

Homework 1

Blind Search and Informed Search

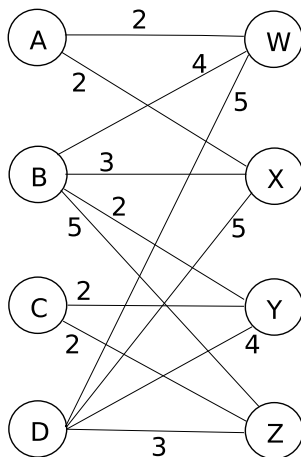
Due: 6:00pm on 09/23/09

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 $2n + 1$.

- Draw the portion of the state space for states 1 to 15.
- 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.)
- Describe how one could apply bidirectional search to this problem. Why is it appropriate?
- What is the branching factor in each direction of the bidirectional search?
- Given the answers above, suggest an even more efficient way to solve this problem than bidirectional search.

Problem 1.2



Sello Construction has 4 backhoes and 4 projects that need backhoes. Backhoes don't get very good gas milage, 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.

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.*, $g(m^*) \leq wg(n^*)$.
- (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.

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 .
- (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)
<hr/>	
while (O is not empty) do	
$P = O, O = \emptyset$	
while (P is not empty) do	
a. delete node $n \in P$ <i>s.t.</i> $f(n)$ 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	

Table 1: Beam Search.