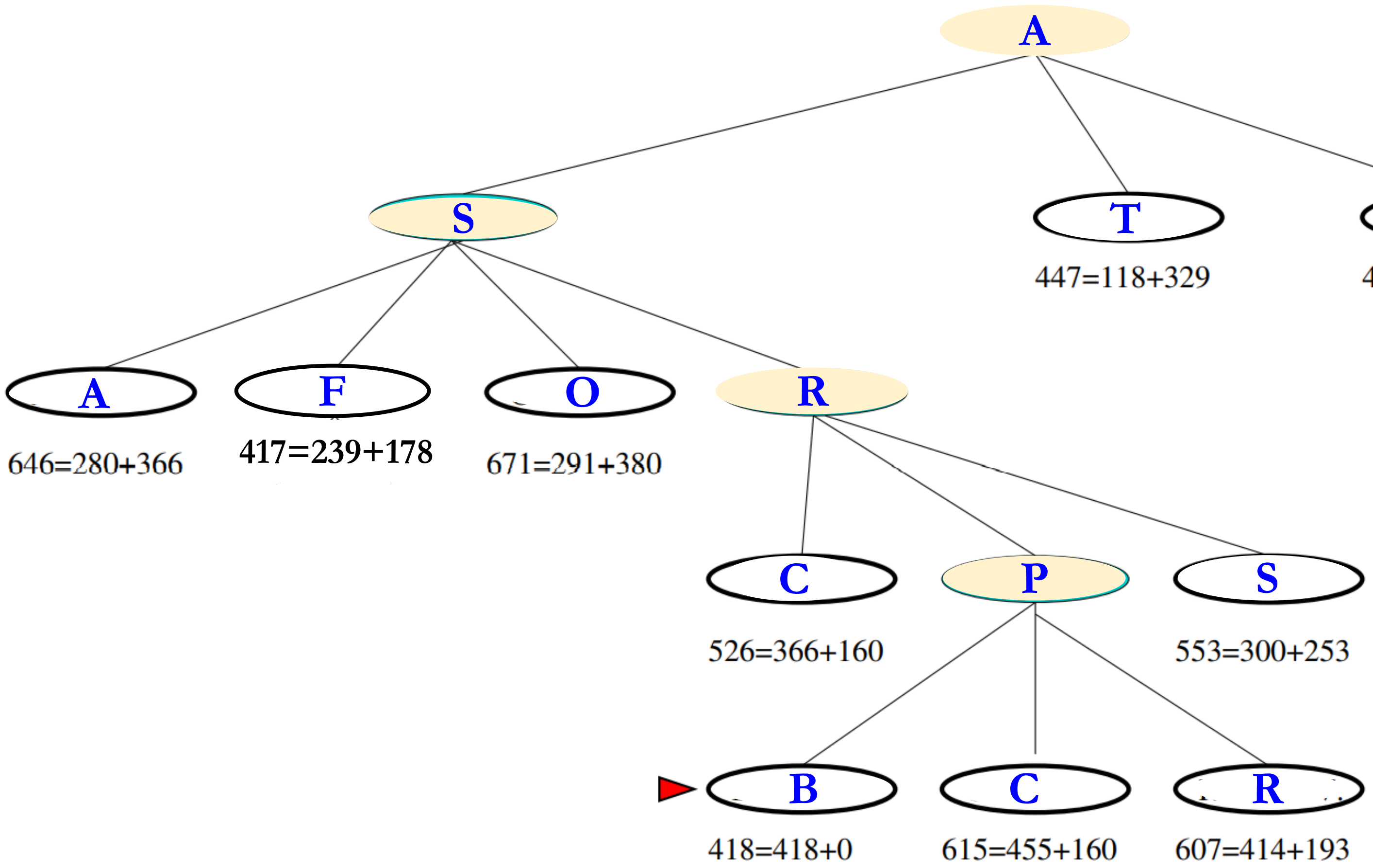


$h(n)$

A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374

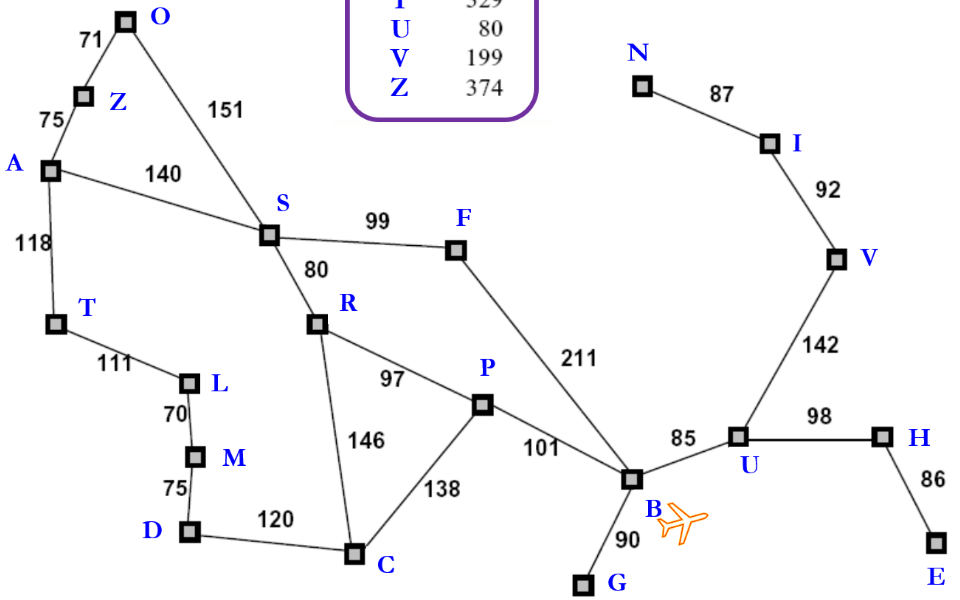


A* Search – Working Example: After *P* Expanded



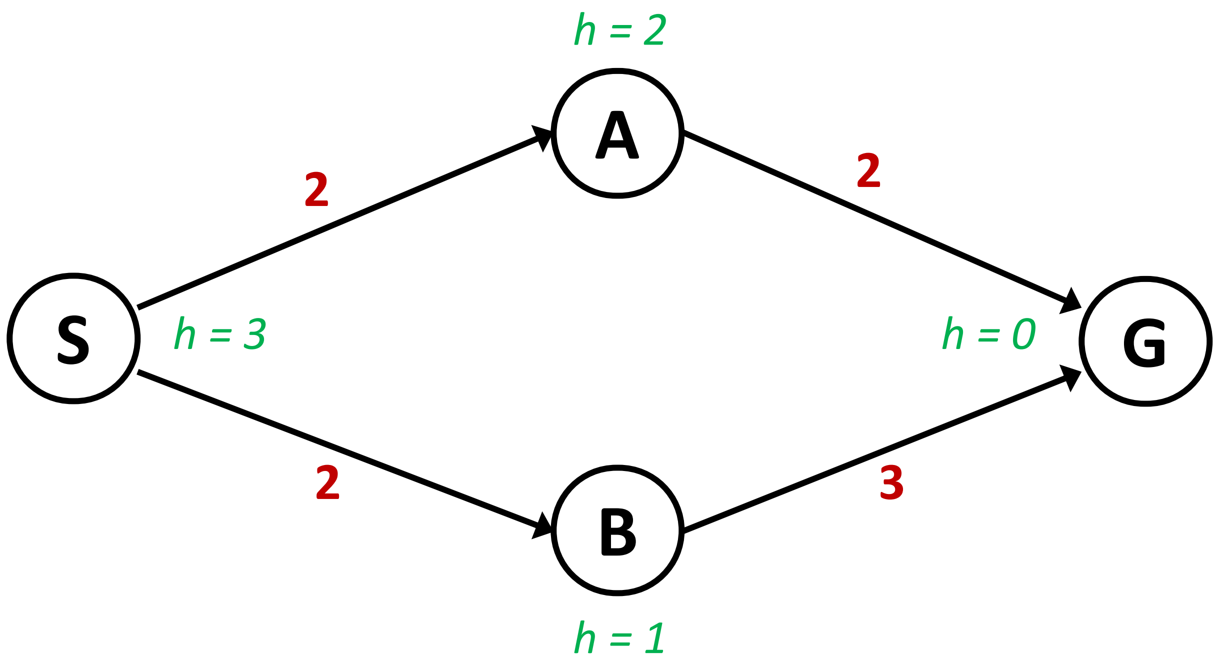
$h(n)$

A	366
B	0
C	160
D	242
E	161
F	178
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	98
R	193
S	253
T	329
U	80
V	199
Z	374



A* Search – Condition for Search Termination

- Should we stop when we enqueue a goal?

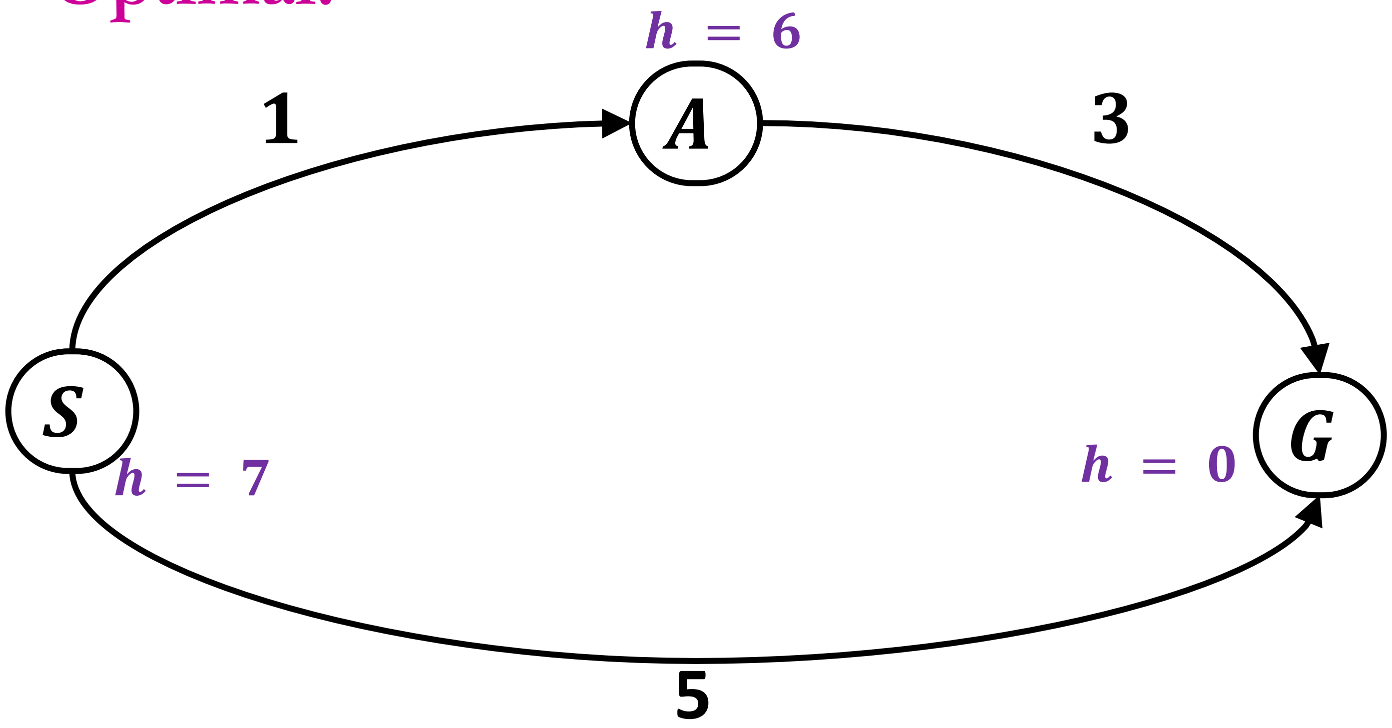


Path	g(n)	h(n)	f(n)

- No! only stop when we dequeue a goal

Properties of A*

- **Optimal?**

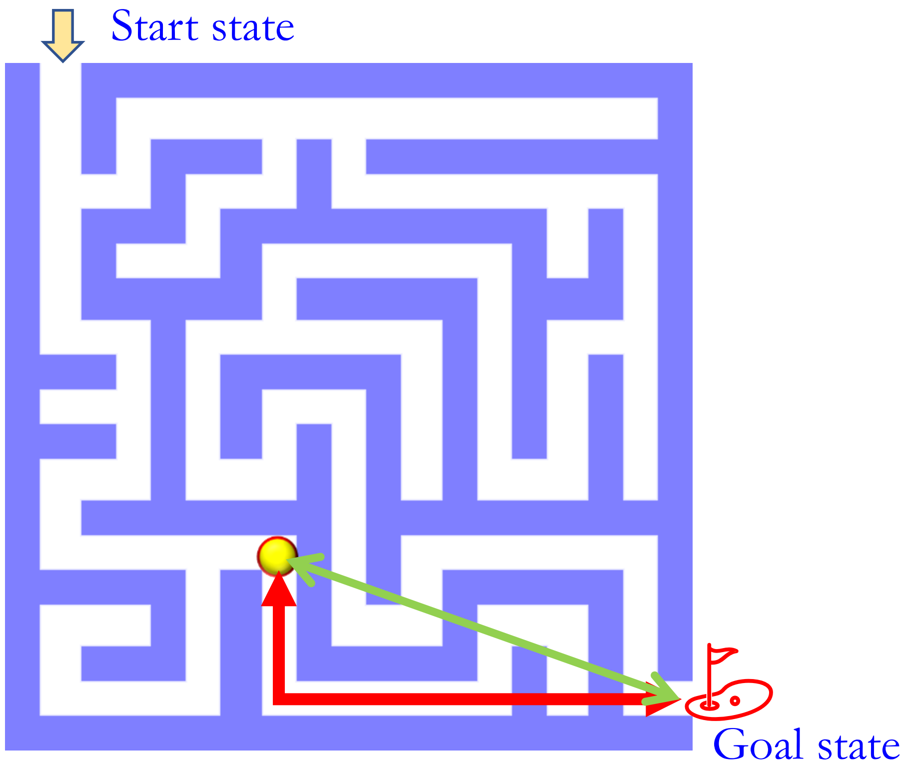


Path	g(n)	h(n)	f(n)

- **Problem**: Actual bad goal cost < estimated good goal cost (5 < 6)
- **Solution:** The estimates must be less than actual costs. Need a good $h(n)$, i.e., an **admissible heuristic**

Admissible Heuristics

- A heuristic $h(n)$ is **admissible** if for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n, and $h(n) > 0$.
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- E.g., **straight line distance** never overestimates the **actual road distance**
- **Theorem:** If $h(n)$ is admissible, A^* is optimal



Properties of A* Cont.

- **Optimal?**

  Yes - cannot expand f_{i+1} until f_i is finished

  A* expands all nodes with $f(n) < C^*$
  A* expands some nodes with $f(n) = C^*$
  A* expands no nodes with $f(n) > C^*$

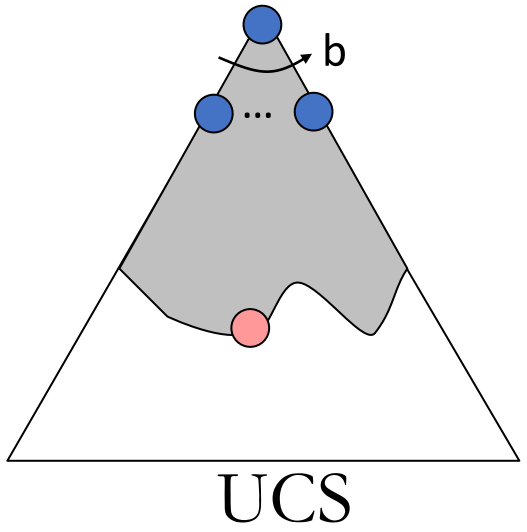
- **Complete?**

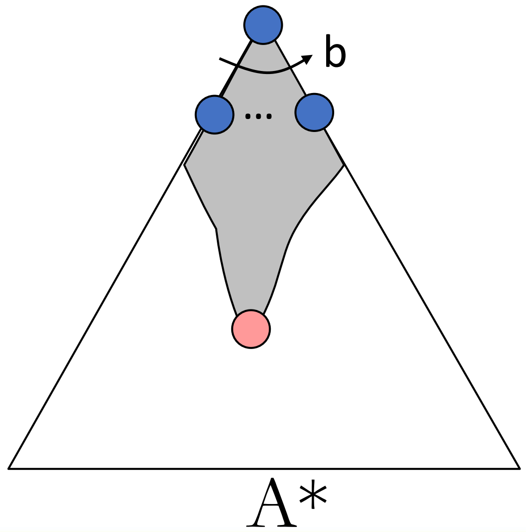
  Yes – unless there are infinitely many nodes with $f(n) \leq C^*$

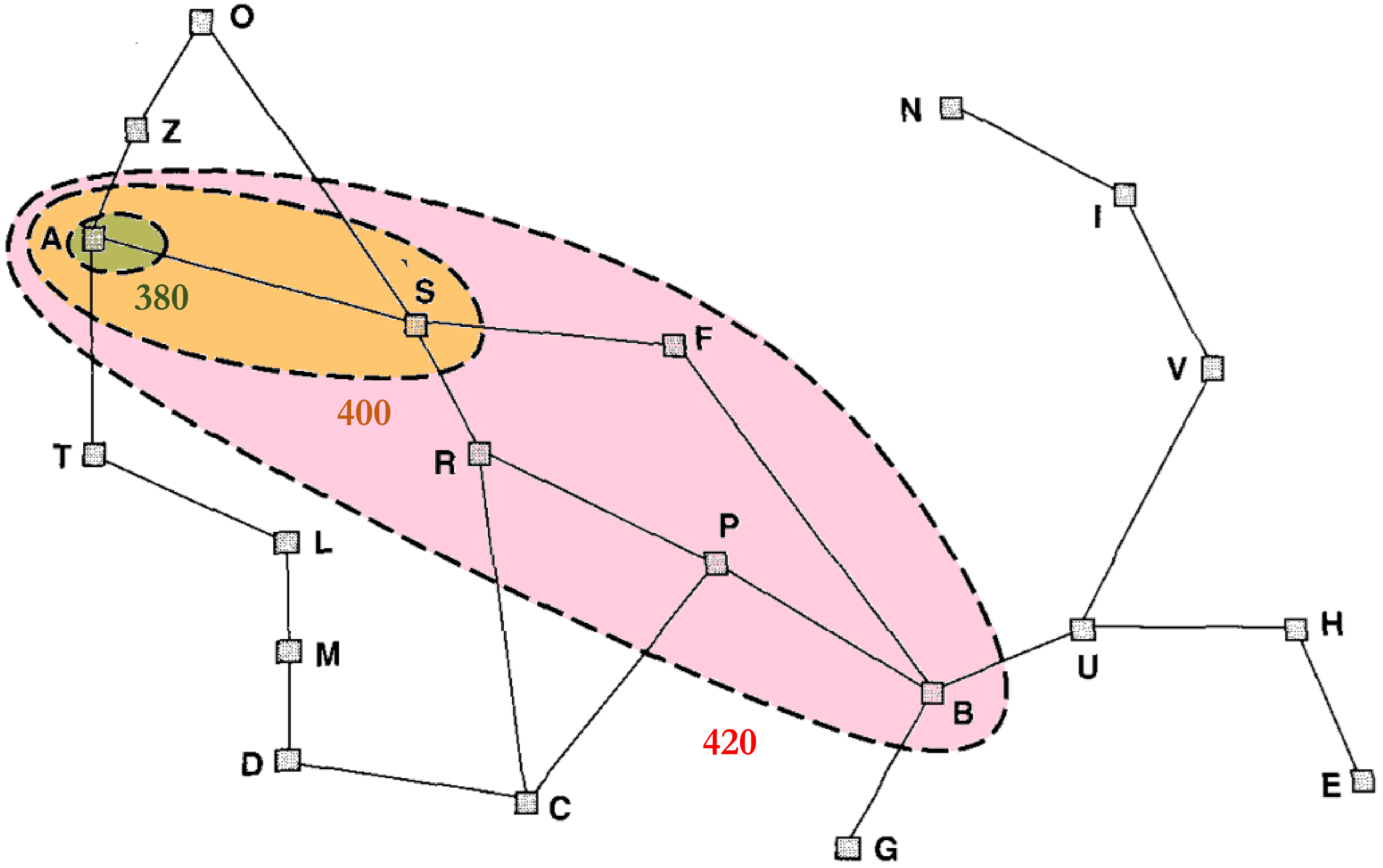
- **Time?**
  - Number of nodes for which $f(n) \leq C^*$
  - Exponential in (relative error in h × length of solution)

- **Space?**
  - Exponential as it keeps all nodes in memory

UCS vs A* Contours

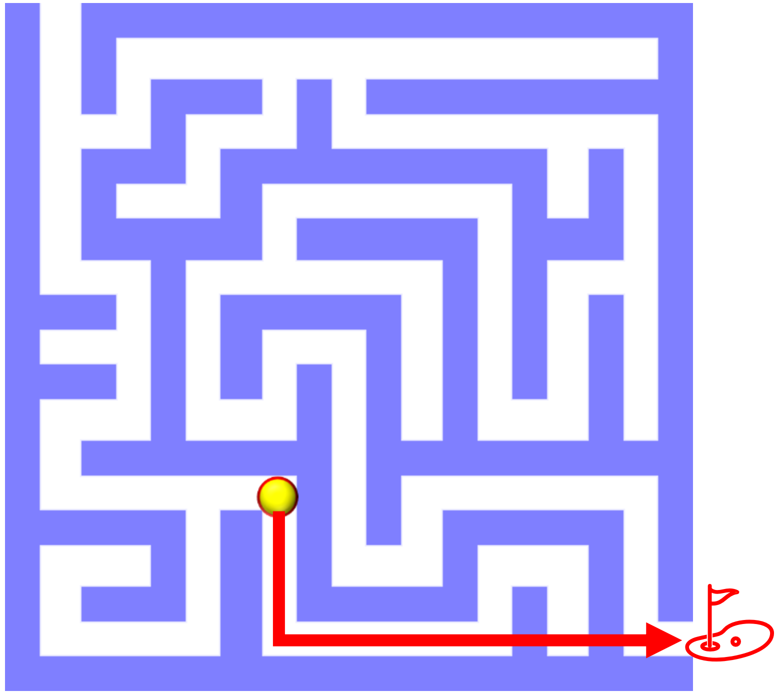




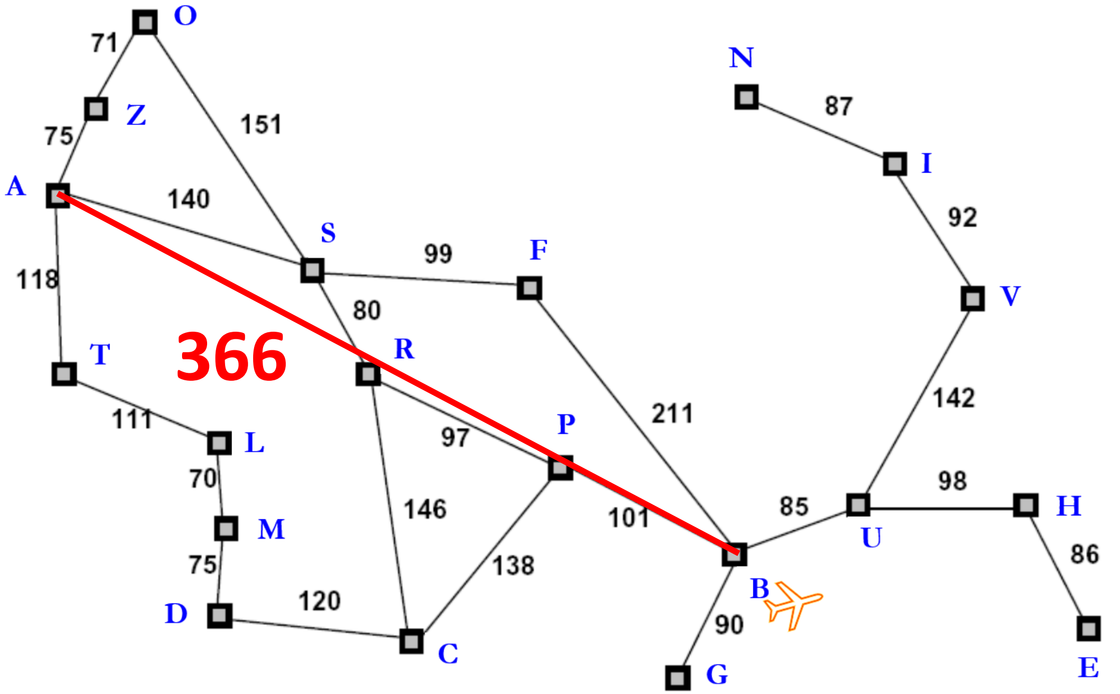


Designing Heuristic Functions

- The optimal solution of A* depends on coming up with **admissible heuristics.**



MD = 2+ 8 = 10



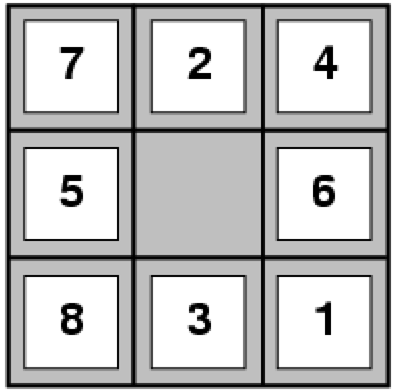
- Often, admissible heuristics are solutions to relaxed problems, where new actions are available

Designing Heuristic Functions Cont.

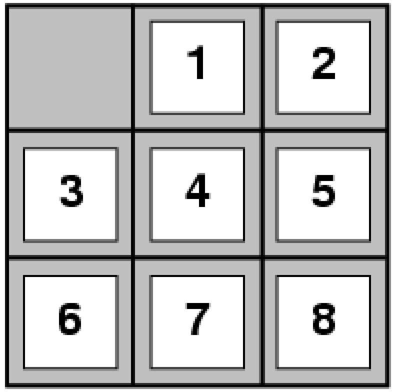
- Heuristics for the 8-puzzle

  $h_1(n)$ = number of misplaced tiles

  $h_2(n)$ = total Manhattan distance (number of squares from desired location of each tile)



Start State



Goal State

- Are h_1 and h_2 admissible?

Heuristics from Relaxed Problems

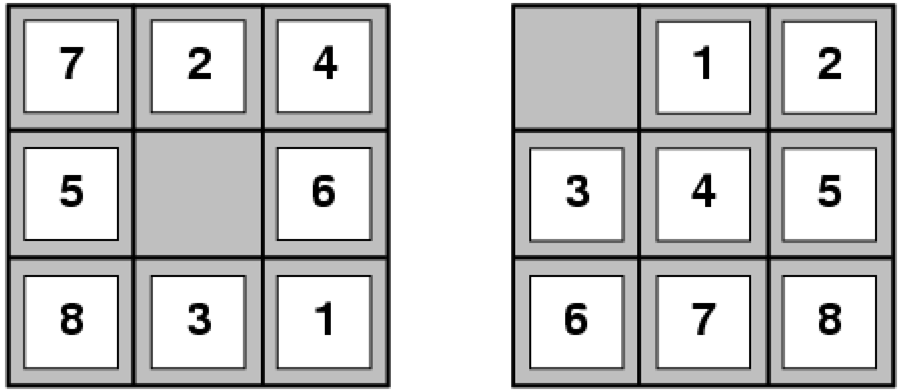
- A problem with fewer restrictions on the actions is called a relaxed problem
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem
- $h_1(n)$: If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution
- $h_2(n)$: If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution

Designing Heuristic Functions

- Heuristics for the 8-puzzle

  $h_1(n)$ = number of misplaced tiles

  $h_2(n)$ = total Manhattan distance (number of squares from desired location of each tile)



Start State Goal State

$h_1(\text{start}) = 8$

$h_2(\text{start}) = 3+1+2+2+2+3+3+2 = 18$

Dominance of Admissible Heuristics

- A* search expands every node with $f(n) < C^*$ or $h(n) < C^* - g(n)$.
- To minimize number of node expansions we need $h(n) \rightarrow$ exact cost.

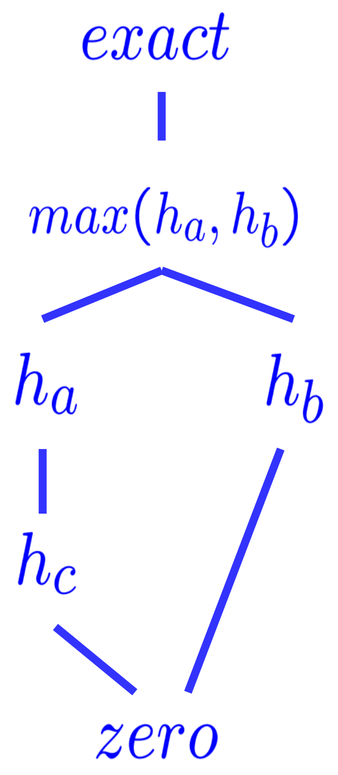
- **Multiple** admissible **heuristics**:
  - If h_1 and h_2 are both admissible heuristics and $h_2(n) \geq h_1(n)$ for all n, (both admissible), then h_2 dominates h_1
    - ✓ $\forall n, : h_2(n) \geq h_1(n)$
  - **Which one is better for A* search?**
    - ✓ $h_2(n)$ will expand fewer nodes, on average, than $h_1(n)$

Combining Admissible Heuristics

- Suppose we have a collection of admissible heuristics $h_1(n)$, $h_2(n)$, …, $h_m(n)$, but none of them consistently dominates the others.

- How do we pick one?
  - Apply max pooling:
    - ✓ $h_{new}(n) = \max\{h_1(n), h_2(n), \ldots, h_m(n)\}$
    - ✓ Max of admissible heuristics is admissible

⚠ **Trade off:** the computation to all the heuristics should not take too long - between quality of estimate and work per node.



Heuristics form a semi-lattice

Image adopted from Nikita Kitaev, CS 188: Artificial Intelligence, University of California, Berkeley

Memory-bounded Search

- The memory usage of A* can still be exorbitant
- How to make A* more memory-efficient while maintaining completeness and optimality?

- **Idea:** perform iterations of DFS - Iterative deepening A* search
  - The cutoff is defined based on the f-cost rather than the depth of a node.
  - Each iteration expands all nodes inside the contour for the current f-cost, peeping over the contour to find out where the contour lies.

Applications

- Video games
- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis

Summary

- Uniformed search strategies can only generate successors and distinguish goals from non-goals
- Informed Strategies that know whether one non-goal is more promising than another

- Greedy (best-first) search using f$(n) = g(n) + h(n)$ and an admissible $h(n)$ is known as $A*$ search

- A* search is complete & optimal with admissible and consistent heuristics

- Heuristic design is key:
  - Finding good heuristics for a specific problem is an area of research
  - Use relaxed problems

References

1. Eberhart, Russell C., and Yuhui Shi. Computational Intelligence : Concepts to Implementations, Elsevier Science & Technology, 2011.
2. Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach,4th Edition, 2020.
3. End-to-End Training of Deep Visuomotor Policies, Sergey Levine*, Chelsea Finn*, Trevor Darrell, Pieter Abbeel, JMLR 17, 2016.
4. AIMA slides (http://aima.cs.berkeley.edu/)