CMPT 417/827
Fall 2024
Sample Late Midterm Exam
11/14/2024
Time Limit: 150 Minutes

Name: ____________________

ID #: ____________________

This exam contains 13 pages (including this cover page) and 6 questions (including a bonus question worth 4 bonus points).
Total of points is 30 plus 4 (bonus).

Grade Table (for teacher use only)

Question	Points	Score
1	4	
2	5	
3	5	
4	9	
5	7	
6	0	
Total:	30	

1. (4 points) Single-choice questions.
   (a) (2 points) Which of the following conditions reduce an A* search to a breath first search?
      A. $g(n) = c$, c is a constant
      B. $h(n) = c$, for all nodes n and c is a constant
      C. Costs of all actions are equal, and $h(n) = c$ for all n and c is a constant
      D. $h(n) = 1$ for all n
   (b) (2 points) What is true about IDA*?
      A. Guarantee to find an optimal solution even with inadmissible heuristics
      B. Can find an optimal solution even with inconsistent heuristics
      C. Can sometimes find a better solution than A*
      D. Guarantee to run faster than A*

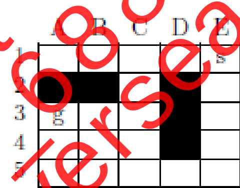
2. (5 points) What are the advantages and disadvantages of a) uniform-cost search and b) pure heuristic search over A* search?

> **Solution:** The only advantage uniform-cost search has is that it does not have to compute a heuristic (which can be very expensive in some domains). Both search methods guarantee optimality. Pure heuristic search offers no optimality guarantees, although it typically finds solutions with many fewer node expansions and thus much faster than A*.

3. (5 points) In MAPF, one approach to optimize a given initial solution using Large-Neighborhood Search (LNS) is to iteratively remove the paths of a selected subset of agents and replan them. Assume that CBS is used for replanning. Experimental observations suggest that replanning the paths of a larger subset of agents generally results in a greater reduction in the total cost of the solution. Based on this, do you think it is always beneficial to replan paths for a very large subset of agents in each iteration?

> **Solution:** Replanning paths for a larger subset of agents is not always beneficial. While increasing the subset size tends to achieve a larger decrease in total cost, it also tends to increase the runtime per iteration. Therefore, there is a tradeoff between the reduction in cost and the runtime required for each iteration.

4. (9 points) We are given a 4-neighbor grid below.



(a) (6 points) In which order do depth-first search and breadth-first search (both with a sensible duplicate-detection strategy) expand the states when searching from s to g? Ties are broken in lexicographic order. That is, $A1$ is preferred over $A2$ and $B1$, and $A2$ is preferred over $B1$.

> **Solution:** The duplicate-detection/node-pruning strategy of breadth-first search is to prune nodes whenever some node in the search tree is already labeled with the same state. The duplicate-detection/node-pruning strategy of depth-first search is to prune nodes whenever some node on the same branch of the search tree is already labeled with the same state (cycle detection).
>
> Breadth-first search: $(E1)$, $(D1, E2)$, $(C1, E3)$, $(B1, C2, E4)$, $(A1, C3, E5)$, $(B3, C4, D5)$, $(A3)$. The parentheses group states based on their depth in the search tree. Depending on the tie-breaking strategy, the order of expansions can change, but only within a group.
>
> Depth-first search: $E1$, $D1$, $C1$, $B1$, $A1$, (backtrack to $B1$ and then $C1$), $C2$, $C3$, $B3$, $A3$.

(b) (3 points) In the arrow puzzle, we have a series of arrows pointing up or down, and we are trying to make all the arrows point up with a minimum number of action executions. The only action available to us is to choose a pair of adjacent arrows and flip both of their directions. Using problem relaxation, come up with a good heuristic for this problem.

> **Solution:** We can relax the problem by allowing our action to flip the directions of any two arrows, instead of two adjacent arrows. In this case, we need to use at least [*number of arrows pointing down* / 2] actions, which can be used as a consistent heuristic to solve the original problem.

5. (7 points) Imagine we create a pattern database for tiles 1, 3, 5 for the TopSpin puzzle. We use a multi-dimensional array PDB[ ][ ][ ] (with indices ranging from 1 to 9 for the three dimensions that indicate the positions of tiles 1, 3, 5, respectively) to store the heuristic values. We want to get a heuristic value for getting from a random configuration (Left) to the goal configuration (Right) by looking up the pattern database.

1	2	3	8	9	5	4	7	6

1	2	**3**	4	**5**	6	7	8	9

(a) (1 point) If we perform a regular lookup, which entry do we query for this random configuration?

> **Solution:** PDB[1][3][6]

(b) (3 points) Dual lookup: If we perform a dual lookup, which entry do we query for this random configuration?

> **Solution:** PDB[1][3][9]
> (Getting tiles 1,3,9 from positions 1,3,5 to positions 1,3,9 uses the same cost in the pattern space as getting tiles 1,3,5 from positions 1,3,9 to positions 1,3,5.)

(c) (3 points) (Geometric) symmetric lookup: If we leverage the geometric symmetry by reflecting over the middle tile 5, we virtually get another pattern database for 3 tiles for free. If we perform a geometric symmetric lookup, which entry do we query for this random configuration?

> **Solution:** PDB[5][2][4]
> (We get a PDB for 9, 7, 5 for free. Getting tiles 9, 7, 5 from positions 5, 8, 6 to 9, 7, 5 uses the same cost in the pattern space as getting tiles 1, 3, 5 from positions 5, 2, 4 to 1, 3, 5.)

6. (4 points (bonus)) Dummy question.