

Homework 1

Blind Search and Informed Search

Due: 6:00pm on 09/23/09

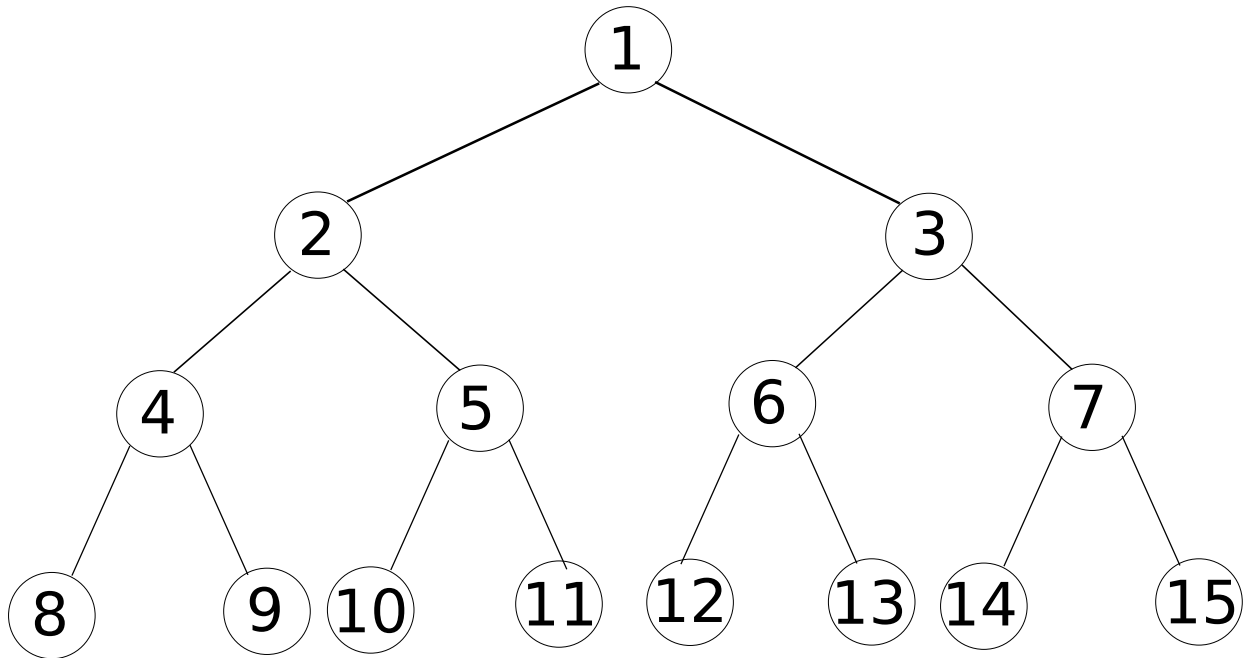
Problem 1.1

Consider a state space where the start state is number 1 and the successor function for state n returns two states, numbers $2n$ and $2n + 1$.

- a. Draw the portion of the state space for states 1 to 15.
- b. Suppose the goal state is 11. List the order in which nodes will be visited for breadth first search, depth-limited search with limit 3, and iterative deepening search. For Depth-Limited search, take the root node to have depth 1, such that a Depth-Limit 1 search explores the root and nothing else. When considering iterated deepening, please report those states visited multiple times as such (i.e., *don't* report only the *first* time the node is visited.)
- c. Describe how one could apply bidirectional search to this problem. Why is it appropriate?
- d. What is the branching factor in each direction of the bidirectional search?
- e. Given the answers above, suggest an even more efficient way to solve this problem than bidirectional search.

Solution:

- a. see figure.



b. BFS: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

DFS: 1, 2, 4, 8, 9, 5, 10, 11

Depth-Limit 3: 1, 2, 4, 5, 3, 6, 7 (Cutoff; Goal node not reached)

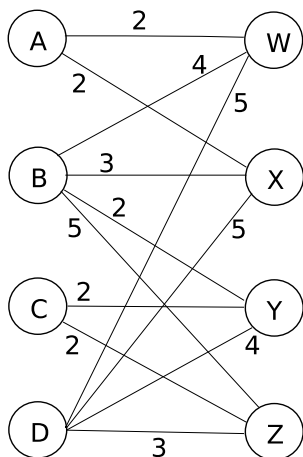
ID: 1; 1, 2, 3; 1, 2, 4, 5, 3, 6, 7; 1, 2, 4, 8, 9, 5, 10, 11

c. Yes, bidirectional search would work well, because there is one explicitly defined goal state and it's easy to generate the predecessor function: if (even) move to $x/2$; else, move to $(x-1)/2$.

d. Branching factor is $2x$ moving forwards and $1x$ moving backwards.

e. Search only backwards from the goal state to 1—no branching, and so a very easy problem.

Problem 1.2



Sello Construction has 4 backhoes and 4 projects that need backhoes. Backhoes don't get very good gas mileage, so Sello Construction has hired you to minimize its gas costs. The goal is to match each of the backhoes A-D with a construction site W-Z. Every backhoe must be matched with exactly one site. Each edge is labeled with a number that represents the cost of using that edge in the matching. You want to minimize the cost of the matching so that Sello Construction saves money and wants to hire you again. What is the optimal matching? How do you know it is optimal? Please show work.

Solution:

The minimum matching is A-W, B-X, C-Y, and D-Z, for a total cost of 10. Even if all possible connections from backhoes to projects were allowed, there would only be $4 \times 3 \times 2 \times 1 = 24$ possibilities. In this case, there are only eight:

Possible matchings:

A	B	C	D	TotalCost
W	X	Y	Z	10
W	X	Z	Y	11
W	Y	X	Z	11
W	Z	Y	X	14
X	W	Y	Z	11
X	W	Z	Y	12
X	Y	W	Z	11
X	Z	W	Y	14

Because we've enumerated all possibilities, we can see that A-W, B-X, C-Y, D-Z is the optimal match.

Problem 1.3

Consider the algorithm wA^* , which is a variant of A^* search that uses the following weighted cost function: for some $w \geq 1$,

$$f_w(n) = g(n) + wh(n)$$

As usual, g is the cost from the root node to n and h is an admissible heuristic.

(a) Prove that the goal node m^* returned by wA^* search on trees is within a factor of w of the optimal goal n^* : *i.e.*, $g(m^*) \leq wg(n^*)$.

Solution:

Since wA^* returns goal node m^* , it must be the case that there exists some node n on the path to n^* for which $f_w(m^*) \leq f_w(n)$. Now

$$\begin{aligned} g(m^*) &\leq g(m^*) + wh(m^*) && \text{since } h(m^*) \geq \delta \\ &= f_w(m^*) && \text{by definition} \\ &\leq f_w(n) && \text{by assumption} \\ &= g(n) + wh(n) && \text{by definition} \\ &\leq g(n) + wh^*(n) && \text{by admissibility} \\ &\leq w(g(n) + h^*(n)) && \text{by } w \geq 1 \\ &= w(g(n^*)) && \text{distance to optimal goal} \end{aligned}$$

(b) The wA^* algorithm uses a weighted cost function that increases the value of the heuristic function h , hoping to proceed more directly towards a goal. An alternative is to ignore g entirely, simply letting $f = h$. Give two advantages of wA^* over this alternative.

Solution:

Note that the alternative is best- h search. 1. Although neither algorithm is optimal, the degree of suboptimality of wA^* is precisely specified above. 2. wA^* is complete; best- h can behave like depth-first search, which need not return a goal, even when one exists.

Problem 1.4

The pseudocode in Table 1 implements **beam** search (on trees), a memory-bounded heuristic variant of breadth-first search.

(a) Give the time and space complexity of beam search in terms of branching factor b , depth d , and beam width w .

Solution:

The time complexity of beam search is $O(bdw)$: at each of the d levels in the search tree, for each of the w nodes on the beam, b children are expanded. The space complexity of beam search is $O(w + b)$, since 2 queues are maintained.

(b) Argue whether or not beam search is optimal and complete.

BEAM(X, S, G, δ, c, w, h)	
Inputs	<ul style="list-style-type: none"> a set of states X a nonempty set of start states S a nonempty set of goal states G a state transition function δ that takes in a state x and outputs the set of all successor states of x a cost function c that takes in two states and computes the cost of transitioning from the first state to the second state beam width w heuristic h
Output	(path to) goal node
Initialize	<ul style="list-style-type: none"> $O = S$ is the list of open nodes P is the beam (<i>i.e.</i>, priority queue)
<pre> while (O is not empty) do $P = O, O = \emptyset$ while (P is not empty) do a. delete node $n \in P$ s.t. $f(n)$ is minimal b. if $n \in G$, return (path to) n c. for all $m \in \delta(n)$ (a) compute $h(m)$ (b) $g(m) = g(n) + c(m, n)$ (c) $f(m) = g(m) + h(m)$ (d) insert node m into O with priority $f(m)$ (e) truncate O to maximum beam width w fail </pre>	

Table 1: Beam Search.

Solution:

Beam search is not complete, since (for example) it is possible that the only goal node lies along a path that traverses through the $(w+1)$ -best successor of the start state. Likewise, beam search is not optimal, since the optimal goal might lie along that same path.