CSB551 – Elements of Artificial Intelligence – Fall 2011

Homework #1
(Search Problems)
Due: 9/15/09 (5:15 pm)

How to complete this HW: Either 1) type your answers in the empty spaces below each problem and print out this document, or 2) print this document and write your answers in the empty spaces on the printout. Return the homework in class, during office hours, or slip it under the door of Informatics East, Room 257 before 5:15 on Thursday, 9/15/09.

Your name: ENRIQUE AREYAN
Your email address:
Note on Honor Code: You must NOT look at previously published solutions of any of these problems in preparing your answers. You may discuss these problems with other students in the class (in fact, you are encouraged to do so) and/or look into other documents (books, web sites), with the exception of published solutions, without taking any written or electronic notes. If you have discussed any of the problems with other students, indicate their name(s) here:
……………………………………………………………………………………………………
Any intentional transgression of these rules will be considered an honor code violation.

General information: Justify your answers, but keep explanations short and to the point. Excessive verbosity will be penalized. If you have any doubt on how to interpret a question, tell us in advance, so that we can help you understand the question, or tell us how you understand it in your returned solution.

Grading:

Problem#	Max. grade	Your grade
I	20	
II	20	
III	20	
IV	40	
Total	100	

Name ……………………………………………………………………..

I. Map coloring (20 points)

1. Description of a state: we could model this problem as an undirected graph *G*, in which the set of nodes *N* represent the regions of the map and the set of edges *E* represent a connection between two adjacent regions. For the example, the map of the United States would be modeled as *N*={Florida, Alabama, Georgia, Tennessee, … , California} and *E* = {(Florida, Alabama), (Florida, Georgia), (Alabama, Tennessee) … (California, Oregon)}.

   Let's consider the set of possible colors *C* ={Red, Blue, Yellow, Green, No Color}. An individual state of the problem would correspond to a color assignment for each region in the map, i.e. a function $N \rightarrow C$. Let's call the set of all possible states of the problem *S,* e.g., one member of *S* would be {Florida, Alabama, Georgia, … , California}, which correspond to an assignment $N_i \rightarrow C$ where only Florida and Alabama have been colored.
   There are $5^{50} = |S|$ possible states or N_i functions.

2. Initial State: S_0 = {Florida, Alabama, Georgia, … , California}, an assignment of a random color to a single region.

3. Goal test: check to see if all states have been colored, e.g. S_g = {Florida, Alabama, Georgia, … , California} and no two adjacent regions have the same color.

4. Successor function: Let's call the successor function SUCC(*p*), where $p \in S$. Then SUCC would check, in an orderly fashion, for any uncolored region inside *p*. If an uncolored region is found, it will found its immediate adjacent regions and colored them, along with the region, so that it does not break the two adjacent color rule.

II. Missionaries and Cannibals (20 points)

1. State the Missionaries and Cannibals puzzle as a search problem. Give precise definitions of the:

a. **State space**: a single state could be represented with the following string: MC | MMCC, where the bar "|" breaks the string in two, the left side of the lake and the right side. Thus, in the previous example we would have one missionary and one cannibal in the left side of the lake and the rest on the right. With this string configuration, we can define the state space *S* as the set of all such strings that there are no more C's than M's at any side of the bar "|", if there are any M's at all in a particular side of the lake, e.g. this configuration would be valid CC|MMMC. Note that we will always order the string such that if there is an M is always before any C.

b. **Successor function**: SUCC(*p*) = takes two characters from the right side of the bar and moves only one to the left side, never leaving more C's than M's except when there are no M's. For instance SUCC(|MMMCCC) = {C|MMMCC}

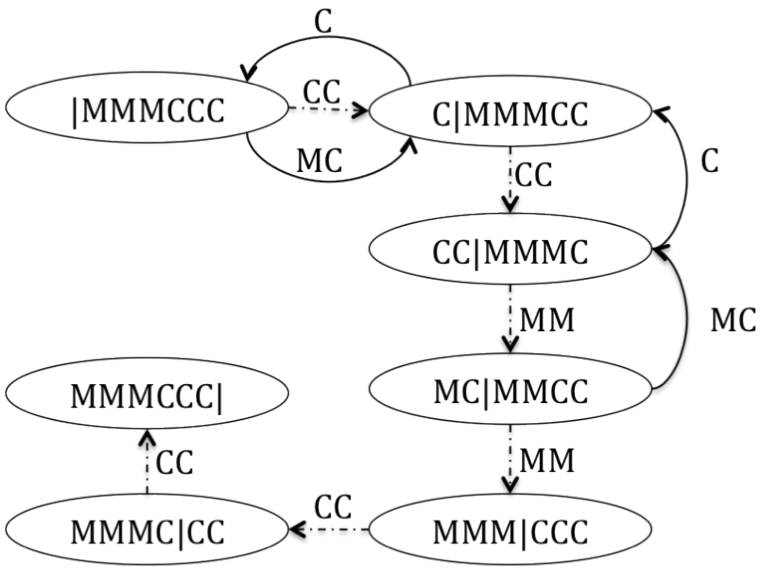
Name ……………………………………………………………………..

c. **Initial state**: all three missionaries and cannibals on the right side of the lake, i.e. the string |MMMCCC.

d. **Goal test**: check if all three missionaries and three cannibals are on the left side of the lake, i.e., the string MMMCCC|

2. Draw a complete, labeled **state graph** for this problem (not a search graph). Indicate an optimal solution path. (The graph should not be prohibitively large if your formulation in problem 1 is correct)

State Graph: the edge labels indicate what characters are being move to obtain a new state. The optimal path is indicated with dashed arrows.



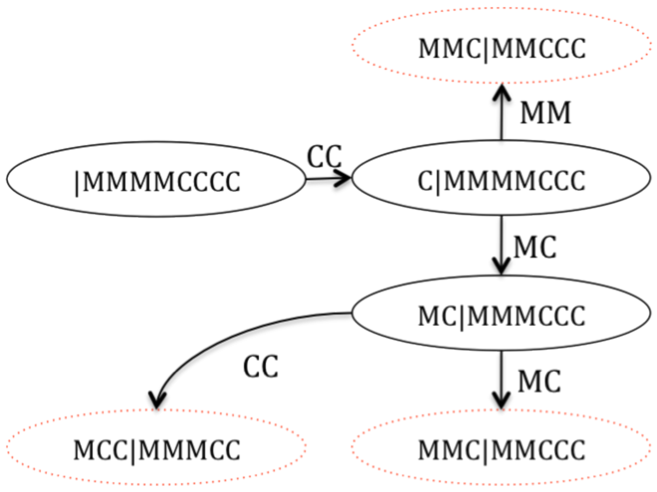
3. What blind search algorithm would you use to solve this problem? Would you check for repeated states?

An: I would use a Breadth-first search algorithm, which is complete and optimal, considering that in this problem the steps cost are uniform. Also, I would check for repeated states to avoid any loops.

4. Consider a generalized Missionaries and Cannibals puzzle with N missionaries and N cannibals. How quickly does the state space grow with N? Can these problems always be solved for N>3? Why or why not?

An: The following is the state graph for the missionaries and cannibals puzzle with 4 missionaries and 4 cannibals. The red, dashed ovals indicate a terminal, non-goal state. From the graph we can deduct that the game has no solution when N=4, therefore it is not scalable for N>3.

Name …………………………………………………………………..



III. Heuristic Search (20 points)

1.

N	State	*g*(*N*)	*h*(*N*)	*f*(*N*)	#exp
1	S	0	10	10	1
2	A	3	5	8	2
3	B	7	8	15	
4	C	4	3	7	3
5	D	9	2	11	5
6	S	6	10	16	
7	D	8	2	10	4
8	G1	14	0	14	6
9	B	11	8	19	
10	G1	15	0	15	
11	B	12	8	20	
12					
13					

2. Is the heuristic function h defined by values provided in the figure admissible? How do you know? How does this affect the search?

No, *h()* is not an admissible heuristic because there exists a node for which the relation $0 \le h(n) \le h^*(n)$ does not hold. Specifically, for node B, *h*(B) = 8 while the optimal path from B to G2 is 6 (through node E). This causes the search not to open node B when it should to find the optimal path.

IV. Word Puzzle (Programming) (40 points)

1. Describe your successor function.
    a. How do you know it provides all legal successors and no other states?

Name ……………………………………………………………………..

An: my successor function implements a double, nested loop, with the outer loop iterating over the length of the input word and the inner loop iterating over the letters in the variable ALPHABET, changing only a fixed letter of the original word with a letter of the alphabet at the time. The new, possible word is tested to be in the self.dictionary. If the word is in the dictionary, then it is added to the successors words' list, if not it is discarded. The outer loops then reinitialize the possible word to be the original word so as to change a new fixed letter. Therefore, the successors list will only contain valid words that can be formed by changing only one letter of the original word, i.e.:

Let *p* be a valid English word with length *n* and characters $p_1p_2\cdots p_n$, A denote the set of letters in the English alphabet (|A| = 26), and D the set of all English words. Then:
SUCC(*p*) = $\{p_xp_2\cdots p_n \mid p_x \in A \textit{ and } p_xp_2\cdots p_n \in D\} \cup \{p_1p_x\cdots p_n \mid p_x \in A \textit{ and } p_xp_2\cdots p_n \in D\}$ $\cup \cdots \cup \{p_1p_2\cdots p_x \mid p_x \in A \textit{ and } p_xp_2\cdots p_n \in D\}$

2. Assume that the input words are of length n. For your successor function, state an upper bound on the branching factor of the search tree.

   An: in the worst-case scenario, any one-letter change for any given state will yield another valid state (we know that this is not the case because most of the changes will result in a non-valid English word, however, we will assume this for the sake of simplicity). There are 26 letters in the English alphabet, so a change in one letter will produce 26 new words or states. By the addition principle there will be 26 + 26 + 26 + … + 26 = $n * 26$ new possible states or branches of the search tree.

3. Describe your heuristic function.
   An: my heuristic counts the number of letters of a word that are different from the goal word. So, let's call the heuristic function *h*(word) so that h: words → natural, positive numbers including zero. For example, given the goal word 'hello' and the word 'hallo', then *h*(hallo) = 1, meaning that there is only one misplaced letter in the word 'hallo' to get to 'hello'. It should be noted that the heuristic of a goal word is always zero (there are no misplaced words once reached our destination)
   a. Show that it is admissible.
      An: next, I show that it is a consistent heuristic. Therefore, it is admissible.
   b. Show that it is consistent.
      An: a heuristic is consistent if for every node *n* and every successor *n'* of *n* generated by any action *a* the following holds:
      $h(n) \leq c(n,a,n') + h(n')$, such that c() stands for the estimated cost of reaching *n'* from *n* by taking action *a*. For this problem we have that c() is always 1 (uniform cost), therefore we only need to prove $h(n) \leq 1 + h(n')$, for every *n* and *n'*.

      Let's assume that for an arbitrary node *n*, h(*n*) = x. Any successor *n'* of *n* will differ in relation to *n* only in one character, therefore h(*n'*) could only either stay the same (h(*n'*) = h(*n*)), if the changed letter does not coincide with the

Name ………………………………………………………………..

corresponding letter in the goal word, or decrease by 1, (h(*n'*) = h(*n*) -1), if the changed letter do coincide with its counterpart on the goal word. In the former case we have that h(*n*) ≤ 1 + h(*n'*) ⇒ x ≤ 1 + x, which is always true; and in the latter, h(*n*) ≤ 1 + h(*n'*) ⇒ x ≤ 1 + x – 1 ⇒ x ≤ x, which is also true.

The following table compares search result using and not using the heuristic

Start Word	Goal Word	Using Heur.?	Nodes	Avg. Bran. Factor
sat	roc	No	1,155	1.843
sat	roc	Yes	81	26.667
arc	lot	No	1,249	1.418
arc	lot	Yes	84	8.3
word	pare	No	1,469	4.828
word	pare	Yes	43	14
hare	fray	No	3,564	4.012
hare	fray	Yes	769	7.917
taupe	brown	No	6,790	1.220
taupe	brown	Yes	459	3.053
smith	felid	No	7,699	1.082
smith	felid	Yes	3,035	1.862
campus	coffee	No	3,409	1.167
campus	coffee	Yes	328	2.919
sweets	pastry	No	8,137	1.005
sweets	pastry	Yes	7,932	1.161

4. How does your heuristic function compare to the null heuristic?
   a. Does it provide shorter paths?
      An: no, it provides the paths of the same length.
   b. Does it visit (i.e., expand) fewer nodes? How do you know?
      An: the heuristic does visit fewer nodes than the null heuristic. I know this both from practical experimentation as can be seen in the table of question 3 and from theoretical implications on the use of the heuristic.
   c. How does it affect the running time of the search as a whole?
      An: running the algorithm with the heuristic lowers the overall search running time because it visit far less nodes than it do otherwise.
   d. Does it take less time to run, per-state?
      An: no, it takes more time because it needs to evaluate the heuristic function per-state. However, because there are considerably less states to evaluate, the running time as a whole is much less with the heuristic.