CMPUT 366, Winter 2021
Assignment #1

Due: Monday, Feb. 8, 2021, 11:59pm
Total points: 150

For this assignment use the following consultation model:

1. you can discuss assignment questions and exchange ideas with other *current* CMPUT 366 students;
2. you must list all members of the discussion in your solution;
3. you may **not** share/exchange/discuss written material and/or code;
4. you must write up your solutions individually;
5. you must fully understand and be able to explain your solution in any amount of detail as requested by the instructor and/or the TAs.

Anything that you use in your work and that is not your own creation must be properly cited by listing the original source. Failing to cite others' work is plagiarism and will be dealt with as an academic offence.

First name: Weixi

Last name: Cheng

CCID: weixi`@ualberta.ca`

Collaborators: ____________________

1. **(Uninformed search)**

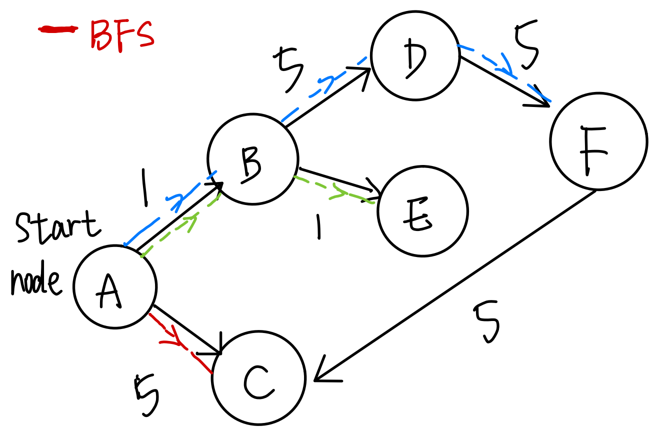
   (a) [**15 points**] Construct a graph search problem with **no more than 10 nodes** for which all of the following are true:

      i. Least-cost search returns an optimal solution.
      ii. Depth-first search returns the highest-cost solution.
      iii. Breadth-first search returns a solution whose cost is strictly less than the highest-cost solution and strictly more than the least-cost solution.

   Note that this means your search problem must have at least 3 goal nodes of differing costs. Be sure to list the start and goal node(s), all edge costs and all edge directions (if your graph is directed). Draw the graph as well.

LCS
DFS

Start node : A



i. For LCS, we have goal node : E
Path : A → B → E
cost : 1 + 1 = 2

ii. For DFS, we have goal node : F
Path : A → B → D → F
Cost : 1 + 5 + 5 = 11

iii. For BFS, we have goal node : C
path : A → C
cost : 5

   (b) [**5 points**] List the paths in the frontier at each step of a depth first search of the problem you specified in part (1a). Also, highlight the path that will be removed from the frontier in that step. Stop when the path removed ends in a goal state.

Goal : F. Selection : Newest

Step1. {<A>}
Step2. {<A, B>, <A, C>}
Step3. {<A, B, D>, <A, B, E>, <A, C>}
Step4. {<A, B, D, F>, <A, B, E>, <A, C>}
path

   (c) [**5 points**] List the paths in the frontier at each step of a breadth first search of the problem you specified in part (1a). Also, highlight the path that will be removed from the frontier in that step. Stop when the path removed ends in a goal state.

Goal : C. Selection : Oldest

Step1. {<A>}
Step2. {<A, B>, <A, C>}
path

   (d) [**5 points**] List the paths in the frontier at each step of a least cost search of the problem you specified in part (1a). Also, highlight the path that will be removed from the frontier in that step. Stop when the path removed ends in a goal state.

Goal : E. Selection : Cheapest

Step1. {<A>}
Step2. {<A, B>, <A, C>}
Step3. {<A, B, E>, <A, C>}.
path

2. **(Heuristic search)**

A farmer needs to move a hen, a fox, and a bushel of grain from the left side of the river to the right using a raft. The farmer can take one item at a time (hen, fox, or bushel of grain) using the raft. The hen cannot be left alone with the grain, or it will eat the grain. The fox cannot be left alone with the hex, or it will eat the hen. For example, the farmer cannot move from one side x of the river to the other side y if it would mean leaving the fox and hen together on side x.

The farmer can load an item onto the raft, move the raft from one side of the river to the other, or unload an item from the raft. The farmer wants to move the items with the fewest number of trips across the river as possible, but does not care about how much time is spent loading or unloading.

(a) [**6 points**] Classify this problem using the primary representational dimensions from lecture 2.

1. Uncertainty:
   1. Fully observable
   2. Deterministic dynamics
2. Interaction: Offline.
3. Number of agents: Single agent

(b) [**20 points**] Represent this problem as a graph search problem. Be sure to include and formally describe each component of the search problem.

Graph Search: find a path from start to goal

States: { in the left/right side of the river with a hen, a fox and a bushel of grain, in the left/right side of the river with a fox and a bushel of grain, in the left/right side of the river with a hen }

Start state: in the left side of the river with a hen, a fox and a bushel of grain

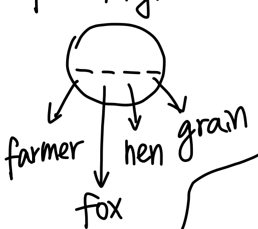
Actions: { go left/right with a hen, go left/right with a fox, go left/right with a bushel of grain, go alone }

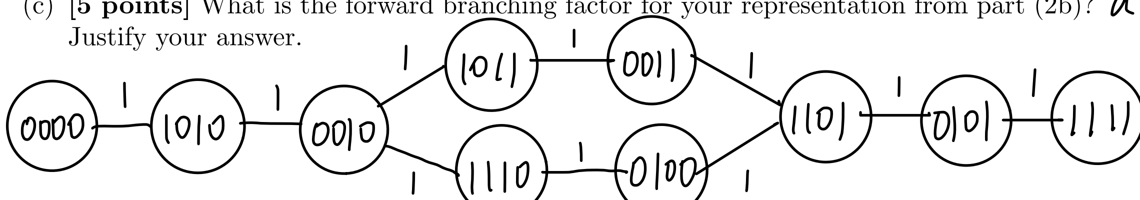
Successor function: 1. one side = fox, grain, other side = hen 2. left = x, raft = y, right = z

goal function: goal(state): (state == in the right side of the river with a hen, a fox and a bushel of grain)

(c) [**5 points**] What is the forward branching factor for your representation from part (2b)? Justify your answer.

0 – left ⇒
1 – right





(d) [**10 points**] Construct a non-constant admissible heuristic for this problem.

Suppose there are n items in the left side of the river, since the farmer can only take one item at a time, ignore all restrictions, we know that
1. when the farmer starts from the left side of the river, it will take at least 2n-1 actions
2. - - - - - - - - - - - right - - - - - - - - - 2n actions

(e) [**5 points**] Argue that the heuristic from part (2d) is admissible.

Yes, it is admissible!

from (2d), we get $h(n) = \begin{cases} 2n-1 & \text{(start from left)} \\ 2n & \text{(start from right)} \end{cases}$ actions

resource: https://math.stackexchange.com/questions/3096904/farmer-hex-fox-and-grain-problem-creating-an-admissible-heuristic-via-graph-s

Since a heuristic function is admissible if h(n) is always less than or equal to the cost of the cheapest path from n to any goal node, from the question, we know hen can not be with grain or fox alone, so that the farmer must take more actions than h(n) because of these restrictions.

(f) [**60 points**] Implement your representation from part (2b) and heuristic from part (2d) in Python 3 by editing the `River_problem` class in the provided `riverProblem.py`. We will run your code with the command `python3 riverProblem_run.py`. Your code must complete within 2 minutes for full marks.[1]

Submit all of your code (including provided boilerplate files) in a single zip file.

3. **(Local search)**

(a) [**6 points**] For each of the following problems, state whether graph search or local search is a more appropriate algorithm, and justify your answer.

i. Solving a Rubik's cube:

Local Search

Since we do not know what actions get us from the start to the goal. Only that we find it in the end.

ii. Solving a maze map:

Graph Search

Since we are trying to get out of the maze map and reach a goal, it needs to figure out what sequence of actions will take us there.

iii. Solving a Sudoku problem:

Local Search

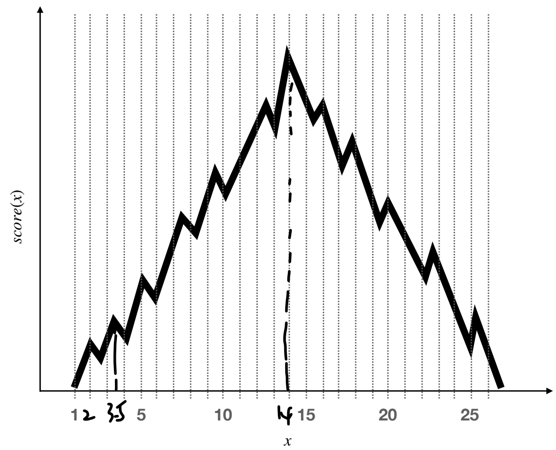
Since we have many choices to start, there are large state spaces.

(b) [**2 points**] Is hill climbing a **complete** algorithm? Why or why not?

No. Since it will not move to a lower value and it may get stuck on the local maximum, it is incomplete.

[1] It should run in far less time than this.

Consider the a constraint optimization problem over a single variable $x \in \mathcal{X} = \{0.1, 0.2, \ldots, 26.9, 27\}$, with cost graph as in the following figure:



(c) [**3 points**] Is hill climbing on the above problem with *neighbourhood* defined as

$$neighbourhood(x) = \{y \in \mathcal{X} \mid x - 0.5 \leq y \leq x + 0.5\}$$

an **optimal** algorithm? Why or why not?

No.

Since 0.5 is too small, we may not find a higher value of x with this neighbourhood(x).

For example, when $x=3.5$, $3 \leq y \leq 4$.

(d) [**3 points**] Is hill climbing on the above problem with *neighbourhood* defined as

$$neighbourhood(x) = \{y \in \mathcal{X} \mid x - 2 \leq y \leq x + 2\}$$

an **optimal** algorithm? Why or why not?

Yes.

Since 2 is more suitable for defining neighbourhood(x), we can more easily find a higher value.

Submission

The assignment you downloaded from eClass is a single ZIP archive which includes this document as a PDF *and* its LaTeX source as well as Python files needed for Question 2f. You are to unzip the archive into an empty directory, work on the problems and then zip the directory into a new single ZIP archive for submission.

Each assignment is to be submitted electronically via eClass by the due date. **Your submission must be a single ZIP file containing**:

1. a single PDF file with your answers;
2. file(s) with your Python code.

To generate the PDF file with your answers you can do any of the following:

- insert your answers into the provided LaTeX source file between `\begin{answer}` and `\end{answer}`. Then run the source through LaTeX to produce a PDF file;
- print out the provided PDF file and legibly write your answers in the blank spaces under each question. Make sure you write as legibly as possible for we cannot give you any points if we cannot read your hand-writing. Then scan the pages and include the scan in your ZIP submission to be uploaded on eClass;
- use your favourite text processor and type up your answers there. Make sure you number your answers in the same way as the questions are numbered in this assignment.