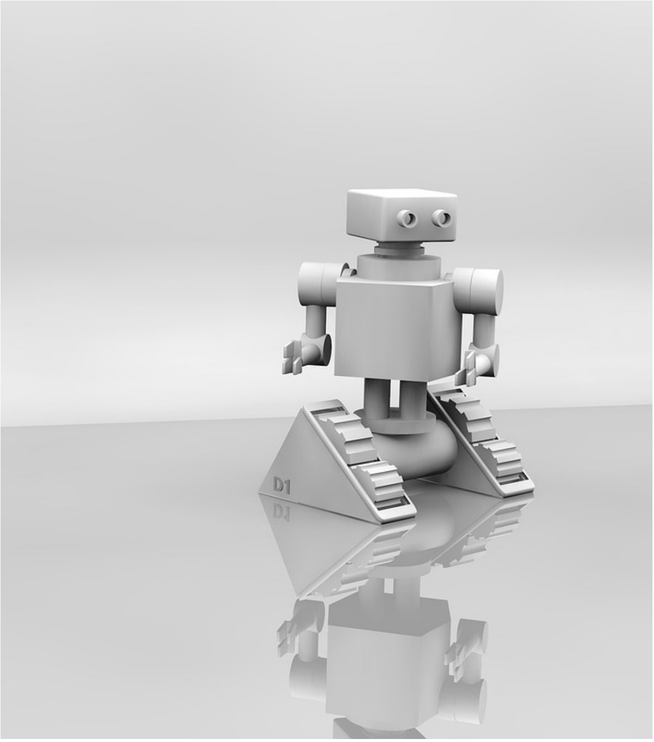


CPSC 481
Artificial Intelligence

Dr. Mira Kim
Mira.kim@fullerton.edu

What we will cover today

- Uninformed search
  - Iterative deepening
  - Uniform cost search

Textbook: Chapter 3.4

Review of DFS and BFS

- DFS
  - Advantages
    - Space complexity – less memory needed compared to BFS (linear vs exponential)
  - Disadvantages
    - Not complete – does not guarantee to find a solution even if there is one
    - Not optimal – does not guarantee to find least-cost path
    - Infinite loops – can get stuck in infinite loops

Review of DFS and BFS

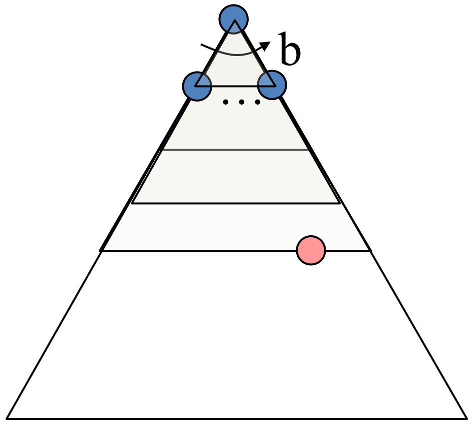
- BFS
  - Advantages
    - Optimality – guaranteed to find shortest path (in unweighted graphs)
  - Disadvantages
    - Space complexity – needs to keep track of all the nodes in the frontier and explored set

Depth limited DFS

- DFS is not complete for infinite spaces
- How can we adapt DFS for infinite spaces?
- Limit depth to
- Assumes: maximum depth of solution is known
- DFS but treat nodes at depth as if they have no children
- Note:
  - Depth limited DFS fails if we incorrectly set
- Time complexity:
- Space complexity:

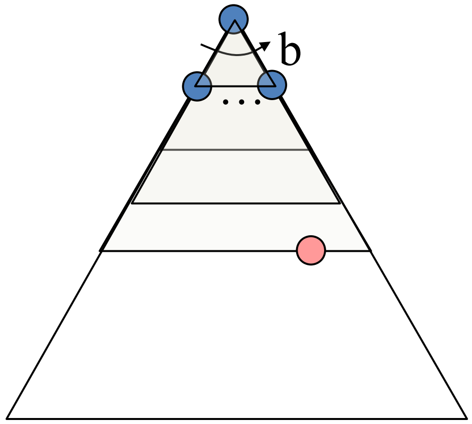
Iterative Deepening

- What is the biggest advantage of DFS?
- What is the biggest advantage of BFS?
- Idea: get the best of both worlds
  - DFS's space advantage with BFS's time / shallow-solution advantages
  - Run a DFS with depth limit 1. If no solution…
  - Run a DFS with depth limit 2. If no solution…
  - Run a DFS with depth limit 3. …..
- Most of the nodes are on the bottom of the search tree, it not a big waste to iteratively re-generate the top



Complexity of Iterative Deepening

- Time complexity:
  - Same as BFS!
- Space complexity: $O(bm)$
  - Linear - has the space complexity benefits of DFS



Some redundancy but the benefits (completeness and memory efficiency) are worth it!

In general, iterative deepening is ***the preferred uninformed search method*** *when the search space is large and the depth of the solution is not known.*

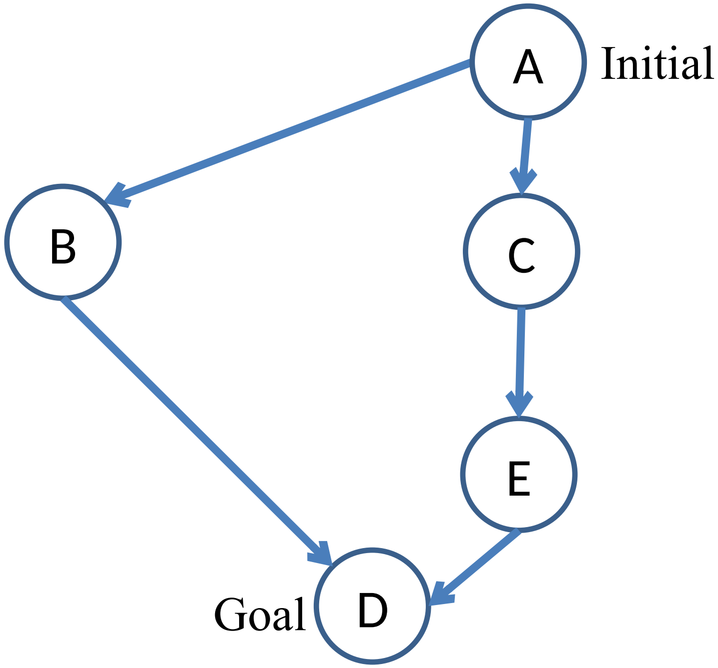
Time vs Space complexity

Depth	Nodes	Time	Memory
2	110	0.11ms	107KB
4	11,110	11ms	10.6MB
6		1.1s	1GB
8		2 minutes	103GB
10		3 hours	10TB
12		13 days	1PB
14		3.5 years	99PB
16		350 years	10 exabytes

b=10, computer generates 1 million nodes/second, 1KB per node in memory

Space requirement is a much more serious issue than time requirement

Cost-Sensitive Search

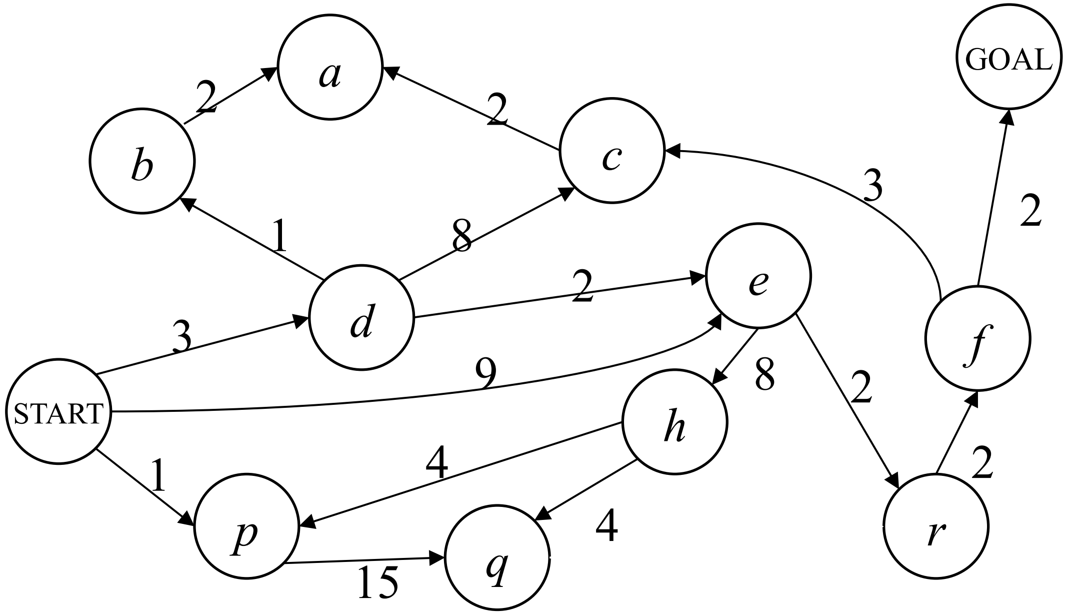


- BFS finds the shortest path in terms of number of actions
- It does not consider the length/cost of an action
- Cost: total cost of all actions on the path
- Can modify BFS to find the least-cost path: uniform cost search or Dijkstra's algorithm

Uniform Cost Search/Dijkstra's

Activity: List nodes in increasing path cost

Path Cost: length of shortest path from START node

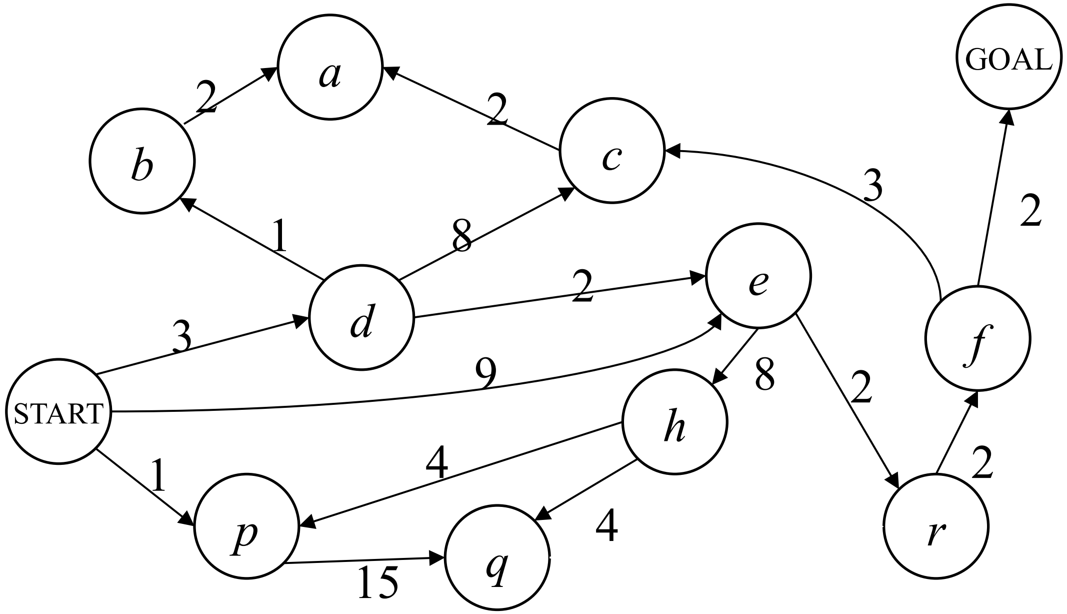


Uniform Cost Search

Strategy: expand the "cheapest" node first

Frontier is a priority queue

priority = cumulative cost



Uniform cost search

- Input: A problem
- Data structures:
  - Frontier (also called “open”)
    - **Priority Queue, ordered by path cost**
  - Explored (also called “closed”)
    - Set, for efficiency

```
Initialize frontier with initial state
Initialize explored to empty
Loop do
   IF the frontier is empty RETURN FAILURE
   Choose lowest cost node from frontier and remove it
   IF lowest cost node is goal RETURN SUCCESS
   Add node to explored
   FOR every child node of node
     IF child not already on frontier or explored
       insert child node to frontier
     ELSE IF child is in frontier with higher path cost
       replace existing frontier node with child node
```

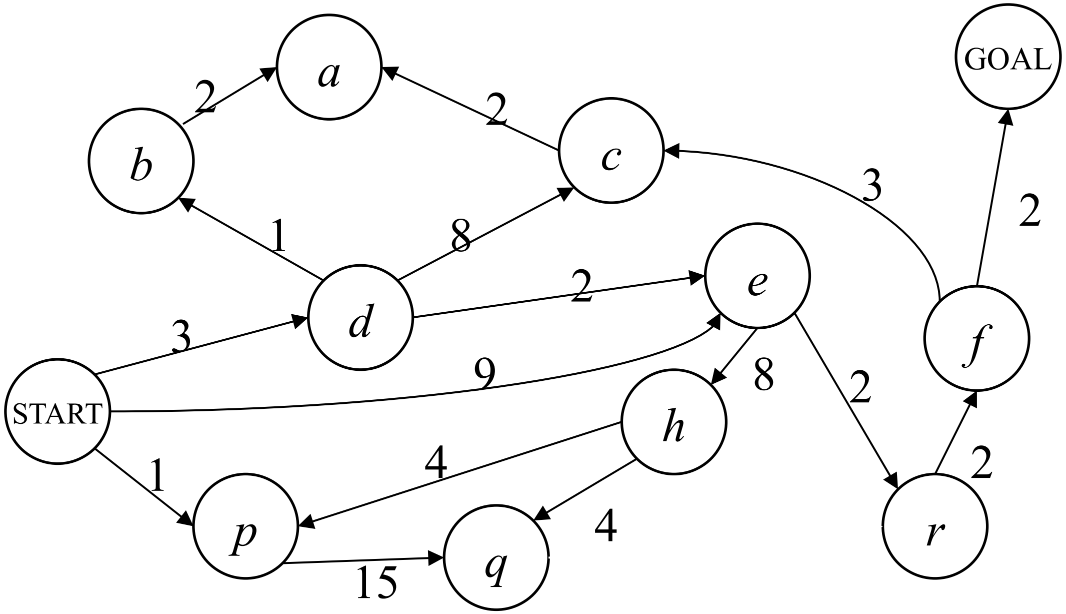
Uniform Cost Search

Strategy: expand the “cheapest” node first

Frontier is a priority queue

priority = cumulative cost

Explored	Frontier – Priority Queue
-	S(0)

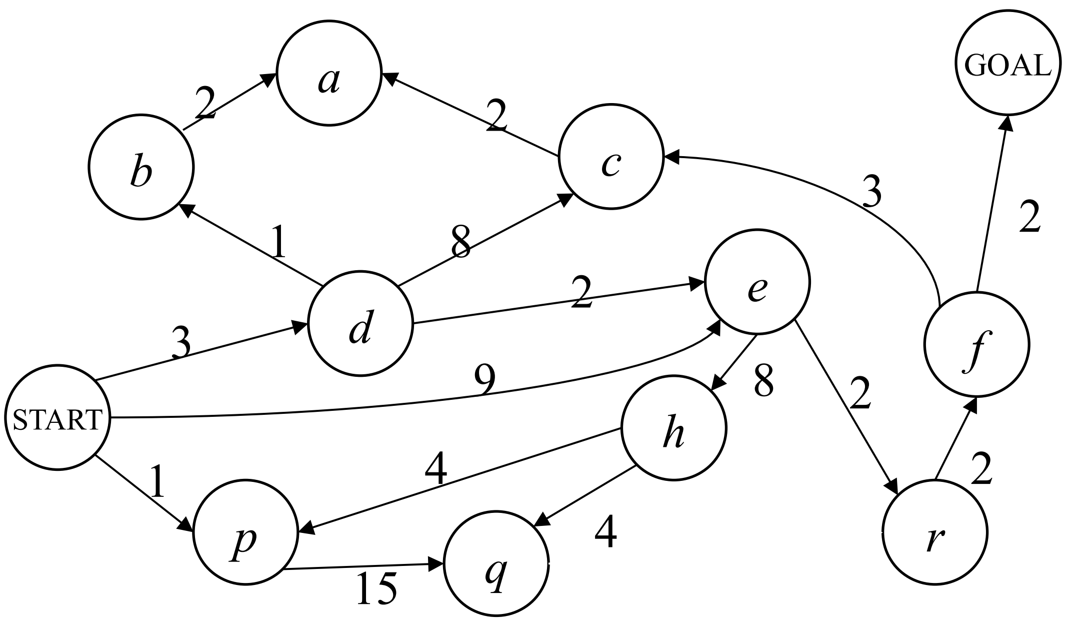


Uniform Cost Search

Strategy: expand the “cheapest” node first

Frontier is a priority queue

priority = cumulative cost



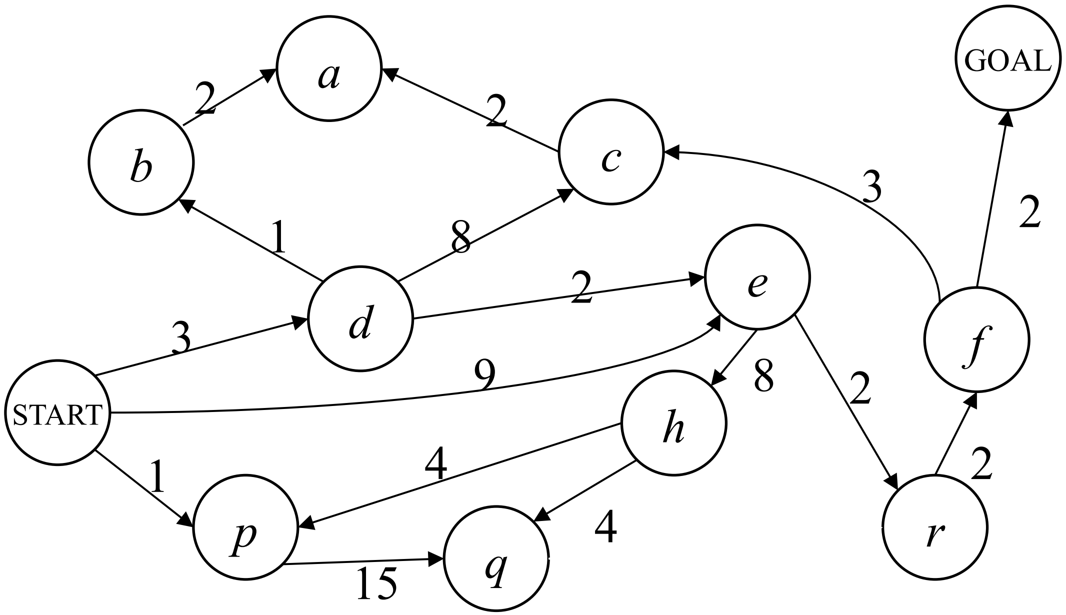
Explored	Frontier – Priority Queue
-	S(0)
S	p(1), d(3), e(9)

Uniform Cost Search

Strategy: expand the "cheapest" node first

Frontier is a priority queue

priority = cumulative cost



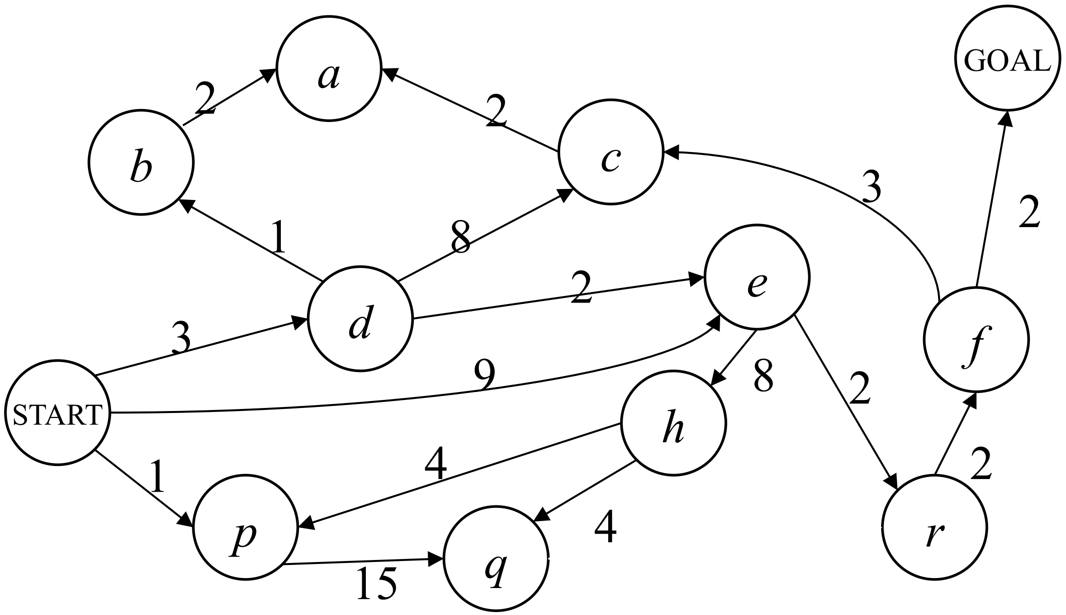
Explored	Frontier – Priority Queue
-	S(0)
S	p(1), d(3), e(9)
S, p	d(3), e(9), q(16)

Uniform Cost Search

Strategy: expand the “cheapest” node first

Frontier is a priority queue

priority = cumulative cost



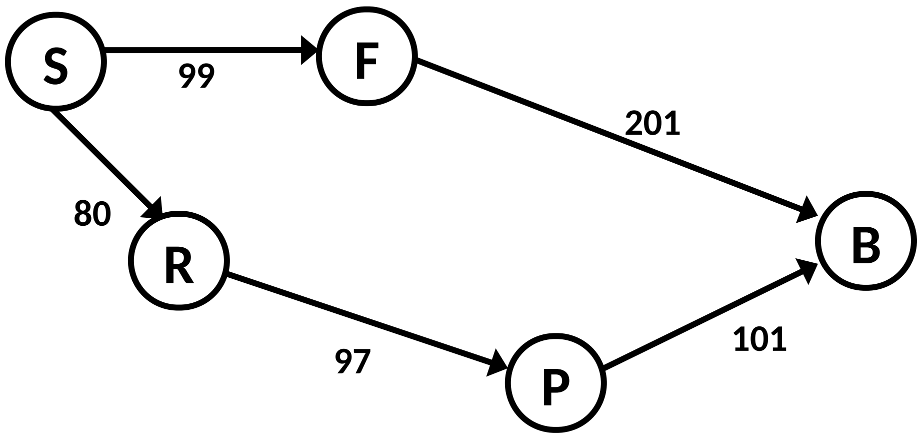
Explored	Frontier – Priority Queue
-	S(0)
S	p(1), d(3), e(9)
S, p	d(3), e(9), q(16)
S, p, d	e(5), q(16), b(4), c(11)

In-class exercise

- Work on the rest of the iterations for the graph
  - Show frontier (priority queue) and explored node in a table using Uniform Cost Search

When should uniform cost terminate?

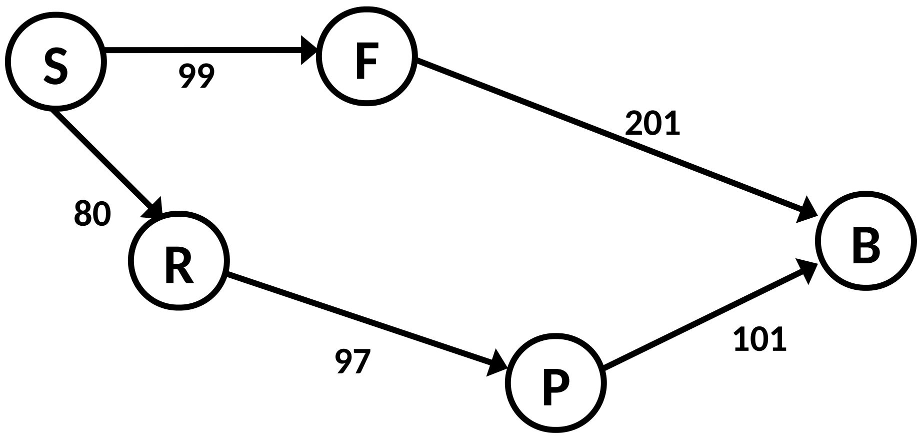
- Should we stop when we discover a goal node?

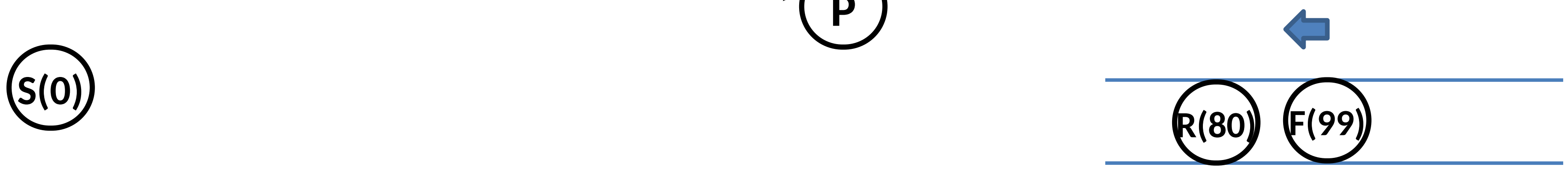


- No: only stop when we expand a goal node

When should uniform cost terminate?

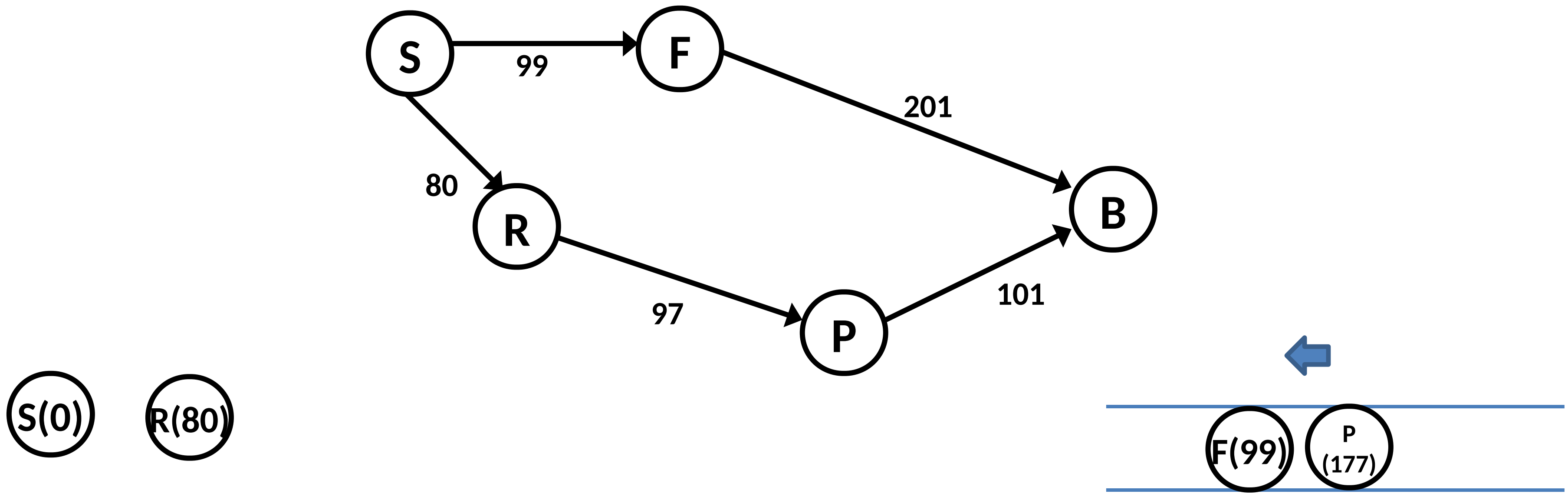
- Should we stop when we discover a goal node?





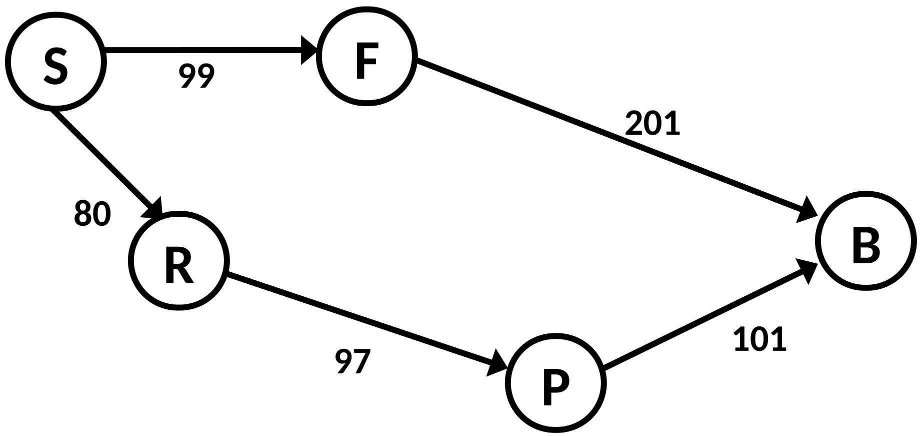
When should uniform cost terminate?

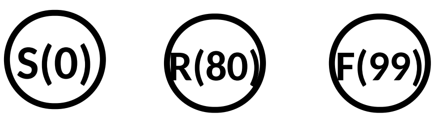
- Should we stop when we discover a goal node?

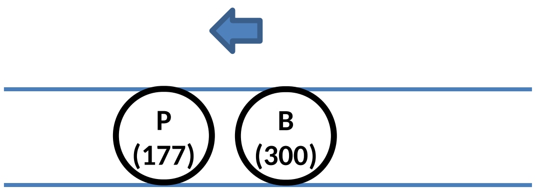


When should uniform cost terminate?

- Should we stop when we discover a goal node?

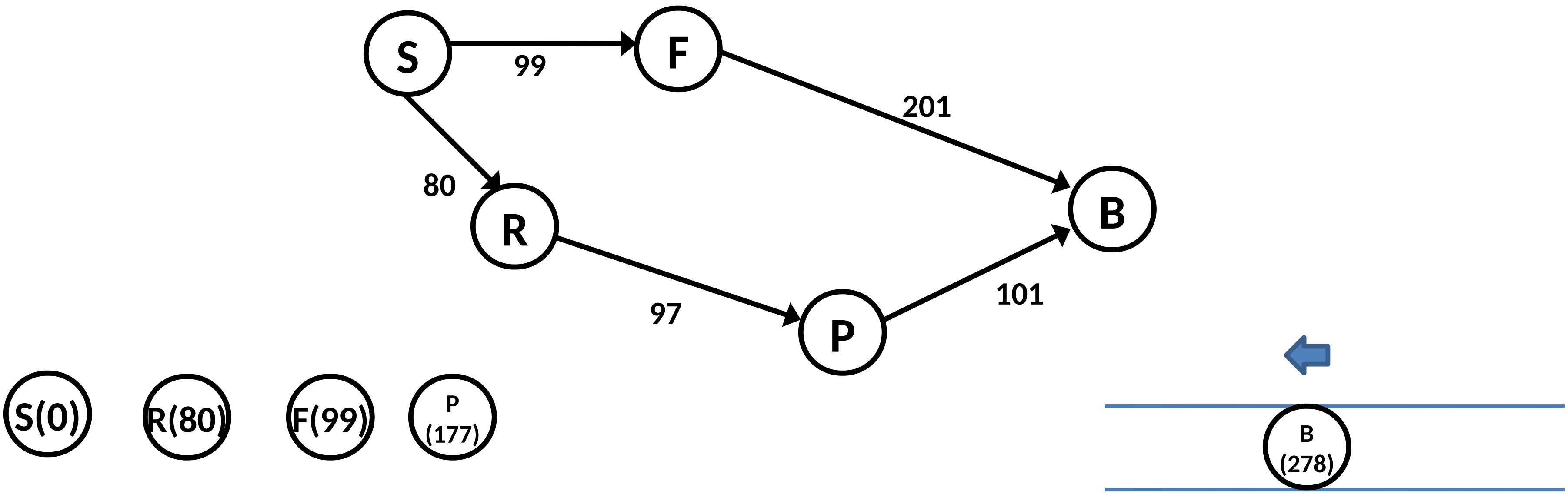






When should uniform cost terminate?

- Should we stop when we discover a goal node?



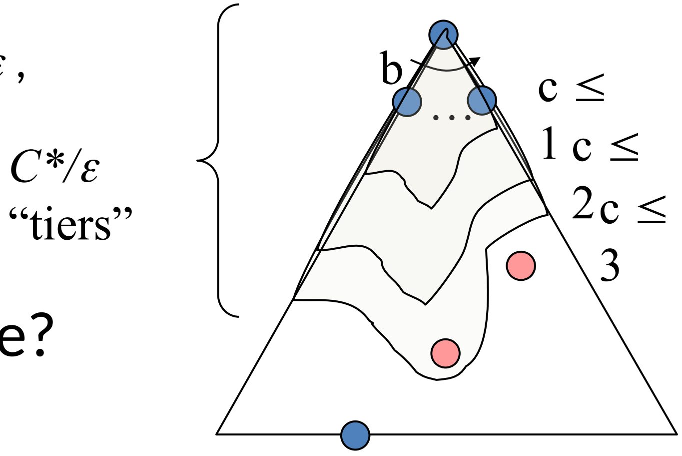
- No: only stop when we expand a goal node

Uniform cost search (UCS)

- Same as Dijsktra’s shortest paths algorithm
  - Dijsktra’s algorithm has no “goal”
  - Shortest paths to *all* nodes
- Implementations of Dijsktra’s algorithm assume all vertices and edges are known before-hand
  - In UCS, vertices and edges are discovered during the search
  - Works for infinite graphs also

Uniform Cost Search (UCS) Properties

- What nodes does UCS expand?
  - Processes all nodes with cost less than cheapest solution!
  - If that solution costs C^* and arcs cost at least ε , then the "effective depth" is roughly C^*/ε
  - Takes time $O(b^{C^*/\varepsilon})$
    - exponential in effective depth
- How much space does the frontier take?
  - Has roughly the last tier, so $O(b^{C^*/\varepsilon})$
- Is it complete?
  - Assuming best solution has a finite cost and minimum edge cost is positive, yes!
- Is it optimal?
  - Yes!

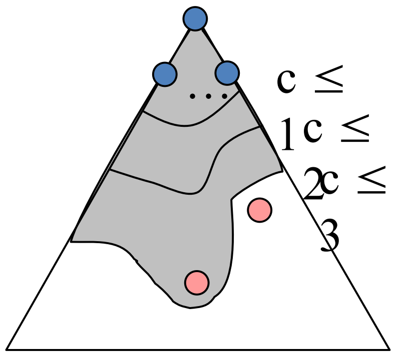


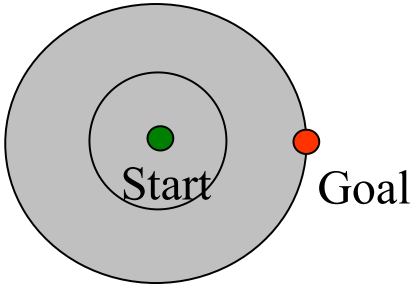
Uniform Cost Issues

- The good: UCS is complete and optimal!

- The bad:
  - Explores options in every “direction”
  - No information about goal location

- Solution: consider distance from goal too





Summary of uninformed search

- Breadth-first search (BFS) - Expand the shallowest node
- Depth-first search (DFS) - Expand the deepest node
- Depth-limited search (DLS) - Depth first search with a depth limit
- Iterative-deepening search (IDS) - DLS with increasing limit
- Uniform-cost search (UCS) - Expand the least cost node

Acknowledgement

- https://inst.eecs.berkeley.edu/~cs188/su20/

References

- Russel and Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Prentice Hall, 2010.