CMPT 310 Assignment 2 Suggested Solutions

1. [0%]

   (a) N/A.

   (b) Depth-limited DFS.

   (c) N/A.

2. (a) [30%] 26 marks for correctness — correct solution paths, reasonable performance statistics, reasonable implementation strategies, correct logic, etc.

   4 marks for style — whether you make an effort to make your code readable (comments, style, etc).

   (b) [8%] 7 marks correctness, 1 mark style.

   No mark will be deducted if you give up solving puzzle #2 because of its overwhelming computational demand. 2% bonus will be awarded to those who have the patience.

3. (a) [8%] 7 marks correctness, 1 mark style.

   (b) [16%] 14 marks correctness, 2 mark style.

4. (a) [6%] In the original sliding-block puzzle problem, a tile at position $\langle i, j \rangle$ can be moved to position $\langle k, l \rangle$ if the following preconditions are satisfied:

      i. $\langle i, j \rangle$ is adjacent to $\langle k, l \rangle$;

      ii. $\langle k, l \rangle$ is empty;

   The relaxed adjacency problem can be obtained by deleting the first precondition. (Note that the Manhattan distance heuristics is obtained by deleting the second precondition, and the misplaced tiles heuristics is obtained by deleting both preconditions.) The state space of the relaxed adjacency problem is therefore an edge-supergraph of the original sliding-block puzzle problem.

   (b) [6%] Consider the following observation:

   Observation 1: Tiles are moved only by line 6 and 8 of the algorithm. Moreover, only misplaced tiles are moved.

   As a result, once a tile is in its goal position, it will no longer be moved by the algorithm.

   Observation 2: Line 6 moves a misplaced tile to its goal position, while line 8 moves a misplaced tile to another non-goal position.

   By observations 1 and 2, the number of times line 6 is executed equals the number of misplaced tiles in the input puzzle.

   Observation 3: After line 8 is executed, line 6 will be executed in the immediately following iteration.

Consequently, line 8 is executed no more frequently than line 6. The total number of moves made by the algorithm is therefore no more than 2 times the number of misplaced tiles in the input puzzle, that is, no more than $2 \times M$.

(c) [8%] 7 marks correctness, 1 mark style.

5. [6%] If your implementation is correct, you will observe figures similar to the following (1% for the table):

Strategy	Puzzle #1	Puzzle #2
Mysterious	2182	—
Mysterious + `cycle-p`	146	31680840
Misplaced tiles + `cycle-p`	15	5748
Relaxed adjacency + `cycle-p`	15	3235
Manhattan distance + `cycle-p`	9	78

The following observations are expected to be picked up (1% each):

(a) DFS is a *blind* search, and it does not make use of any *domain information/knowledge* to guide its node expansion strategy. Its performance is the worst.

(b) Cycle checking *prunes* away numerous repeated states.

(c) The Manhattan distance heuristics *dominates* the misplaced tiles heuristics, and therefore yields better performance.

(d) The relaxed adjacency heuristics *dominates* the misplaced tiles heuristics, and therefore yields better performance.

(e) Although the relaxed adjacency heuristics dominates the misplaced tiles heuristics, the theory only promises that the number of nodes expanded by the former heuristics is *no more* than that by the latter. As a result, the two heuristics *expands the same number of nodes* in certain cases.

(f) Although the Manhattan distance and relaxed adjacency heuristics are not comparable, *empirical results* suggest that Manhattan distance is usually a much superior choice.

(g) The effect of pruning and heuristics *magnify exponentially* when problem size increases. This is due to the *exponential growth rate* of search truee.

6. [6%] The Independent Set Problem.

(a) [3%] Let the propositional symbol p_i indicates if node i of the given graph is covered. The CNF formula contains the following clauses:

- To guarantee that no two adjacent nodes are covered, we include $(\neg p_i \vee \neg p_j)$ for each pair of adjacent nodes i, j. Note that $(\neg p_i \vee \neg p_j) = (p_i \Rightarrow \neg p_j) = (p_j \Rightarrow \neg p_i)$.

- To guarantee that at least M of the N nodes are covered, we include $(p_{i_1} \vee p_{i_2} \vee \ldots \vee p_{i_{(N-M+1)}})$ for every $(N-M+1)$-subset of nodes $\{i_1, i_2, \ldots, i_{(N-M+1)}\}$. Note that at least M of the N nodes are covered iff a random selection of $N-M+1$ nodes always includes a covered one.

(b) [2%] There are $\begin{pmatrix} N \\ 2 \end{pmatrix}$ clauses of the first type, and $\begin{pmatrix} N \\ N-M+1 \end{pmatrix} = \begin{pmatrix} N \\ M-1 \end{pmatrix}$ clauses of the second type. The maximum size for each clause is $M-N+1$.

(c) [1%] Since the CNF is large, GSAT should be used.

7. [6%] Note that every wff in *IMP* has the form $p \Rightarrow q$. The following algorithm decides if $\Gamma \vdash (p \Rightarrow q)$:

   (a) We construct the following graph. For each propositional symbol p' occurring in either Γ or $p \Rightarrow q$, create a node with label p'. For each wff $p' \Rightarrow q' \in \Gamma$, create a directed edge from node p' to node q'.

   (b) Use either BFS or DFS to check if there is a path from node p to node q. Return "YES" if such a path exists, and "NO" otherwise.

   Each of the above steps takes $O(N+E)$ time to complete, where N is the number of propositional symbols occurring in Γ and $p \Rightarrow q$, and E is the number of formulas in Γ.