15381 Artificial Intelligence: Assignment 1

Due September 16th, beginning of class

September 9th, 2010

Instructions: There are 3 questions on this assignment. Please send us emails at *15381-tas@lists.andrew.cmu.edu* when you have questions. Refer to the web page for policies regarding collaboration, due dates, and extensions. The TA names listed next to each problem show which TA wrote and is responsible for grading which problem. However, please feel free to speak to any TA or instructor regarding any problem.
Late Policy: Each student is allowed up to 2 late days for the whole semester, to be used with advance permission only. That means you must contact an instructor and request permission to use a late day for an assignment. Any late assignment without an instructor-permitted late day will not be graded.
Collaboration Policy: You can discuss questions with classmates, but the write-up should be of your own. If you discuss the problems with collaborators, you must list their names on your write-up.
Outside Sources: We expect you to solve these problems with the help of the textbook and the lecture slides (and the TAs and instructors) rather than searching for previous solutions and copying them. If you use any source other than the textbook or lecture slides, you must cite the source.

1 Functions, logic and search algorithms[25 pts] [Ben]

This problem combines functions, logic, and search algorithms. First, let Q be a set of possible states as defined in class, and let $q \in Q$ be one such state. We will call a function $f(q) \rightarrow \{0, 1\}$ a property of q and a function $g(q, q') \rightarrow \{0, 1\}$ a property of q and q'. For example, if Q is a set of people, then $b(q)$ could be true if and only if person q is a boy and $r(q, q')$ could be true if and only if person q is related to person q'.

For each of the following problems, you must define the set of states Q, the properties that apply to $q \in Q$, the start states S, the goal states G, the successor function succ(q), and the cost function cost(q, q'). Then, describe what algorithm you would use to solve the

problem efficiently and why you chose that algorithm. An example problem and solution follows below. Please note that you will be graded on both the conciseness and completeness of your problem representations as well as your pick and justification for your solution of choice. Some problems may have more than one good algorithmic choice: you do not have to name them all to get full credit, but you must give a justification that is factually correct and well-reasoned. All sets of states, including succ(q) and cost(q, q') must be defined using propositional logic notation, as shown below.

Example Problem

You accidentally left your phone in your friend's car and you need it back as soon as possible. However, you don't have your friend's cellphone number. Further, you know that there must be someone in your circle of friends who does know this friend's number. Unfortunately, nobody in your circle of friends, including yourself, knows everybody else's cellphone number. Find your phone.

Example Solution

Q = individuals in your circle of friends.

$P(q)$ is true if and only if person q has your phone.

$K(q, q')$ is true if and only if person q knows the phone number of person q'.

S = You.

$G = \{g | g \in Q \wedge P(g)\}$

$\text{succ}(q) = \{q' | q, q' \in Q \wedge K(q, q')\}$

$\text{cost}(q, q') = 1$

This problem can be solved with either breadth-first search or depth-first search because all edge weights are equal and positive. Either option could be more effective depending on which party guest remembers which other party guest(s).

Problem Parts

1.a) [10 Points] You receive a phone call from a very drunk friend who needs you to pick them up. Unfortunately, they slur a lot when drunk so you couldn't hear their exact location. You know your friend only drinks at certain bars, but those bars are all over

the city. You don't want to waste gas or time finding your friend, so you need an optimal search algorithm.

1.b) [5 Points] Your drunk friend called you *again* and needs a ride *again*. Fortunately, this time, you were prepared! You have your friend's last few credit card statements and, thus, you know how often he or she drinks at each bar. Find your friend even more efficiently than before (on average, over many drunk calls).

2.a) [10 Points] When you were last on vacation, you tasted a really great smoothie. You can't stop thinking about it, but you don't know the recipe. Fortunately, SCS just got a new smoothie-making robot. The robot can put in any one piece of fruit at a time. You don't know the ratio of fruits in the ultimate smoothie, so it could require, for example one banana and one orange, or 17 strawberries and one apple, but you know that there are no more than 20 pieces of fruit in the smoothie. Finally, you know you can recognize the smoothie if you taste it, but you can't really tell if a smoothie is close to the great smoothie or not, only if it *is* the great smoothie. Write an algorithm to find the great smoothie. Do not use heuristics.

2.b) [3 Points Extra Credit] Now you want to make the *ultimate* smoothie. Each time you taste an SCS-robot smoothie, you (the drinker) can give it (the drink) a rating from 0 to 100, with 100 being perfect and 0 being terrible. Once again, limit the number of fruits to 20. Define a heuristic to improve your search from before. Heuristics will be graded for originality, effectiveness of the heuristic, and the quality of the supporting rationale. Do not assume anything about the probability distribution of delicious smoothies. You may, however, assume that the ultimate smoothie exists in your search space and is not a myth fabricated by late-night infomercials. Expect no more than 2 extra credit points unless your solution is significantly better than the TA's solution.

2 Search **[25 pts] [Sam]**

1. Suppose we are given a search problem whose search space is a tree T with branching factor b and total depth m. Assume that the shallowest goal g has depth d, and that no other goals have depth d. Also assume that g is the last node of depth d added to the search queue by both Breadth First Search (BFS) and Iterative Deepening Search (IDS). Please justify your answers to the following questions, evaluating all summations.

   (a) [2 Points] For $i \in [0, m]$ how many nodes are there in T of depth i?

   (b) [2 Points] How much space does BFS require in this search problem?

   (c) [2 Points] How much space does IDS require in this search problem?

   (d) [2 Points] How many nodes does BFS visit before finding the first goal?

(e) [2 Points] How many nodes does IDS visit at iteration i (where iteration 0 is the iteration where we remove the start node from the queue for the first time)?

(f) [2 Points] How many total nodes does IDS visit before finding the first goal?

(g) [2 Points] Which algorithm visits more nodes—BFS or IDS? What are the asymptotic running times (i.e., in big O notation) of the two algorithms, treating b, d, and m as parameters? Please justify your answers using the prior parts of this question—do not just cite results from lecture.

2. 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) [2 Points] Draw the portion of the state space for states 1 to 15.

(b) [2 Points] Suppose the goal state is 11. List the order in which nodes will be visited for breadth-first search, depth-limited search with limit 4, and iterative deepening search.

(c) [3 Points] Would bidirectional search be appropriate for this problem? If so, describe in detail how it would work.

(d) [2 Points] What is the branching factor in each direction of the bidirectional search?

(e) [2 Points] Does the answer to 2d suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

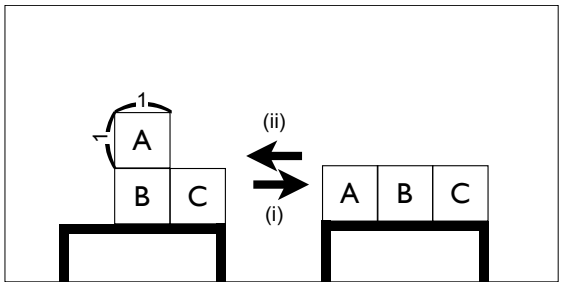
3 Squares World [25 pts] [Wooyoung]

In Squares World, there are squares and a table big enough to hold all the squares. Each square is on the table or on a single other square. For each square a, either it is clear or it has another unique square b that sits on it. Here we can only pick a single clear square and i) move it from another square onto the table(Figure 1(a)), ii) move it from the table onto another clear square(Figure 1(a)), iii) from a square onto another clear square(Figure 1(b)). In a single move, squares are not allowed to hop over others(Figure 1(c)).

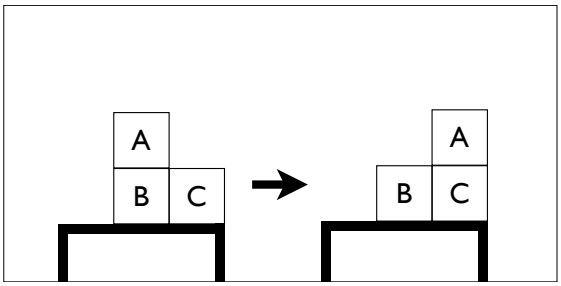
In this problem, we assume that there are only three squares with the same size (the edge length is one), A,B and C. Our table is a line and its length is three. Of course, the squares can only stay on the table. Your mission is to use the search methods covered in class and start from the initial state and reach the goal state each described in (Figure 1(d)). When expanding a node in a way results in a cycle, you can ignore that expansion.

1. [6 Points] Show the search tree for depth-first search. Mark the nodes with the orders they are visited. How many nodes are visited until you reach the goal?

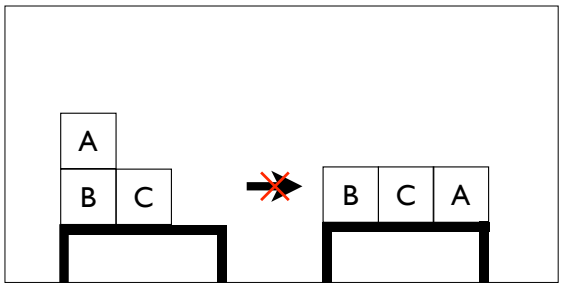
2. [6 Points] Show the search tree for breadth-first search. Mark the nodes with the orders they are visited. How many nodes are visited?
3. [6 Points] Show the search tree for A^* with $g(s)$ being the number of moves so far and $h(s)$ of your own definition. Mark the nodes with the values of $f(s) = g(s) + h(s)$. How many nodes are visited?
4. [7 Points] Which search method worked the best in this specific problem? Why?



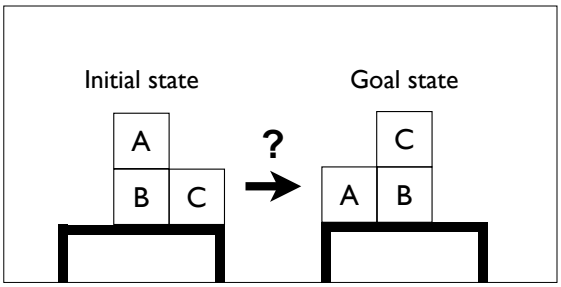
(a) You can move a single clear square onto the table or vise versa.



(b) You can move a single clear square from a square to another clear square.



(c) Squares cannot hop over others in a single move.



(d) Your mission

Figure 1: (a),(b) Possible single moves. (c) Not possible single move. (d) The initial and Goal states for your mission.