CS3243 Tutorial Group 7

Tutorial 2 – Uninformed Search

Expectations

- Before the tutorial:
  - You review the lecture notes
  - You attempt all tutorial questions and submit the assignment solution
    - **if you didn't try a question, we won't go through it during the tutorial**
- During the tutorial:
  - I will not spoon-feed solutions
  - You ask questions when you are confused or don't understand something
    - the purpose is to learn, not to seem "smart"
  - You participate: ask questions, offer answers, volunteer to present a solution

Quick recap

- Solve problems by searching for solutions (path from initial to goal)

- Formal definition of a search problem:
  - states, initial state, actions, transition model, goal test, path cost (*g*)

- Tree search vs graph search (does *not* revisit nodes)
  - node expansion, frontier, explored set
  - frontier implemented as a queue: FIFO, LIFO or priority
  - algorithms judged by **completeness**, **optimality**, time & space **complexity**

- BFS (FIFO)
- UCS (priority queue; pick lowest *g*)
  - goal test when node selected for expansion, rather than when added to frontier
  - if find better path cost to node on frontier, replace it
- DFS (LIFO), DLS (depth limited), IDS (iterative deepening)
- Bidirectional search

Criterion	BFS	UCS	DFS	DLS	IDS	Bidirectional
Complete?	Yes (if b is finite)	Yes (if all step costs are $\geq \varepsilon$)	No (infinite depth graphs)	No (if $l < d$)	Yes (if b is finite)	Yes (if b is finite & both directions use BFS)
Optimal?	No (unless uniform step costs)	Yes	No	No	No (unless uniform step costs)	No (unless uniform step costs & BFS)
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Space	Max size of frontier: $O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(bm)$ (can be $O(m)$ if actions are reversible)	$O(bl)$ (can be $O(l)$ if actions are reversible)	$O(bd)$ (can be $O(d)$ if actions are reversible)	$O(b^{d/2})$

b = branching factor
d = depth of the *shallowest* solution

m = maximum depth of the search tree
l = depth limit

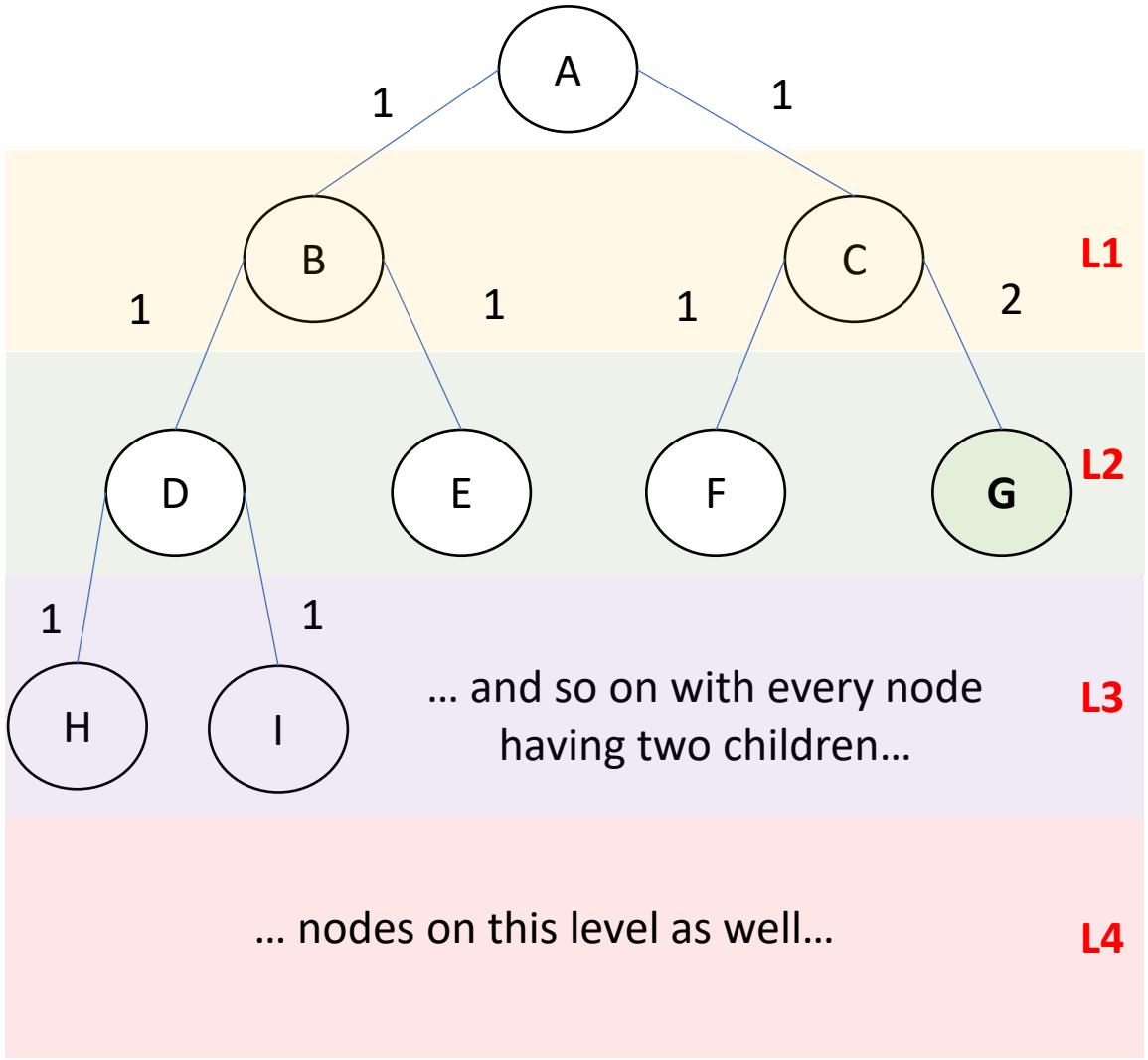
Simple proof that UCS optimal (step costs $\geq \varepsilon$)

- We use these invariants of UCS, which follow directly from the construction of the algorithm:
  1. The frontier expands by exploring all edges out of the initial node, i.e. it forms a perimeter around it.
  2. Since all step costs $\geq \varepsilon$, $g(n)$ is monotonically non-decreasing along any path.
  3. When a node n is chosen for expansion, $g(n)$ is $\leq$ the cost of any other nodes in the frontier.
- Together, these invariants imply the property P:

  whenever a node n is chosen for expansion, $\mathrm{g}(n)$ is the shortest path cost from the initial node to n
- Proof (by contradiction):
  - n is chosen for expansion.
  - Suppose there exists a path from the initial node to n that has lower cost than $\mathrm{g}(n)$.
  - By inv. 1, the only way to find such a path is by expanding one of the nodes in the frontier (expanding the perimeter).
  - But by inv. 3, all nodes q in the frontier have $g(q) \leq g(n)$, since n was selected for expansion.
  - However, by inv 2., all paths from any node q have cost at least $\mathrm{g}(q) + \varepsilon$, which is strictly greater than $g(n)$. Contradiction.
- By P, when a goal node is expanded, it has the lowest path cost
  - i.e. UCS is optimal

Optional: read *Position Paper: Dijkstra's Algorithm versus Uniform Cost Search or a Case Against Dijkstra's Algorithm*

Why does UCS have $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$ complexity?



- A is the initial node and G is the only goal node.
- $C^* = 3$ is the optimal path cost, $\varepsilon = 1$ is the minimum action cost. Branching factor $b = 2$.
- $\lfloor C^*/\varepsilon \rfloor$ is the worst-case for how "deep" you can go expanding nodes on bad paths until you are guaranteed to expand the goal. In our case, we have to expand all nodes up to 3 levels deep, i.e. we expand H, I and the children of E and F (and thus populate L4).
- *Complexity is given by how many nodes are generated (added to the frontier)* = how many times you evaluate the transition function.
- Q: Where does the + 1 come from, then?
- A: It comes because we generated one more level than $\lfloor C^*/\varepsilon \rfloor$.