CMPUT 366, Winter 2022
Assignment #1

Due: Friday, Feb. 4, 2022, 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: Peng

Last name: Cheng

CCID: pcheng1@ualberta.ca

Collaborators: _____

1. **(Uninformed search)**

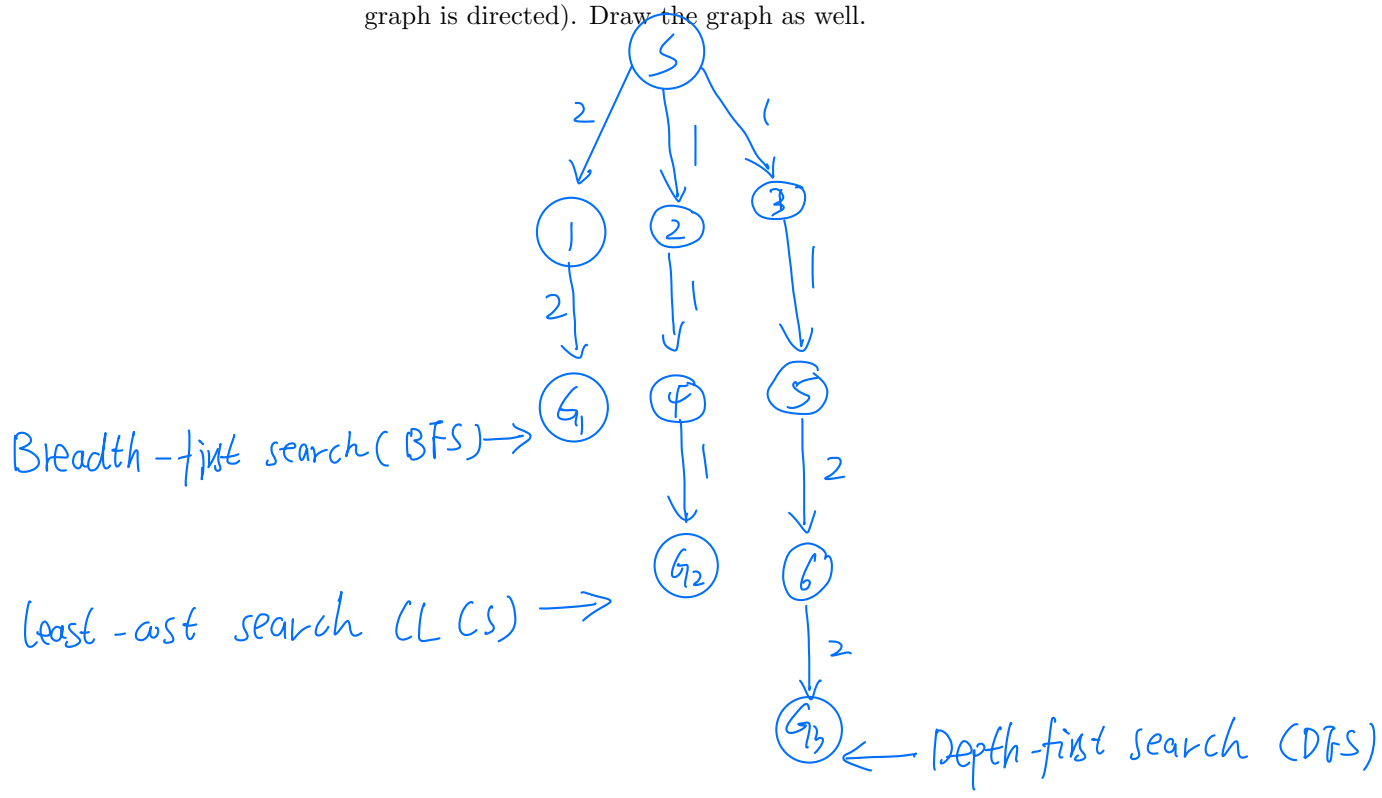
   (a) **[15 points]** Construct a graph search problem with **no more than 10 nodes** for which all of the following are true:

   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 solutions 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.



all edges cost:

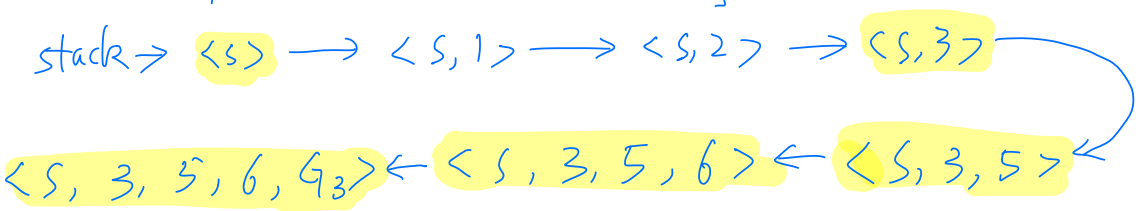
- DFS: 4 — S →1 1 →2 3 →2 G3
- LCS: 3 — S →1 4 →1 5 →1 G2
- BFS: 6 — S →2 6 →2 G1

(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.

we assume DFS always go to right

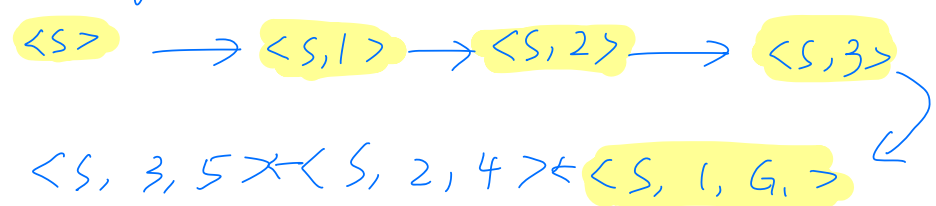
Since it use stack:

frontier = { <s> | s is a start node}



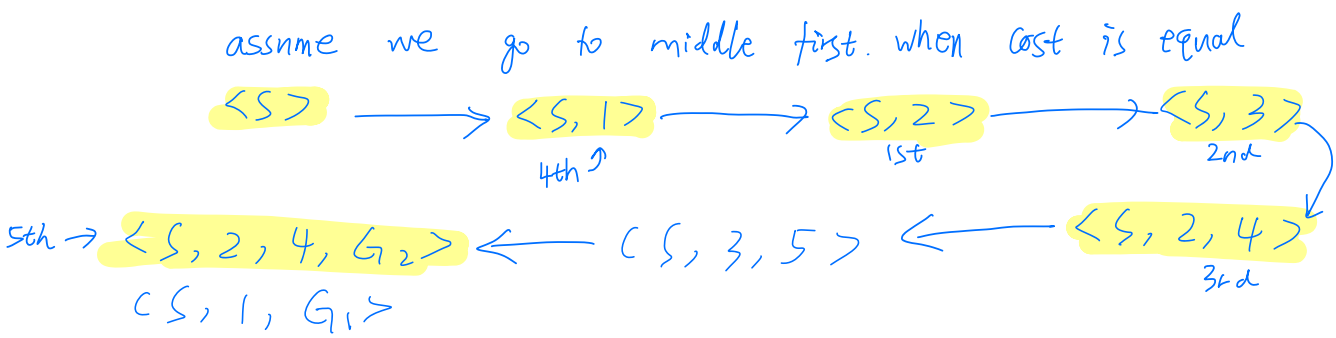
(c) **[5 points]** List the paths in the frontier at each step of a breadth first search of the problem you specified in part (1a). Also, highlight the path that will be removed from the frontier in that step. Stop when the path removed ends in a goal state.

assume go to right first



(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.

assume we go to middle first. when cost is equal



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.

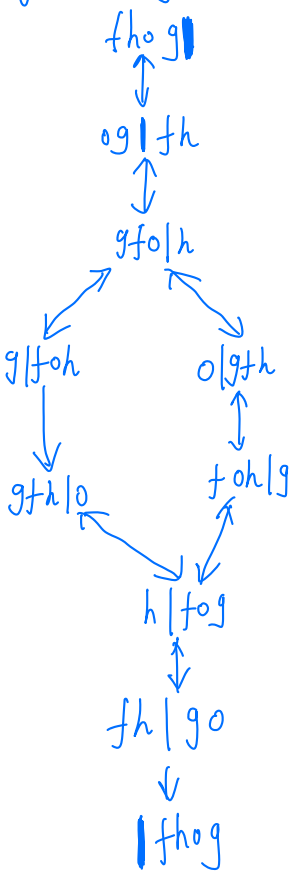
The farmer can load an item onto the raft, move the raft from one side of the river to the other, or unload an item from the raft. The farmer wants to move the items with the fewest number of trips across the river as possible, but does not care about how much time is spent loading or unloading.

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

Uncertainty: deterministic dynamics, fully observable
Interaction: offline
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.

farmer_x : f
hen_x : h
fox_x : o
grain_x : g



| is river.

start state : (farmer_x, hen_x, fox_x, grain_x)
goal state : (farmer_y, hen_y, fox_y, grain_y)
set of state: each node is named stringName_side , stringName is 4 names, side is river side x or y.
action : go to side x and go to side y
edges :
(farmer_x, hen_x, fox_x, grain_x |) farmer takes hen to y side
(fox_x, grain_x, | farmer_y, hen_y) farmer returns alone to x side
(farmer_x, grain_x, fox_x, | hen_y) farmer takes fox to y side
(grain_x | farmer_y, fox_y, hen_y) farmer returns back with hen
(farmer_x, hen_x, grain_x, | fox_y) farmer takes grain to y side
(hen_x, | fox_y, fox_y, grain_y) farmer returns back alone
(hen_x, farmer_x, | fox_y, grain_y) farmer takes hen to y side.
(farmer_y, hen_y, fox_y, grain_y)

| is river left is x, right is

edges cost: increase 1

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

In the start state, factor is highest equal to 2
If fox and hen in one side, or if hen and grain in one side, the branching factor is 0 since fox eats hen.
Thus the range of branching factor is [0, 2]

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

In this case, fox cannot be left alone with hen,
hen cannot be left alone with grain;
farmer can only take one item at a time.

we need to drop one constraint:
two items can stay on the same side. (fox, hen) (hen, grain)

Thus, there are only 5 or 6 moves (farmer on left or right side)
(hen_x, grain_x, | farmer_y, fox_y) farmer takes fox to y side
(hen_x, grain_x, farmer_x, | fox_y) farmer returns alone
(grain_x, | farmer_y, hen_y, fox_y) farmer takes hen to y side
(grain_x, farmer_x, | hen_y, fox_y) farmer returns alone
(| farmer_y, grain_y, hen_y, fox_y) farmer takes grain to y side
| is river, left is x, right is y

code:
If farmer on side x:
h(n) = 2 · (number of items in side x) − 1
else:
h(n) = 2 · (number of items in side x)

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

2d is admissible because h(n) is less than or equal to the real cost.
The real cost is always bigger than h(n), since we ignore the truth that fox will eat hen. If we do not ignore that truth, the real cost will higher.

(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:

graph search, since we know the goal state is same color in one surface.

ii. Solving a maze map:

graph search, since we know the start state is entrance, goal state is exit.

iii. Solving a Sudoku problem:

local search, since we do not know the goal state (answer) we will try to solve it.

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

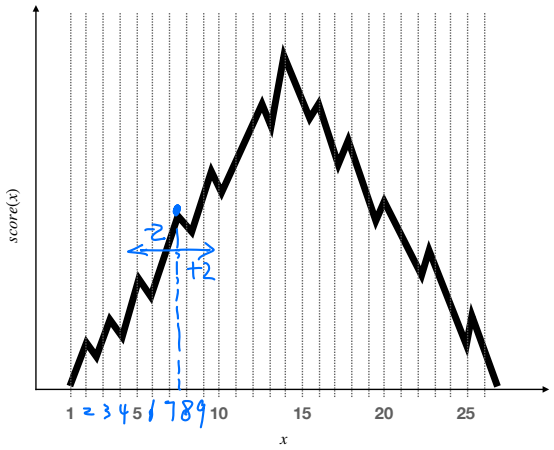
No
Since we will meet some problem such as local maxima, plateau

In the local maxima, it will make worse when all moves appear

It also will not change when in the plateau.

[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,
We can not find global maxima between -0.5 and 0.5.
In the figure above, the range between column is 1,
the given range 1 is too narrow.

(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.
The given range is wide enough to find global maxima.

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.