CPSC 322 2018W2 Assignment 1

Make sure you follow all assignment instructions on the course website and in the assignment description on Canvas. Failure to do so may result in heavy penalties.

Make sure that in your answers you clearly indicate the exact section you are answering.

Please start each question on a new page (eg. don't put your answer to Q3 on the same page as your answer to Q2).

Question 1 (27 points): Comparing Search Algorithms

Consider the problem of finding the shortest path from *a* to *z* in the following directed graph:

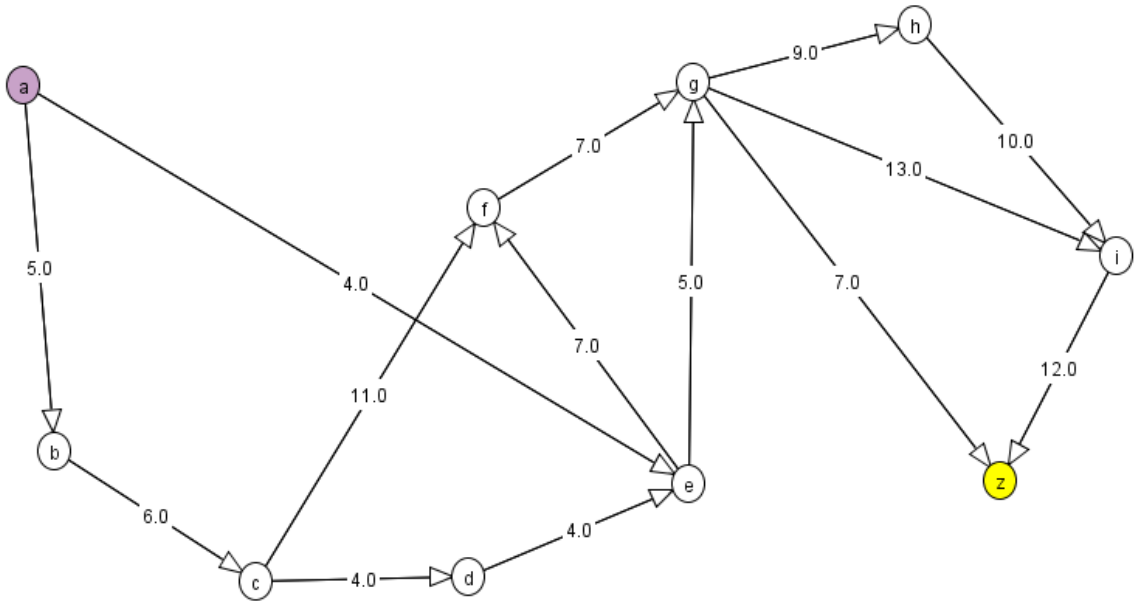


Figure 1. Graph for question 1. The arcs are labeled with their costs.

Start at node **a** and use **z** as the goal node. **Expand neighbours of a node in alphabetical order**, and **break ties in alphabetical order** as well. For example, suppose you are running an algorithm that does not consider costs, and you expand **a**; you will add the paths <*a*,*b*> and <*a*,*e*> to the frontier in such a way that <*a*,*b*> is expanded before <*a*,*e*>.

Additional notes:

- A node is expanded when the algorithm checks to see if it is a goal node, then returns the path if it is (or adds its neighbor paths to the frontier if it is not).
- If a given node is expanded multiple times, it should appear in your list of expanded nodes multiple times.

For the algorithms in questions 1.1-1.4, answer the following questions:

(a) What nodes are expanded by the algorithm? Order the nodes from first expanded to the last. *(Note that a given node may be listed multiple times.)*
(b) What path is returned by the algorithm?
(c) What is the cost of this path?

1.1. **[5 points]** Depth-first search
1.2. **[5 points]** Breadth-first search
1.3. **[5 points]** A* (use the uninformative but admissible heuristic $h(x) = 0$ for all nodes)
1.4. **[5 points]** Branch-and-bound, a.k.a B&B (again, use $h(x) = 0$ for all nodes)

1.5. **[7 points]**

(a) [1 point] Did BFS and B&B find the optimal solution for this graph?
(b) [3 points] Are BFS and B&B optimal in general? Explain your answer.
(c) [3 points] Did B&B expand fewer nodes than A*? Explain if your answer is true in general for these two algorithms and why.

Question 2 (36 points): Uninformed Search: Peg Solitaire

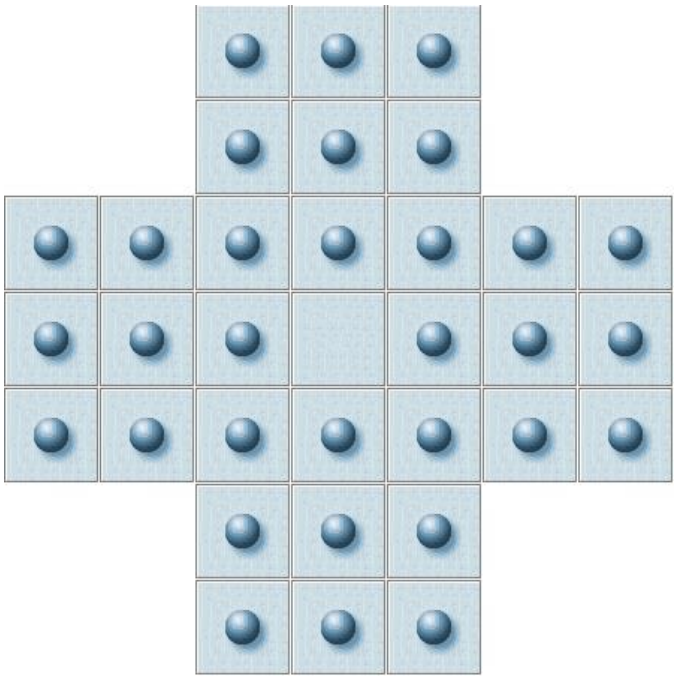


Figure 2. A sample Peg Solitaire board at the start of the game

Peg Solitaire is a board game for one player involving movement of pegs on a board with holes. The standard game fills the entire board with pegs except for the central hole. The objective is to empty the entire board except for a solitary peg in the central hole, by making valid moves. A valid move is to jump a peg orthogonally over an adjacent peg into a hole two positions away and then to remove the jumped peg.

From the Wikipedia entry: "Peg Solitaire"

You can play the game at http://www.novelgames.com/en/spgames/peg/

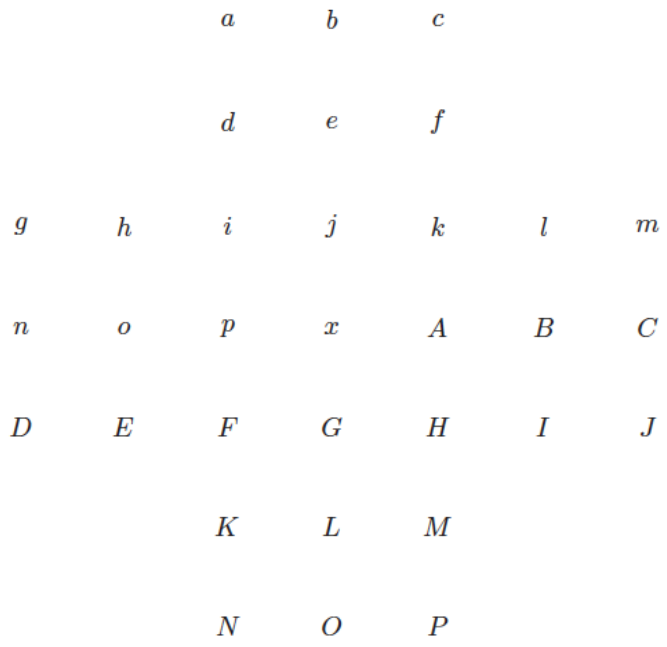


Figure 3. Peg Solitaire Board

2.1. **[12 points]** Represent peg solitaire as a search problem. (You may use the labels provided in Figure 3 for referring to spaces on the board if you wish).

(a) **[4 points]** How would you represent a node/state?
(b) **[2 points]** In your representation, what is the goal node?
(c) **[3 points]** How would you represent the arcs?
(d) **[3 points]** How many **possible** board states are there? **Note:** this is **not** the same as the number of "valid" or "reachable" game states, which is a much more challenging problem.

2.2. **[12 points]** The search tree:

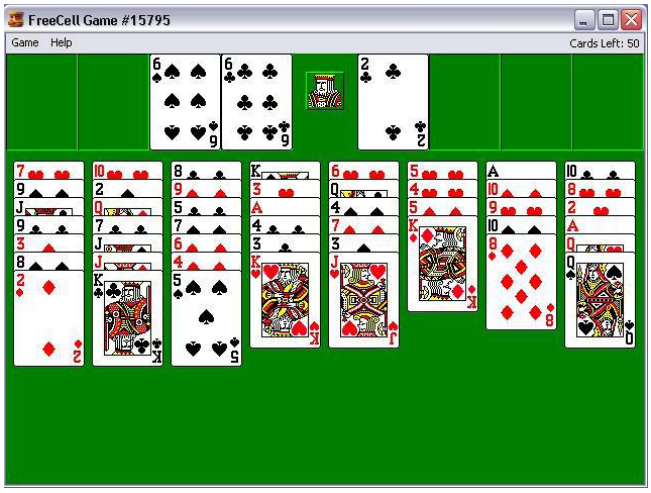
(a) **[9 points]** Write out the first three levels (counting the root as level 1) of the search tree. (Only label the arcs; labeling the nodes would be too much work).
(b) **[3 points]** What can you say about the length of the solution(s)?

2.3. **[12 points]** The search algorithm:

(a) **[4 points]** What kind of search algorithm would you use for this problem? Justify your answer.
(b) **[4 points]** Would you use cycle-checking? Justify your answer.
(c) **[4 points]** Would you use multiple-path-pruning? Justify your answer.

Question 3 (24 Points) Free Cell

Consider the problem of trying to solve a game of FreeCell in the minimum number of moves. You can play the game at http://www.freecell-cardgame.com/.
FreeCell comes standard on nearly every version of Microsoft Windows (including Windows 10, under the Microsoft Solitaire Collection), and full rules are also available at the Wikipedia entry:
https://en.wikipedia.org/wiki/FreeCell



The object of FreeCell is to move all the cards to the home cells [foundations], using the free cells as placeholders. To win, make four stacks of cards on the home cells [foundations], one for each suit, stacked in order of rank, from lowest (ace) to highest (king).

- *When moving cards to columns, cards must be moved in order from highest (king) to lowest (ace), alternating suit colors.*
- *When moving cards to home cells [foundations], cards must be moved in order from lowest (ace) to highest (king), same suit.*
- *A card from the bottom of a column can move to a free cell, the bottom of another column, or a home cell [foundation].*
- *A card from a free cell can move to the bottom of a column, or to a home cell [foundation].*

Microsoft Windows FreeCell help file

3.1. **[15 points]** Represent this as a search problem.
(a) **[7 points]** How would you represent a node/state?
(b) **[3 points]** In your representation, what is the goal node?
(c) [5 points] How would you represent the arcs?

3.2. **[9 points]** Give an admissible heuristic for this problem; explain why your heuristic is admissible. More points will be given for tighter lower bounds; for example, h=0 is a trivial (and useless) heuristic, and thus it is not acceptable.

Question 4 (15 points) Modified Heuristics

To answer this question, the AISpace search applet will be useful. It can be found at
http://aispace.org/search/
For this question you are to think about the effect of heuristic accuracy on *A** search. That is, you are to experiment using a sample graph, and think about how the closeness of *h(n)* to the actual distance from node *n* to a goal affects the efficiency and accuracy of *A**.
Use the sample graph- Vancouver Neighbourhood Graph- in AISpace as an example. Set *h(n)* as an underestimate. To achieve this, you should first set the cost values automatically by clicking on the option '*Set edge costs automatically*' and then set the h(n) values to an underestimate by clicking on the option '*Set node heuristic automatically*' in the '*Search Options*' menu of the applet (***you need to be in CREATE mode in order to do this***). See how *A** works with this setting and report the number of nodes expanded as well as the optimal path found.
Then you will need to experiment with changing the *h(n)* values as described below. You can change the *h*(n) value of individual nodes by clicking on '*Set Properties'* in Create mode and then right click on the desired node. Or you can change the *h*(n) value globally by selecting '*set costs and heuristics*' under "*search options*". Remember not to modify the costs.
Notes:

1. You need to set the cost values for this problem only once when you load the problem by selecting 'Set edge costs automatically' option. You will use these values throughout this question.
2. You need to reset the *h(n)* values to an underestimate using the '*Set node heuristic automatically*' as described above before each observation for this question.

4.1. [6 points 2+2+2] Reduce *h*(n), trying in three different ways:

- First, subtract a large constant from all *h(n)*s; make sure all h values remain positive
- Second, only reduce *h(n)*s of the nodes on the solution path.
- Last, only reduce *h(n)*s of the nodes which are not on the solution path.
- *Note that the goal node is to be exempted from these changes, since its h(n) is already 0.*

Explain the effects of these changes, specifically:
(a) When can *A** still find the optimal path?
(b) When is the efficiency of *A** improved or reduced?
(c) Try to draw a more general conclusion regarding the changes in efficiency and optimality.

4.2. [3 points 1+1+1] Set *h*(n) as the exact distance from n to a goal. To do this, you should add the costs of arcs on the optimal path from every node to the goal node, and set the sum as the heuristic function of the node.
(a) Can A* still find the optimal path?
(b) Is the efficiency of *A** increased or reduced?
(c) Try to draw a more general conclusion regarding the changes in efficiency and optimality.

4.3. [6 points: 2+2+2] Increase *h*(n), also trying in three different ways.

- First, add a large constant to all *h(n)*s;
- Second, only increase *h(n)*s of the nodes on the solution path
- Last, only increase *h(n)*s of the nodes which are not on the solution path.

Discuss the effects of these changes, specifically:
(a) When can *A** still find the optimal path?
(b) When is the efficiency of *A** improved or reduced?
(c) Try to draw a more general conclusion regarding the changes in efficiency and optimality.