Last name:_____ First name:_____ SID:_____ Class account login:_____

Collaborators: _____

CS188 Spring 2010 Written 1: Search

Due: Thursday 1/28 in 283 Soda Drop Box by 11:59pm (no slip days)

Policy: Can be solved in groups (acknowledge collaborators) but must be written up individually.

1 [8pts] All roads lead to Rome

This map shows approximate mean driving times (in hours) between pairs of cities. For each of the following graph search strategies, work out the order in which states are expanded, as well as the path returned by graph search. In all cases, assume ties resolve in such a way that states with earlier alphabetical order are expanded first. The start and goal states are Warsaw and Rome, respectively. Remember that in graph search, a state is expanded only once.

[1pt] (a) Depth-first search.

[1pt] (b) Breadth-first search.

[1pt] (c) Uniform cost search.

[1pt] (d) Greedy search with the heuristic: $h(\text{Odesa}) = 20$ hrs, $h(\text{Budapest}) = 12$ hrs, $h(\text{Munich}) = 3$ hrs, $h(\text{Venice}) = 3$ hrs, $h(\text{Rome}) = 0$ hrs, $h(\text{Warsaw}) = 30$ hrs.

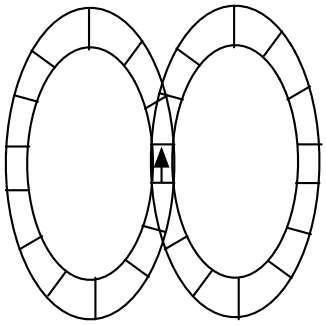
[1pt] (e) $A^\star$ search with the same heuristic.

[2pts] (f) Is the heuristic admissible? Consistent? Briefly justify your answers.

[1pt] (g) Which of the following is required for $A^\star$ graph search to return an optimal solution?
(i) Admissible
(ii) Admissible and consistent.
(iii) It does not depend on the choice of heuristic, it always returns the optimal solution.

2 [7pts] Trains

Imagine a train-like agent wishing to do at least one revolution in each of the loops of the railway shown here. The agent cannot turn around and is always moving clockwise in the right loop and counter-clockwise in the left loop. The agent's initial velocity is zero, but can move at the integer velocities in [0, V]. At each time step, the agent can either coast (not change velocity), accelerate (increase velocity by 1), or decelerate (decrease velocity by 1). Additionally, if the train will traverse the central square in the next time step, it has the choice to continue on the left or the right loop. Once an action is selected, the agent then moves a number of squares equal to its NEW adjusted velocity. For example, if the first action is to accelerate and pick the right loop, the agent will end up one square to the top right with a new velocity of 1. The agent's goal is to find a plan which parks it (at zero velocity) on the central square after having done at least one revolution in each of the loops, while using as few actions (time steps) as possible. Assume the agent begins in the central square as well.



Let L denote the number of squares per loop. (Hence there is a total of $2L - 1$ squares.) Assume that $V < L$.

[1pt] (a) What is the size of the state space? Briefly justify your answer. You need not consider whether a state is actually reachable from the start state.

[1pt] (b) What is the maximum branching factor of this problem? Briefly justify your answer.

[2pts] (c) Is the shortest forward distance to the central square an admissible heuristic? Briefly justify your answer.

[2pts] (d) State and justify a non-trivial, admissible heuristic for this problem which is not the shortest forward distance to the central square.

[1pt] (e) If we used an inadmissible heuristic in A* tree search, could it change the completeness of the search in this problem?

3 [6pts] Sequence Alignment

In the last decade, many AI algorithms have been applied to solve computational biology problems. In this exercise, we show how the search algorithms we have seen in class have been used as the foundation of the current state-of-the-art algorithm for *sequence alignment*, a core problem in computational biology.

The input of a sequence alignment problem is a set of related DNA sequences, where each one corresponds to a species. Consider for example the following toy genome:

Mouse: AGACG
Turtle: CAGTGCT

Note that both sequences share the substring AG, suggesting that the common ancestor of the species may also have had this subsequence in its genome. Observe also that the sequences have different lengths. This is due to the fact that evolution operates not only by substitution of nucleotides (e.g. maybe the ancestor had an A at the end of its genome, which became a G in turtles), but also by insertions and deletions.

Sequence alignment consists of padding sequences by using a special character (called a gap, denoted by "-") so that they all have a common length, while revealing the patterns supporting shared ancestry. Here is an example of an alignment:

Mouse: -AGA--CG
Turtle: CA-GTGCT (1)

The *cost* of an alignment is computed as follows. First, for each gap that was used, a penalty g is incurred. Second, for each column containing two different nucleotides, a mismatch penalty m is incurred.[1] Finally, there is no penalty when a column contains two identical nucleotides. The cost of the alignment is the sum of these penalties. For example, the cost of alignment (1) is $4g + 2m$. We assume that $g > m > 0$. The task is to find an alignment of minimal cost.

Sequence alignment can be formulated as a search problem in many ways. For this question, consider the following formulation, where sequences are aligned from left to right. The start state is denoted by $(0, 0)$. The first zero indicates zero nucleotides have been aligned so far in the first sequence. The second zero indicates zero nucleotides have been aligned so far in the second sequence. At each step, the available actions are: (i) read one nucleotide in from each of the sequences and align them, this will transition a state (k, l) to $(k+1, l+1)$, or (ii) read one nucleotide from only the first sequence (and align it to a gap), this will transition a state (k, l) to $(k+1, l)$, or (iii) read one nucleotide only from the second sequence (and align it to a gap), this will transition a state (k, l) to $(k, l+1)$. We let L_1 and L_2 denote the length of the first and second input sequences, respectively. The state (L_1, L_2) is the goal state: in this state all L_1 nucleotides from sequence 1 and all L_2 nucleotides from sequence 2 have been aligned.

[1pt] (a) Give a non-trivial bound on the size of the state space as a function of the lengths L_1 and L_2 of the first and second input sequences, respectively.

[1] In practice, mismatch penalties have more structure—for example the cost of a mismatch between `A` and `T` is higher than the cost between `A` and `G`. In the context of this written assignment, we do not consider this differentiation in cost depending on the particular mismatch.

[5pts] (b) Let ΔL be the absolute length difference between the parts of the two sequences that are not yet aligned. Concretely, for the state (k, l) we have $\Delta L = |(L_1 - k) - (L_2 - l)|$. Also, let ΔC be the absolute difference between the number of Cs in the parts of the two sequences that are not yet aligned (and similarly for ΔA, ΔG,ΔT). Which of the following heuristics are admissible? Briefly justify your answers.

1. $H_1 = g \cdot \Delta L$

2. $H_2 = g \cdot \max\{\Delta\text{C}, \Delta\text{A}, \Delta\text{G}, \Delta\text{T}\}$

3. $H_3 = m \cdot \max\{\Delta\text{C}, \Delta\text{A}, \Delta\text{G}, \Delta\text{T}\}$

4. $H_4 = m \cdot \max\{\Delta\text{C}, \Delta\text{A}, \Delta\text{G}, \Delta\text{T}, \frac{1}{2}(\Delta\text{C} + \Delta\text{A} + \Delta\text{G} + \Delta\text{T})\}$

5. $H_5 = H_1 + H_3$