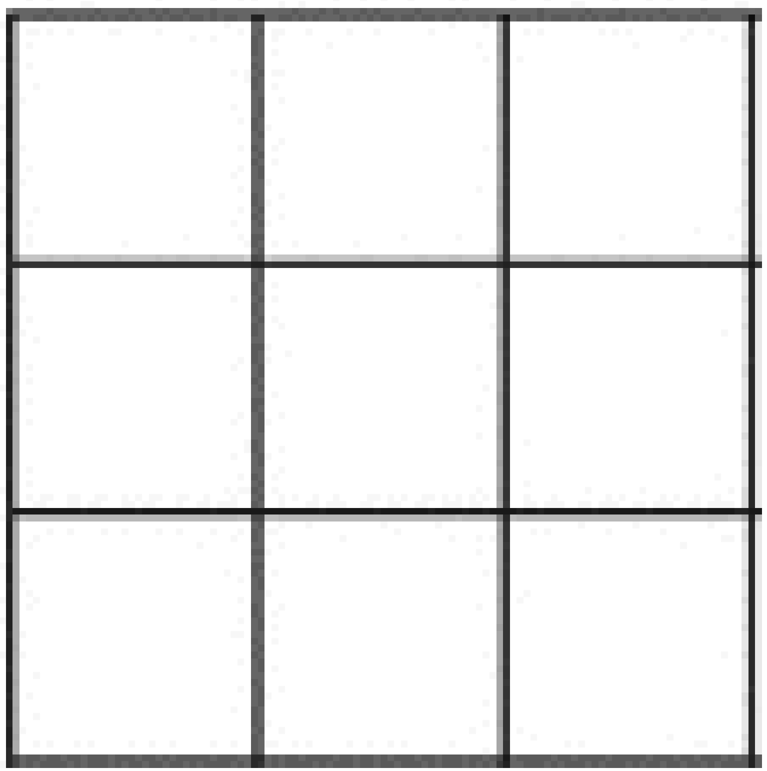
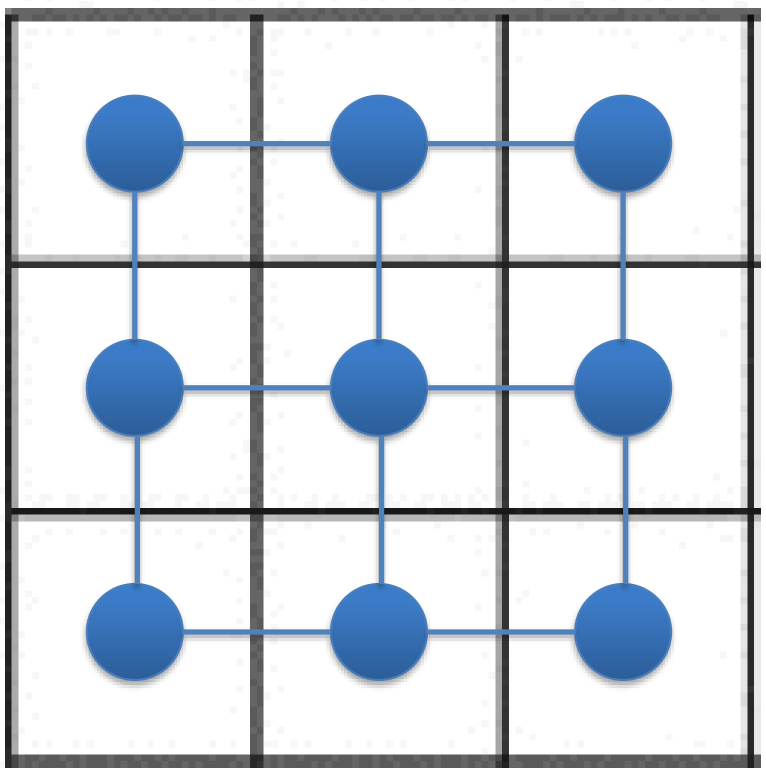
Graphs: Killer App in GAI

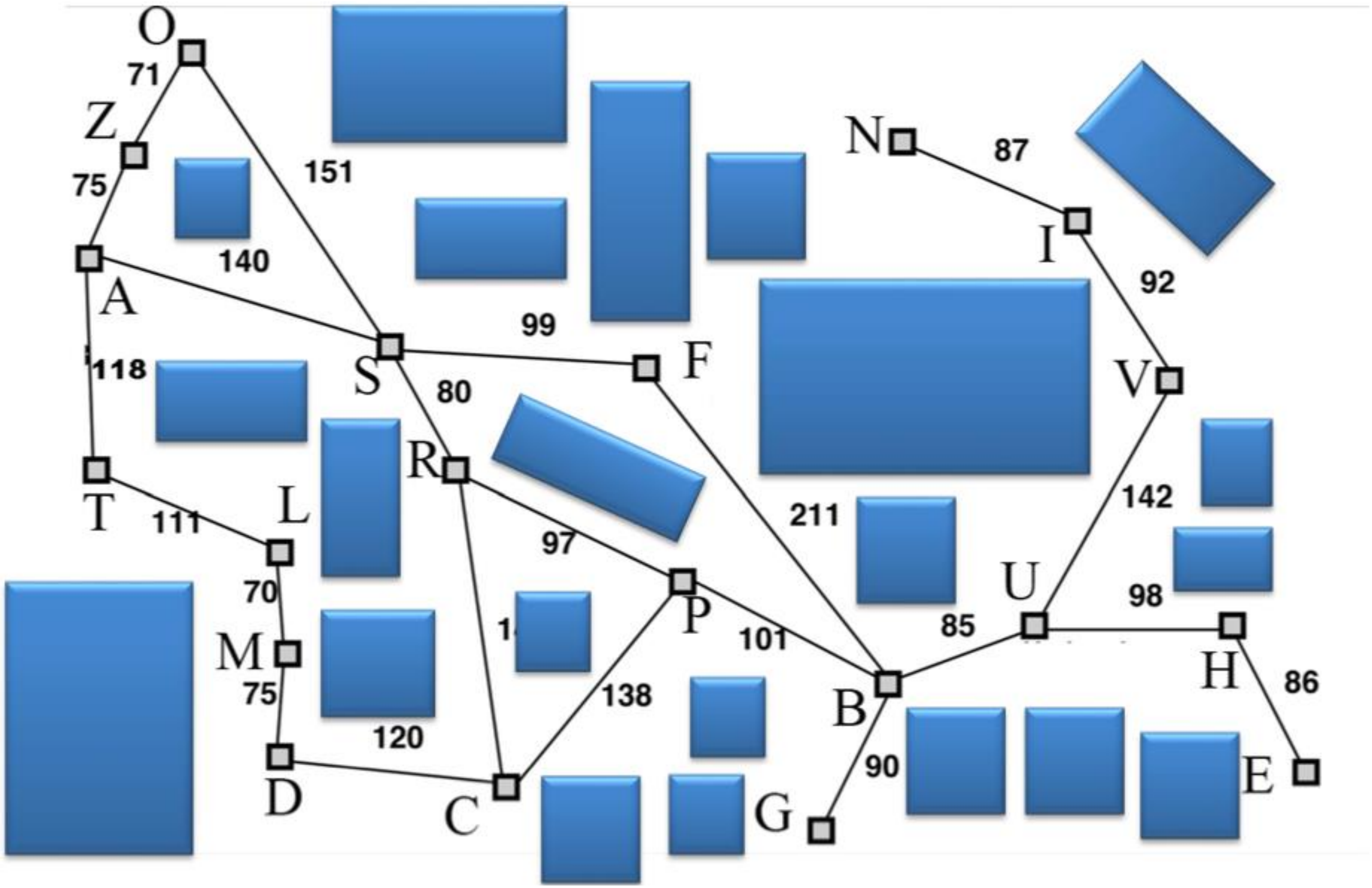
- Navigation / Pathfinding
- Navgraph: abstraction of all locations and their connections
- Cost / weight can represent terrain features (water, mud, hill), stealth (sound to traverse), etc
- What to do when …
  – Map features move
  – Map is continuous, or 100K+ nodes?
  – 3D spaces?

Grid as Graph



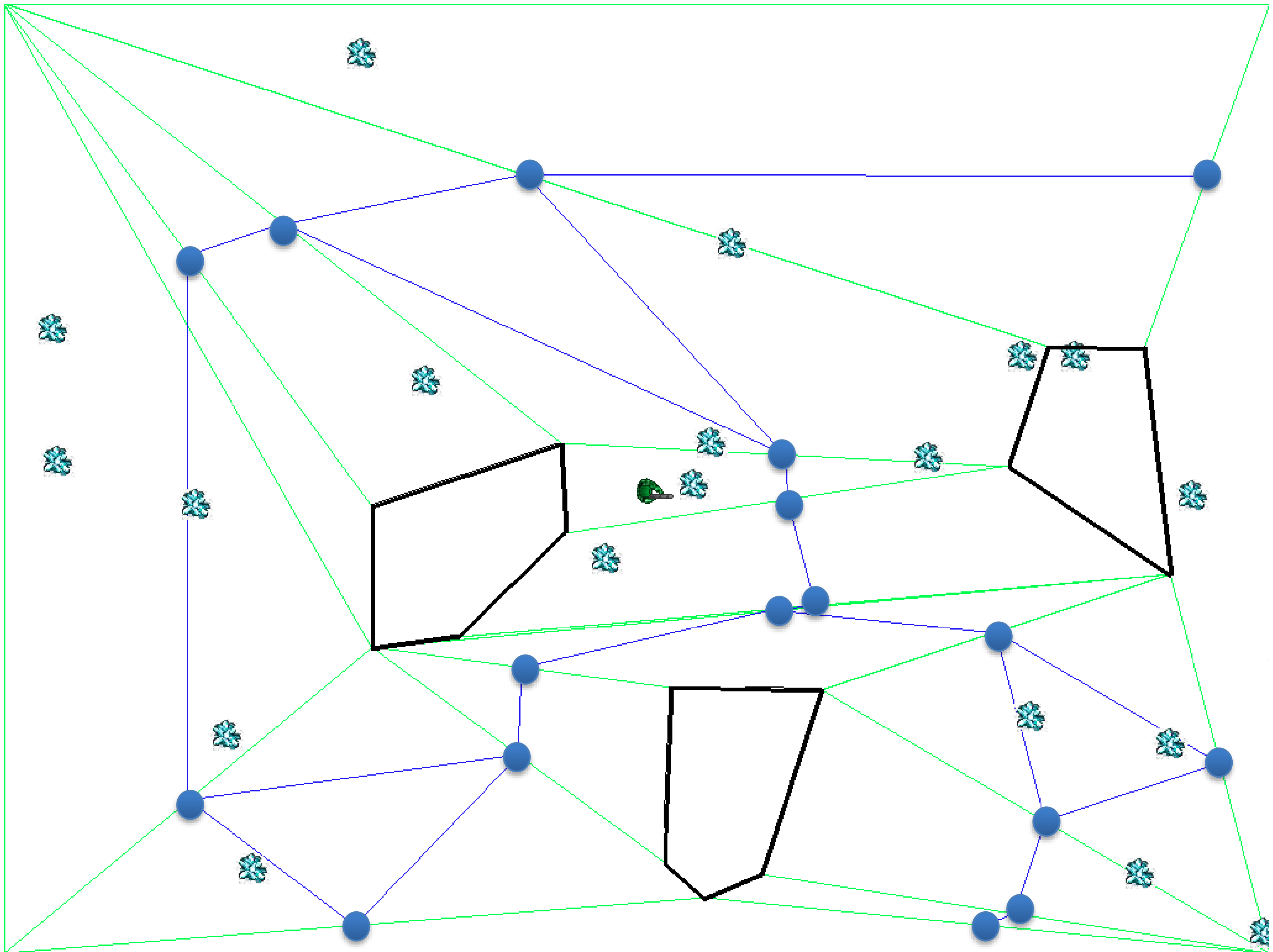


Path Network as Graph



Nav Mesh as Graph

(well actually path network again)

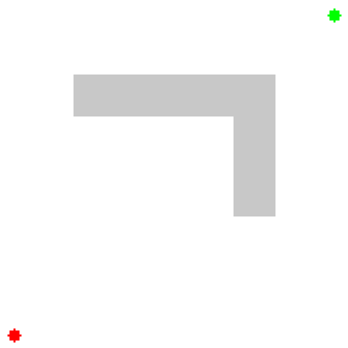


Why talk about these as graphs?

- Standard, abstract way to discuss different spatial representations
- Allows for quantifiable comparison between different spatial representations (e.g. number of edges/nodes)
- Allows us to discuss different search approaches without worrying about the exact spatial representation

Graph Search: Sorting Successors

- Uninformed (all nodes are same)
  - Greedy
  - DFS (stack – lifo), BFS (queue – fifo)
  - Iterative-deepening (Depth-limited)
- Informed (pick order of node expansion)
  - Dijkstra – guarantee shortest path ($E\log_2 N$)
  - Floyd-Warshall
  - A* (IDA*)…. Dijkstra + heuristic
  - D*
- Hierarchical can help

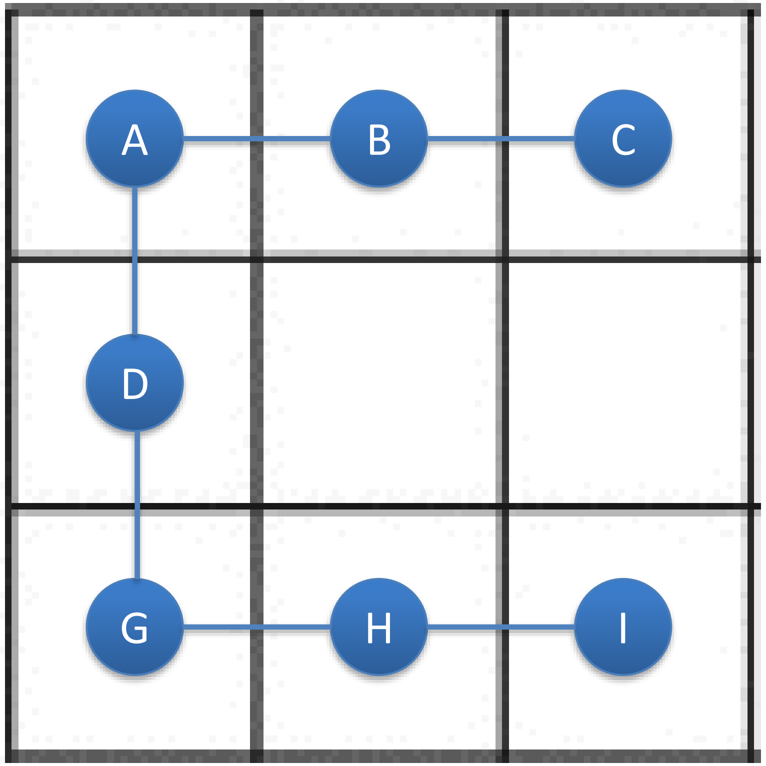


http://en.wikipedia.org/wiki/A*_search_algorithm

Greedy Algorithm Review

Find a path from start to goal node

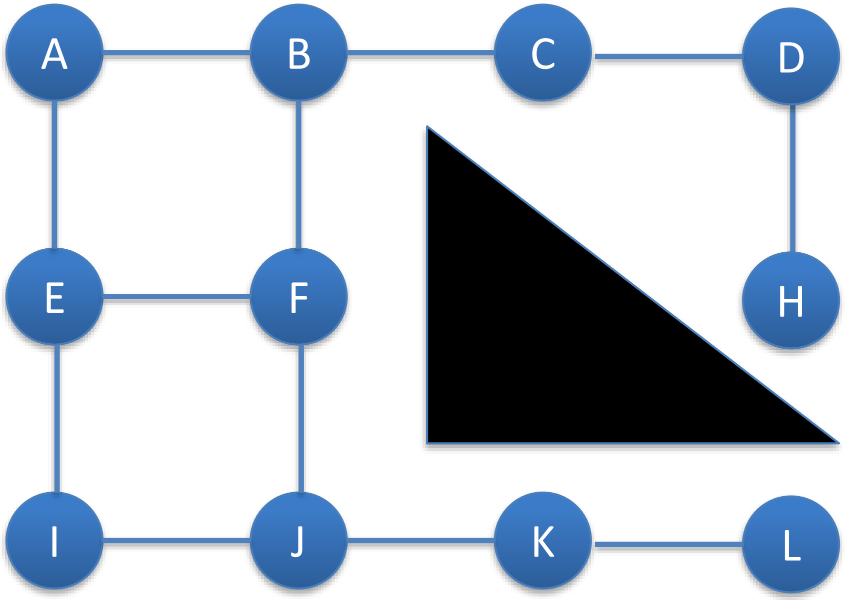
1. Add the neighbors of the current node to some open set list
   – We can get here!
2. Pick next current node from open set
3. If next node is goal, backtrack to start for path

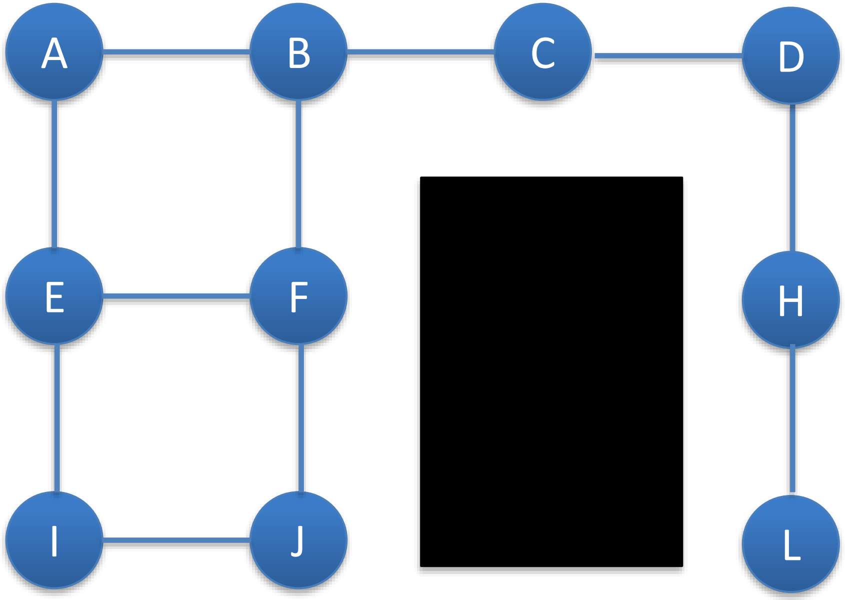


Question

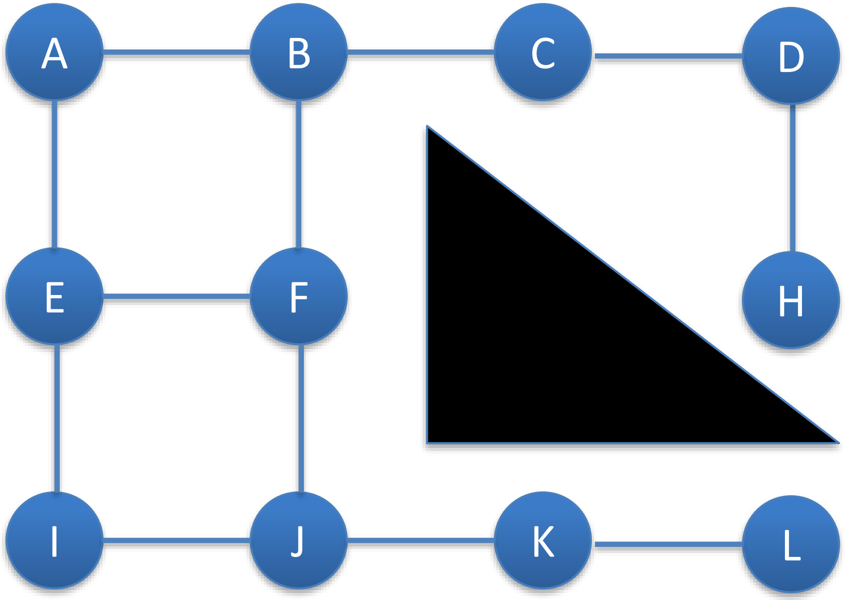
What heuristic could be used to get from B to L in both graphs the fastest?

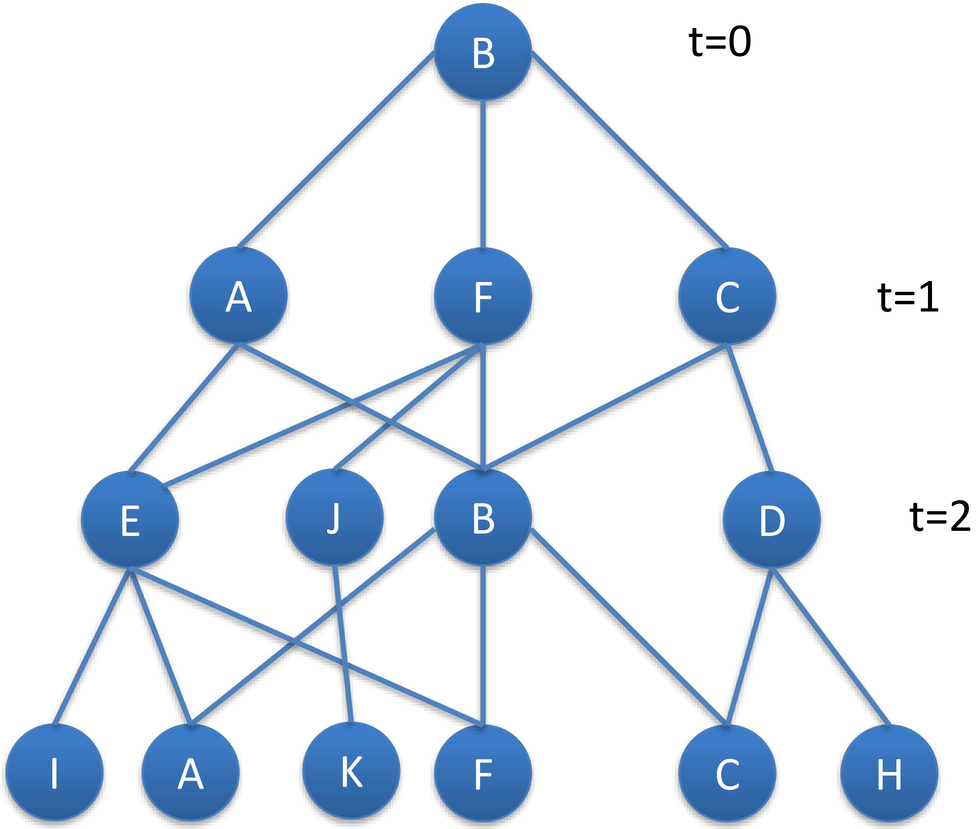
- Fastest meaning with fewest current nodes chosen





Greedy as a tree

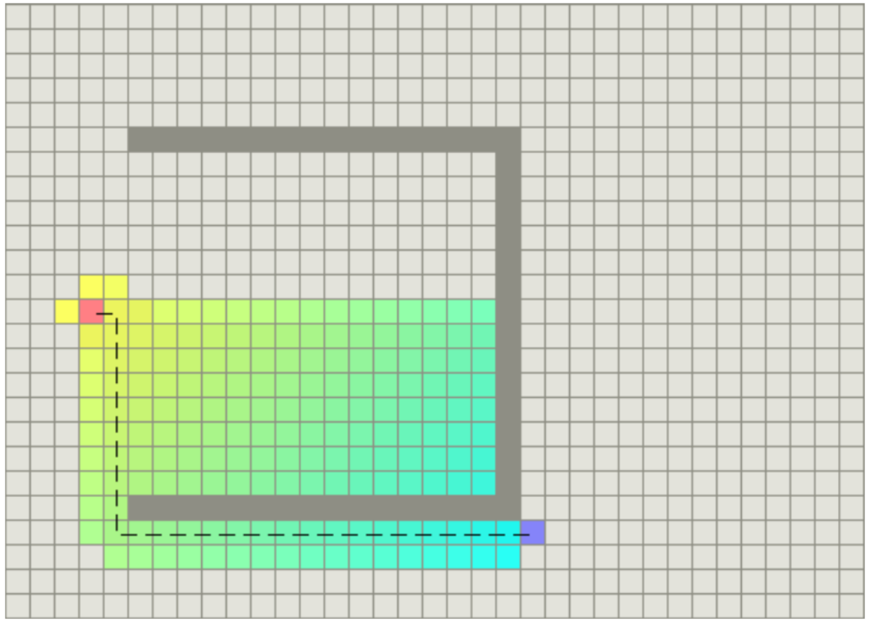




Improvement over Greedy

- Beyond improving the heuristic, how can we improve the greedy pathing algorithm?

- When does it fail?



A*

- Won’t just have an open set, but also a closed set (nodes already evaluated)
- Open set will be a priority queue, so if we discover a better node we can immediately pick it

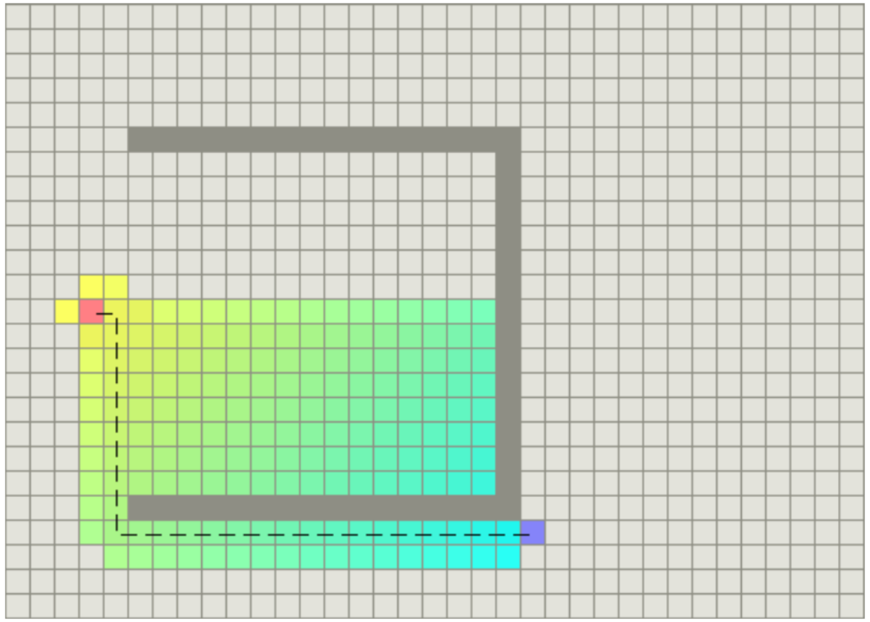
- Priority Queue: A queue that automatically sorts itself so minimum cost is at the top

A* Search

- 1968: Single source, single target graph search
- Guaranteed to return the optimal path if the heuristic is admissible
- Evaluate each state: $f(n) = g(n) + h(n)$
- Open list: nodes that are known and waiting to be visited
- Closed list: nodes that have been visited

A*

- Nodes will have two costs:
  - G score: Cost from getting from start to here
  - H score: Estimated cost of getting from here to goal
  - F score: G+H
- We will pick which node to choose next based on both of these scores

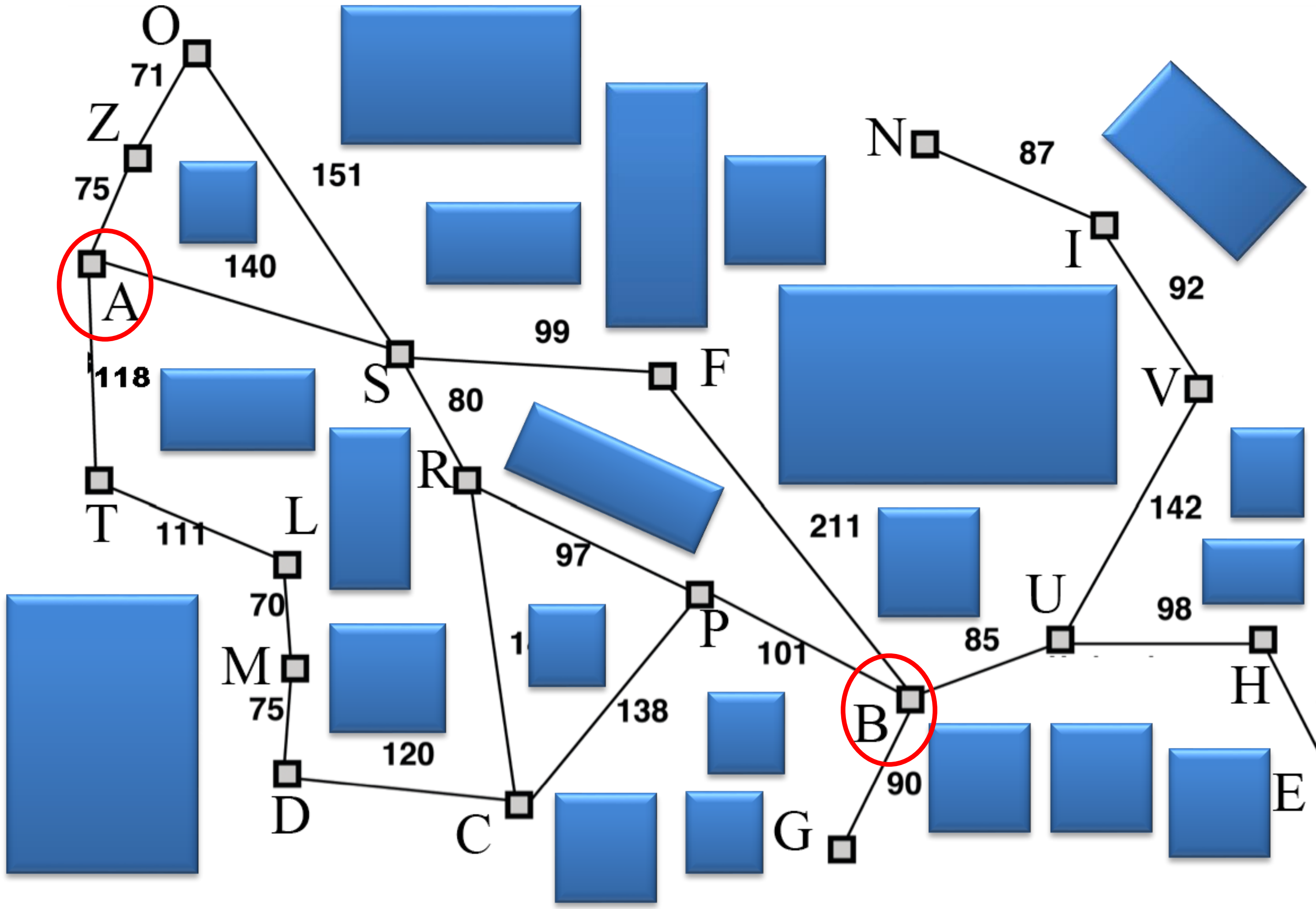


A*

```
add start to openSet
while openSet is not empty:
        current = openSet.pop()
        if current == goal:
                return reconstruct_path(current)
        closedSet.Add(current)
        for each neighbor of current:
                if neighbor in closedSet:
                        continue
                gScore = current.gScore + dist(current, neighbor)
                if neighbor not in openSet:
                        openSet.add(neighbor)
                else if gScore< openSet.get(neighbor).gScore
                        openSet.replace(openSet.get(neighbor), neighbor)
```

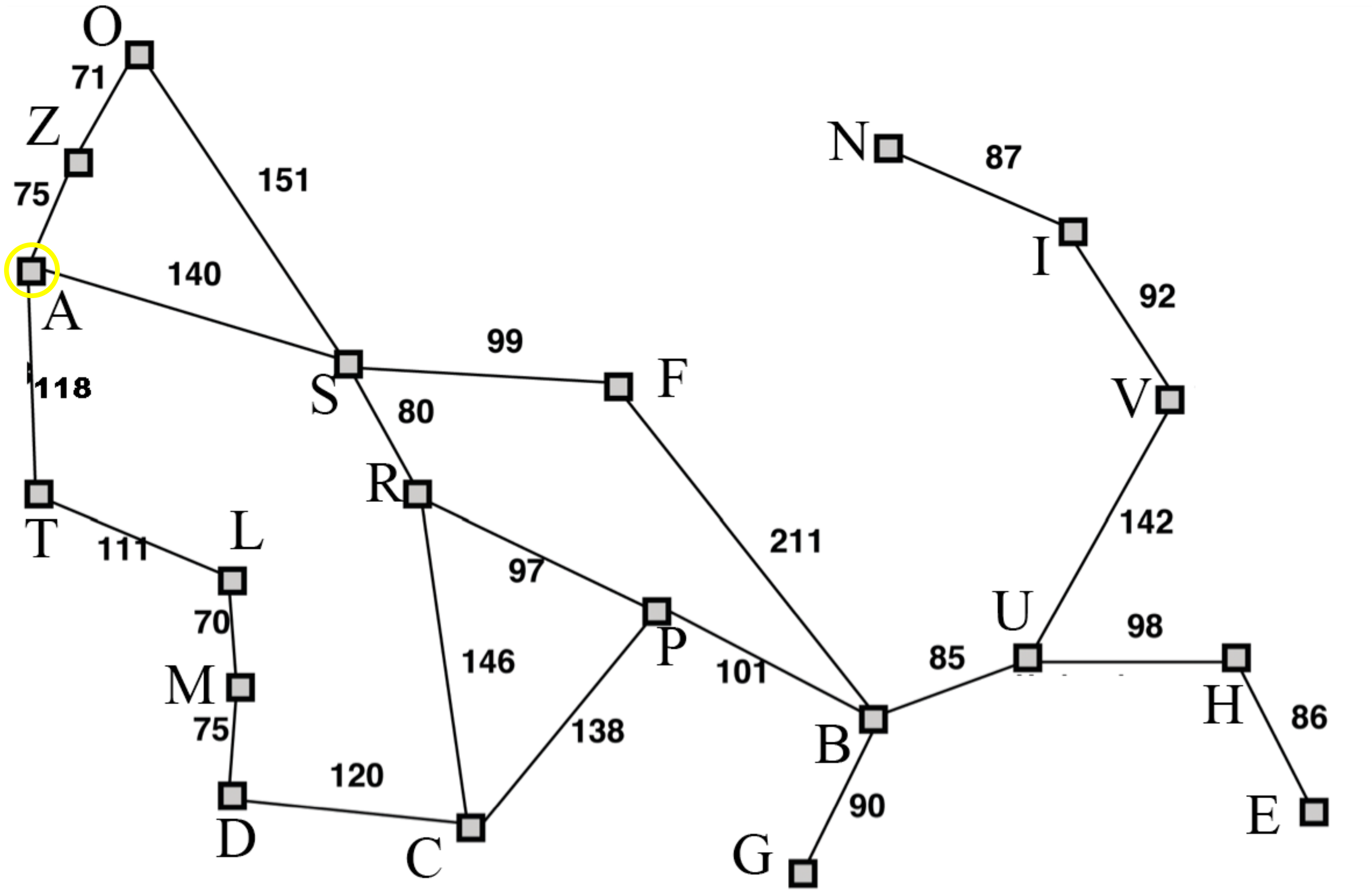
Heuristic Distance, A→B

A	366
B	0
C	160
D	242
E	161
F	176
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	100
S	253
T	329
U	80
V	199
Z	374



Evaluation function f(n) = g(n) + h(n)

A	366
B	0
C	160
D	242
E	161
F	176
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	100
S	253
T	329
U	80
V	199
Z	374

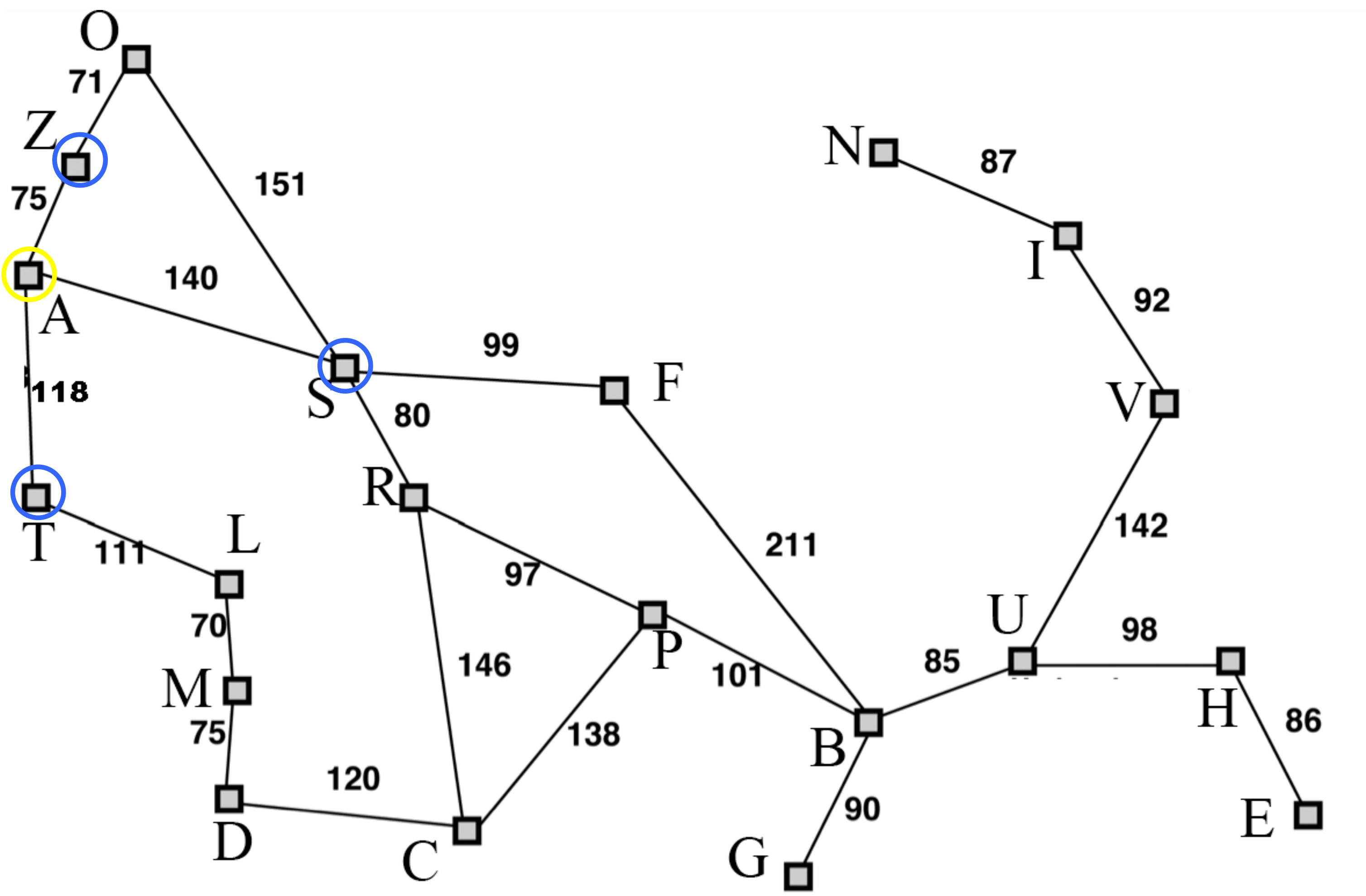


Open: A(366)

Closed:

Evaluation function f(n) = g(n) + h(n)

A	366
B	0
C	160
D	242
E	161
F	176
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	100
S	253
T	329
U	80
V	199
Z	374

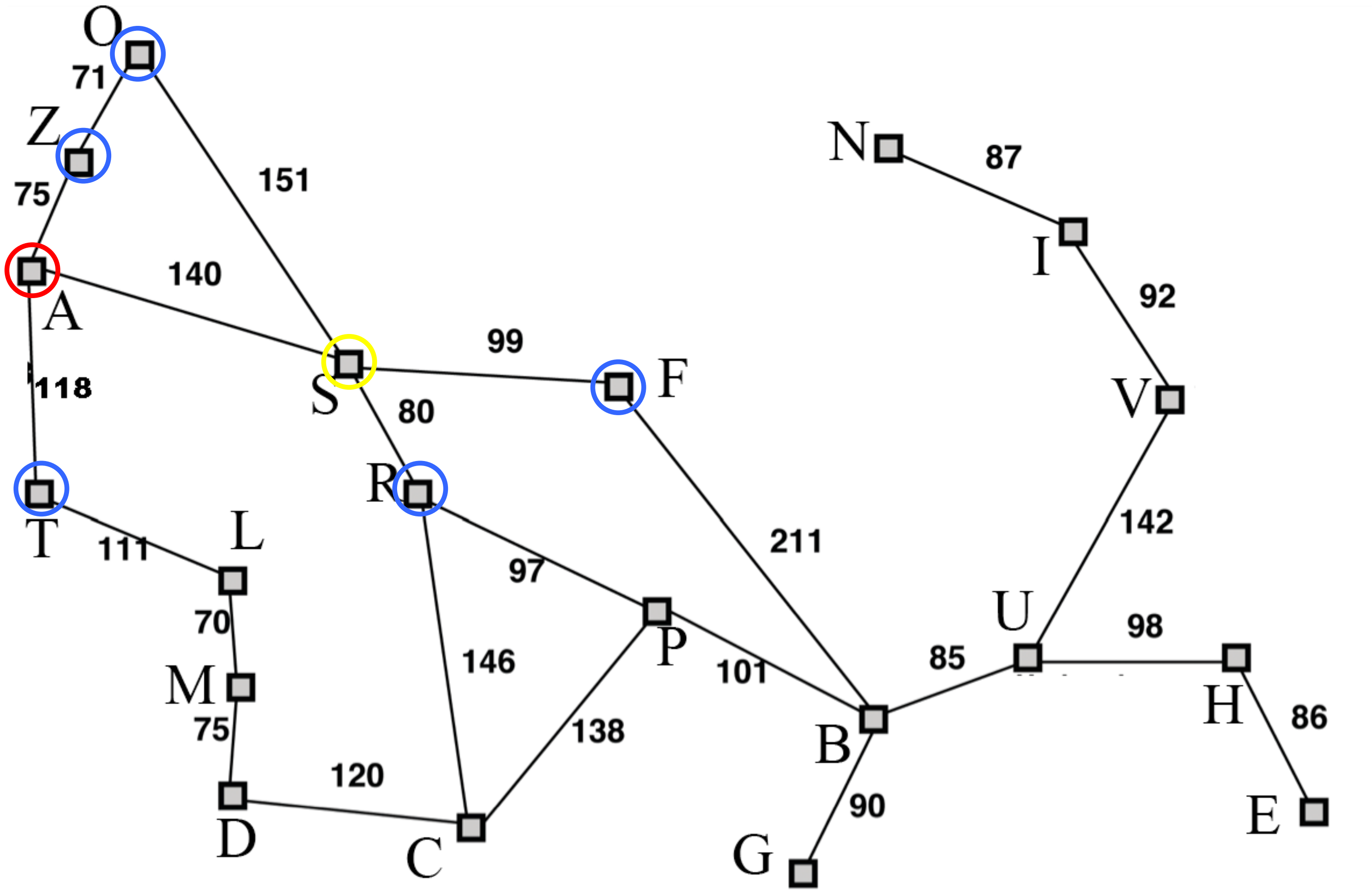


Open: S(253+140=393), T(329+118=447), Z(374+75=449)

Closed: A(366)

Evaluation function f(n) = g(n) + h(n)

A	366
B	0
C	160
D	242
E	161
F	176
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	100
S	253
T	329
U	80
V	199
Z	374

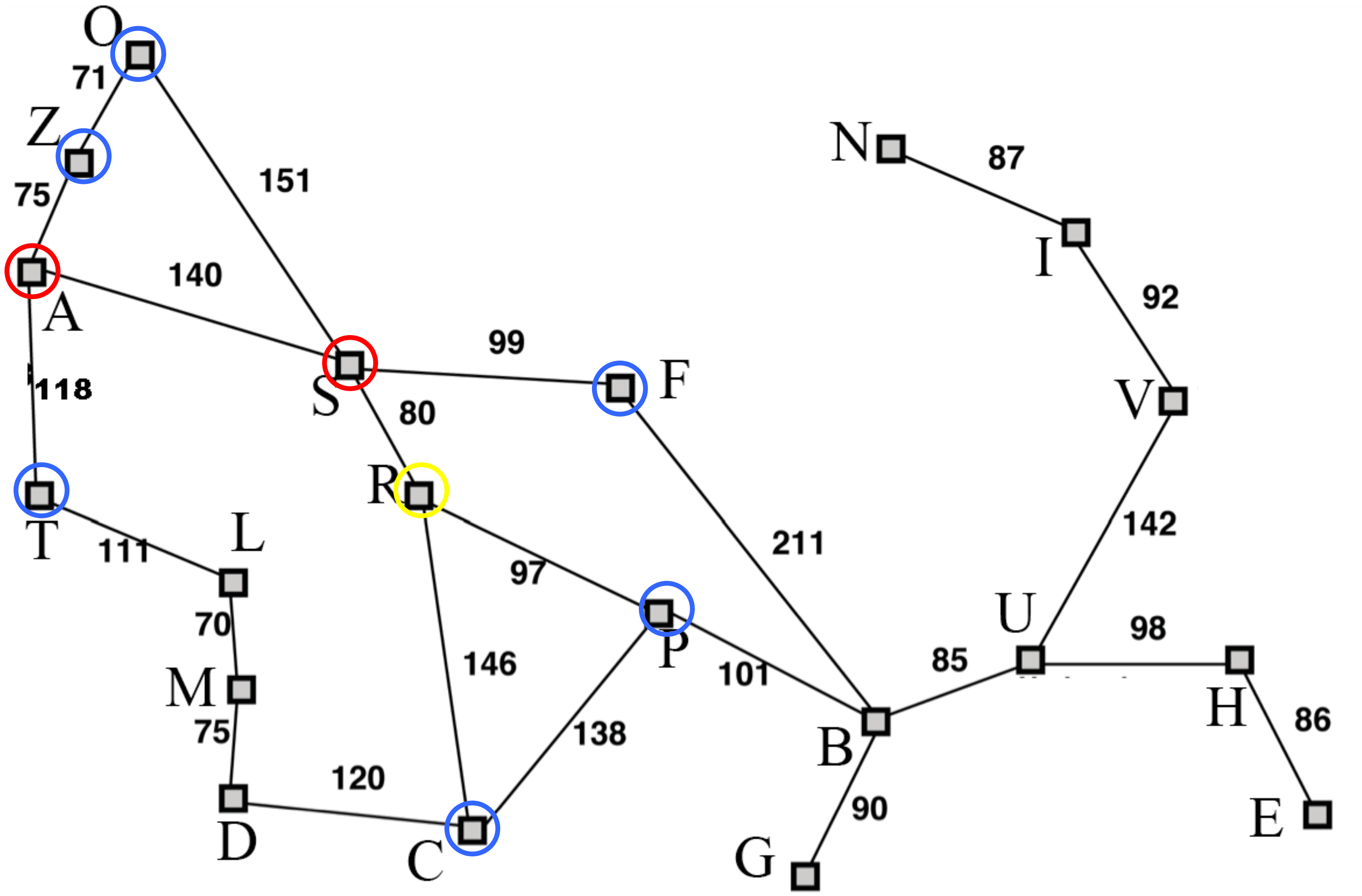


Open: R(220+193=413), F(239+176=415), T(329+118=447), Z(374+75=449), O(291+380=671)

Closed: S(393), A(366)

Evaluation function f(n) = g(n) + h(n)

A	366
B	0
C	160
D	242
E	161
F	176
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	100
S	253
T	329
U	80
V	199
Z	374

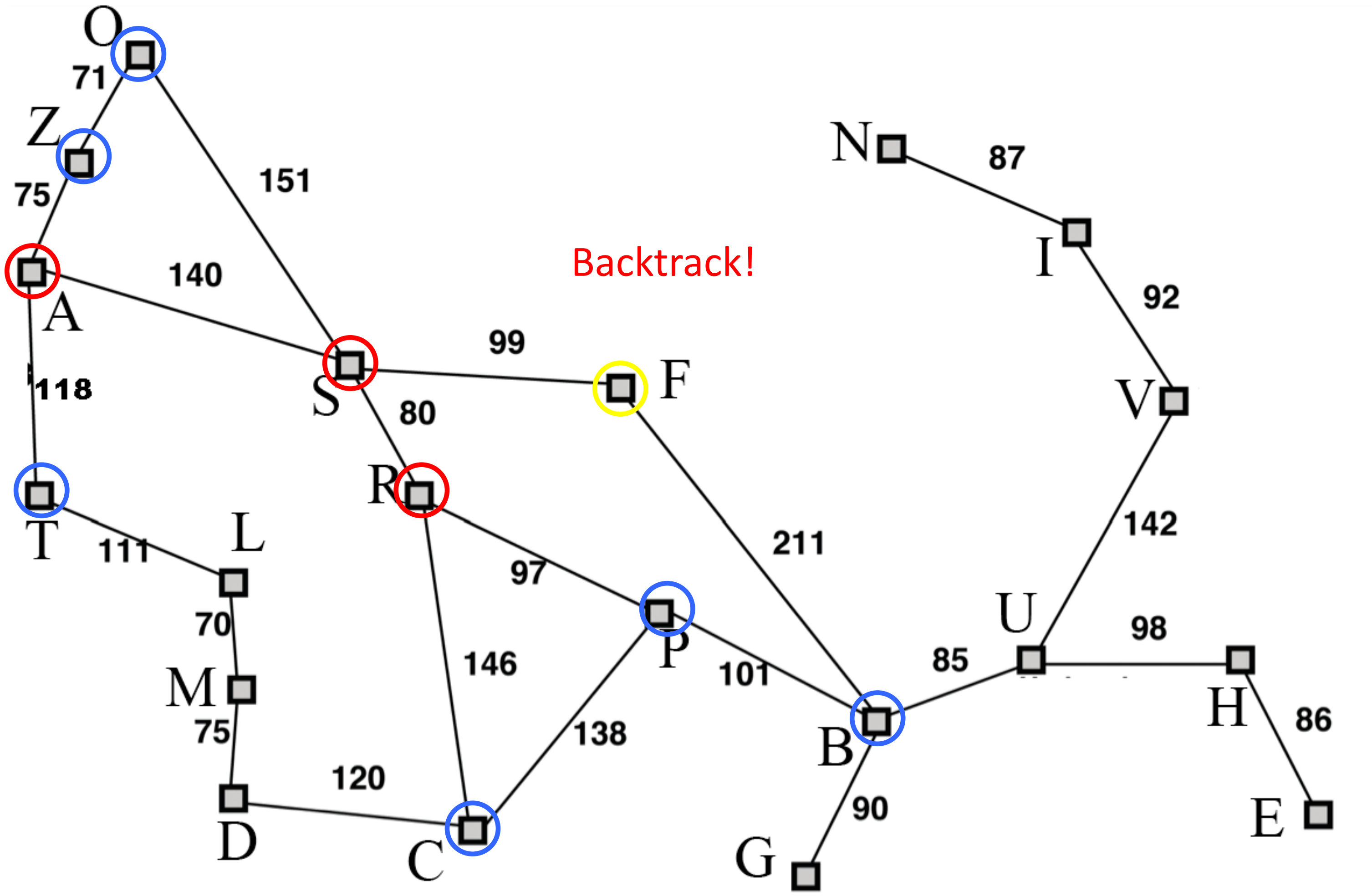


Open: F(239+176=415), P(317+100=417), T(329+118=447), Z(374+75=449), C(366+160=526), O(291+380=671)

Closed: R(413), S(393), A(366)

Evaluation function f(n) = g(n) + h(n)

A	366
B	0
C	160
D	242
E	161
F	176
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	100
S	253
T	329
U	80
V	199
Z	374

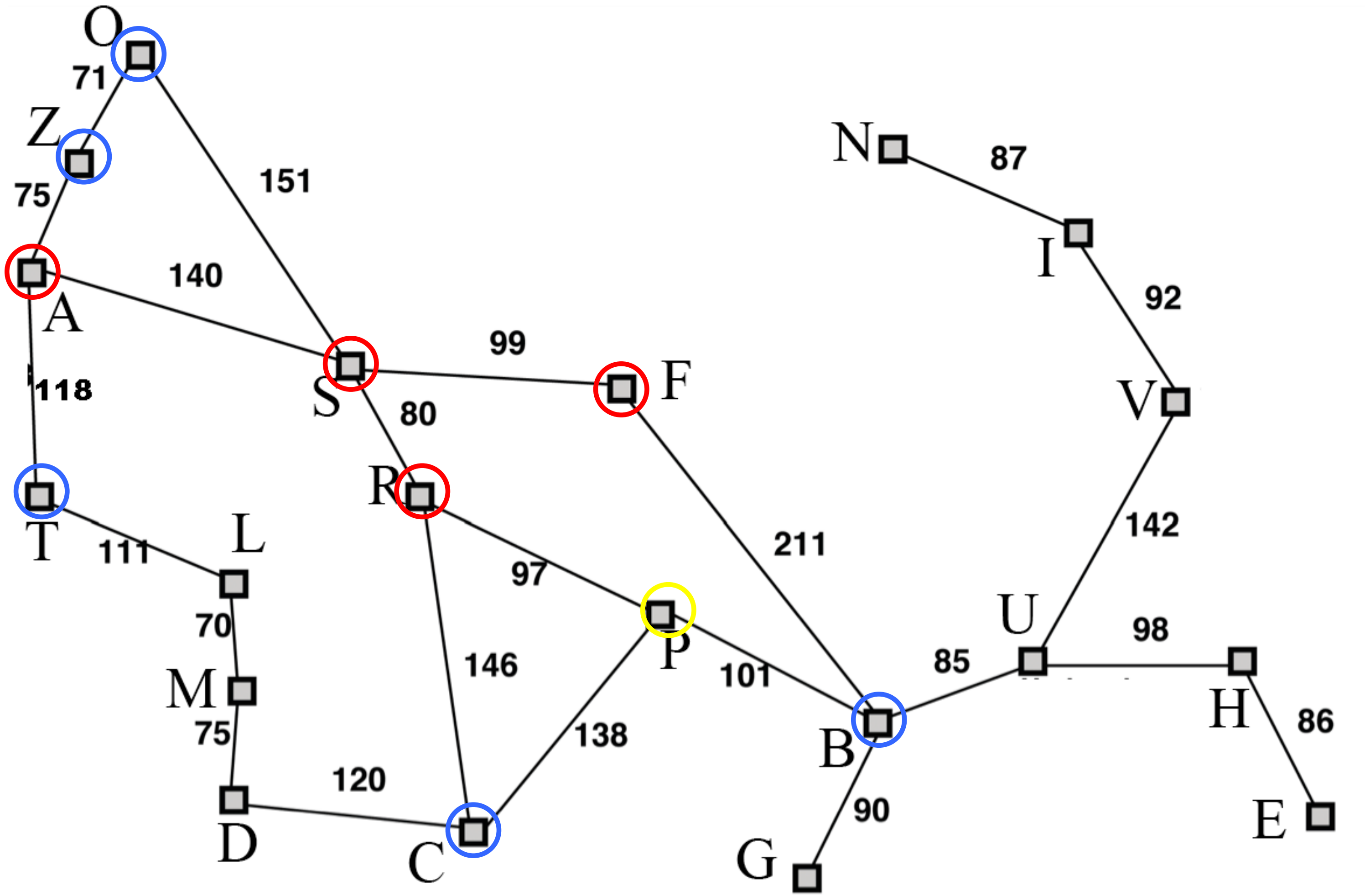


Open: P(317+100=417), T(329+118=447), Z(374+75=449), B(450+0=450), C(366+160=526), O(291+380=671)

Closed: F(415), R(413), S(393), A(366)

Evaluation function f(n) = g(n) + h(n)

A	366
B	0
C	160
D	242
E	161
F	176
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	100
S	253
T	329
U	80
V	199
Z	374

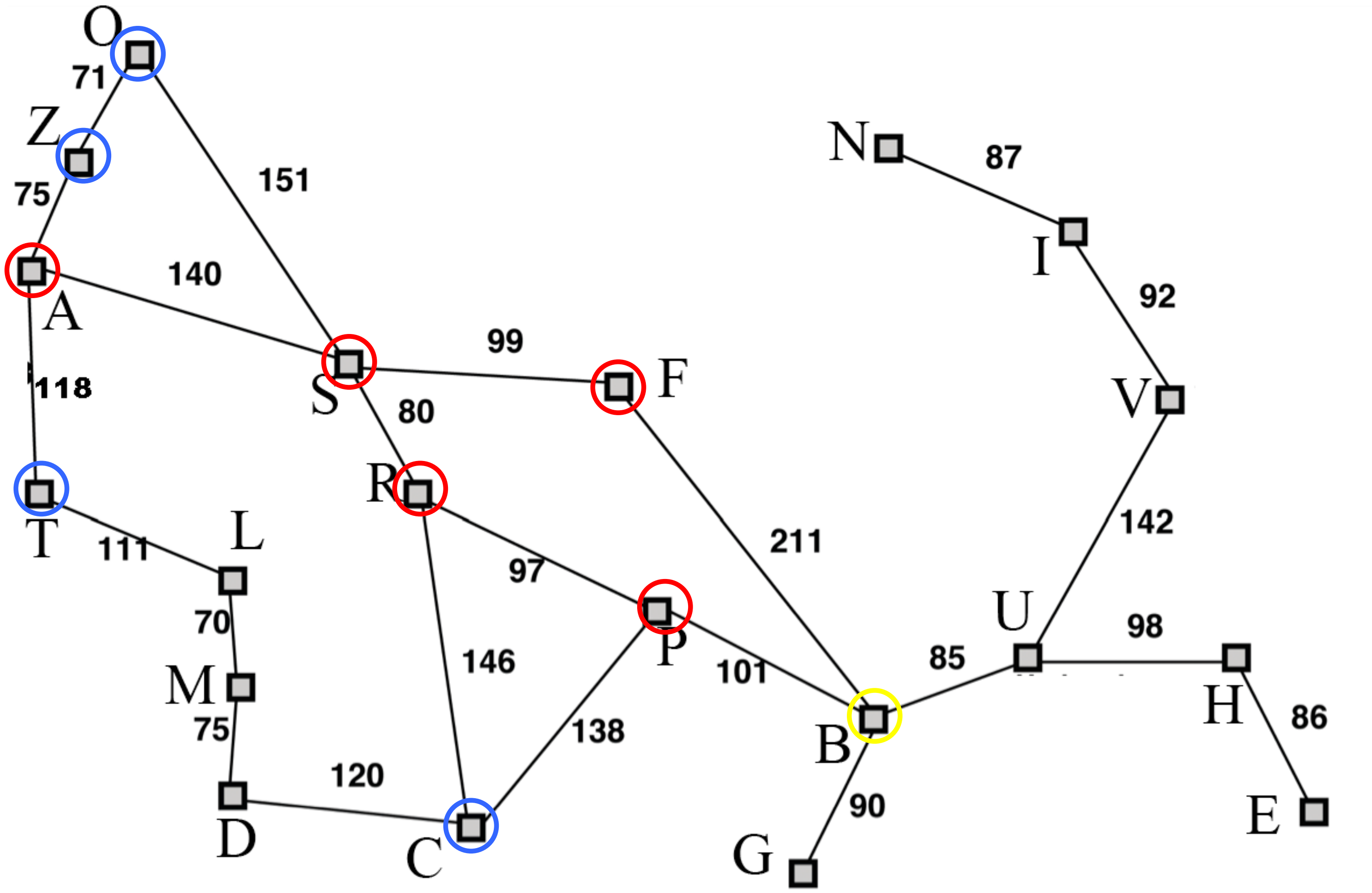


Open: B(418+0=418), T(329+118=447), Z(374+75=449), C(366+160=526), O(291+380=671)

Closed: P(417), F(415), R(413), S(393), A(366)

Evaluation function f(n) = g(n) + h(n)

A	366
B	0
C	160
D	242
E	161
F	176
G	77
H	151
I	226
L	244
M	241
N	234
O	380
P	100
S	253
T	329
U	80
V	199
Z	374



Solution: A-S-R-P-B

Open: T(329+118=447), Z(374+75=449), C(366+160=526), O(291+380=671)

Closed: B(418), P(417), F(415), R(413), S(393), A(366)

A* Search

- A* is optimal…
- …but only if you use an **admissible** heuristic
- An admissible heuristic is mathematically guaranteed to underestimate the cost of reaching a goal
- What is an admissible heuristic for path finding on a path network?