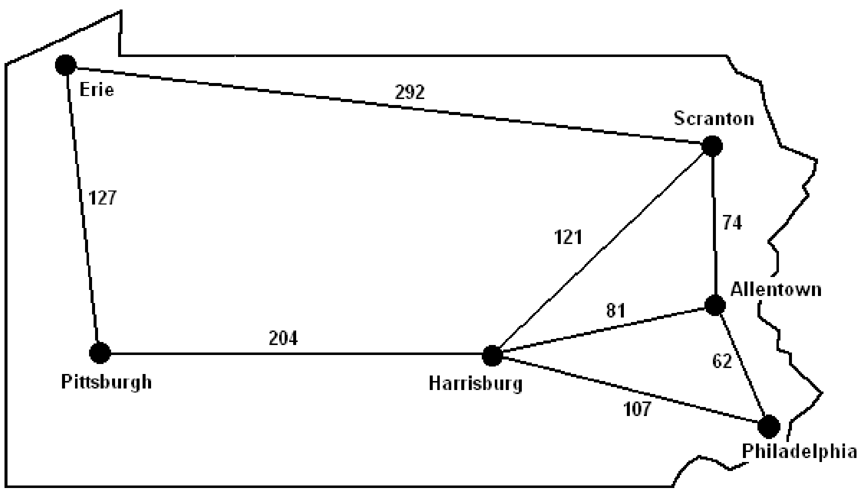
Homework #2: Chapters 3 and 5

The following exercises are due at the beginning of class on **Friday, February 18.** This assignment includes an extra-credit assignment worth up to an additional 20 points.

1. *[15 points]* A hurried traveler is seeking an efficient route across Pennsylvania, from Philadelphia to Erie. Use Greedy Best-First search to find a route, where step costs are as labeled on the map and the values for the heuristic function are given by the table. Show your search tree, including the h(n) value for each node, and label each node with the order in which it is expanded (note, this may be different from the order it is generated). In order to reduce unnecessary search, you can use graph-based search.



Heuristic Estimates		
Erie	Pittsburgh	100
Erie	Harrisburg	225
Erie	Scranton	250
Erie	Allentown	300
Erie	Philadelphia	350
Pittsburgh	Harrisburg	180
Pittsburgh	Scranton	225
Pittsburgh	Allentown	200
Pittsburgh	Philadelphia	225
Harrisburg	Scranton	100
Harrisburg	Allentown	60
Harrisburg	Philadelphia	85
Scranton	Allentown	65
Scranton	Philadelphia	100
Allentown	Philadelphia	55

2. *[15 points]* Now repeat the exercise above, but use A* instead of greedy best-first. Show your search tree, complete with f(n), g(n) and h(n) values for each node, and label each node with the order in which it is expanded. Once again, use a graph search approach, but note that if a generated node has a lower f(n) than a frontier node with an identical state, then it must still be explored.
3. *[20 points]* Use A* to solve the 8-puzzle with the initial and goal states shown below. Assume that your path cost is 1 per move and that your heuristic function is the Manhattan distance of all tiles from their correct placement (note, the blank does not count as a tile). Show your search tree, complete with f(n), g(n) and h(n) values for each node and label each node with the order in which it is expanded. Also show the current game board at each node in the tree. Once again, when expanding nodes, assume that you can ignore actions that return you to the previous state.

Initial State

1		3
4	2	5
7	8	6

Goal State

1	2	3
4	5	6
7	8	

4. *[10 points]* The **heuristic path** algorithm (Pohl 1997) is a best-first search in which the evaluation function is $f(n) = (2 - w)g(n) + wh(n)$. For what values of w is this complete? For what values is it optimal, assuming that h is admissible? What kind of search does this perform for $w = 0$, $w = 1$, and $w = 2$?

5. *[40 points]* This problem looks at playing the game tic-tac-toe. Assume that X is the MAX player. Let the utility of a win for X be 10, a loss for X be -10, and a draw be 0. There are two parts to this question, each using one of the two game boards given below:

board1

	O	
X	O	X
	X	O

board2

		X
	X	O
O		

a) Given the game board **board1** above where it is X's turn to play next, show the entire game tree. Mark the utilities of each terminal state and use the minimax algorithm to calculate the optimal move.

b) Given the game board **board2** on the opposite page where it is X's turn to play next, show the game tree with a cut-off depth of two ply (i.e., stop after each player makes one move). Use the following evaluation function on all leaf nodes:
$\text{Eval}(s) = 10X_3(s) + 3X_2(s) + X_1(s) - (10O_3(s) + 3O_2(s) + O_1(s))$
where we define $X_n(s)$ as the number of rows, columns, and diagonals in state s with exactly n X's and no O's, and similarly define $O_n(s)$ as the number of rows, columns, and diagonals in state s with exactly n O's and no X's. Use the minimax algorithm to determine X's best move.

Extra Credit *(+20 points)*:
This optional exercise requires you to do some Java programming in order to conduct an experiment that compares uniform-cost, greedy best-first, and A* search. In order to do this, download and read the code I have made available from the course web site (under Additional Class Materials). This code implements all three search algorithms and provides two *abstract* classes: SearchProblem and State. In order to solve a particular problem, you only need to extend these two classes with details specific to your problem.

Your task is to extend the code to solve the following three configurations of the 8-puzzle, where the goal state for all three configurations is as specified to the far right.

Initial State #1

7	1	4
6	3	2
	8	5

Initial State #2

4	8	2
6	3	5
1		7

Initial State #3

7	5	3
6		4
8	1	2

Goal State

1	2	3
8		4
7	6	5

You will then run each of the three search algorithms on each puzzle, recording the path cost of the solution (if found), number of nodes expanded, number of nodes generated, and time to perform the search. To do this, you'll need to write a class that extends State and can record the state of the game (i.e., the current position of each tile). It is important that this class implements an equals() method that can be used to compare the current state to another state. You'll need to write a second class that extends SearchProblem and implements the four methods: getInitialState(), goalTest(), getSuccessors(), and getHeuristicValue(). You may want to include a constructor that allows you to initialize the class with different initial states. Note, getSuccessors() returns a list of Successor elements, where each Successor records the State, a string describing the action to reach it, and the step cost of executing that action. This one method effectively determines the ACTIONS(s), RESULT(s,a) and makes it easy to compute g(n) for the generated node. For the heuristic, use the sum of the Manhattan distances of all tiles from their goal positions. Finally, you'll need to write a main() method that runs the tests.

After you collect your data, write an analysis of it. What appears to be the strengths and weaknesses of each algorithm based on your experiment? Did the experimental results agree with the theoretical properties of the algorithms discussed in class and in the book? What, if anything, surprised you?

Attach a hardcopy of your code, the output of your experiment, and your analysis to your homework submission. Submit your source (.java) and compiled (.class) files to via e-mail to heflin@cse.lehigh.edu with subject line: "CSE 327: HW #2 Extra Credit".