CS490 Quiz 2

NAME: ____________________________

STUDENT NO: ______________________

SIGNATURE: ___________________________

This is the written part of Quiz 2. The quiz is closed book; in particular, no notes, calculators and cell phones are allowed.

Not all questions are of the same difficulty, so be sure to pick up the easy points first, before tackling hard problems. The quiz is divided into two parts:

Part 1: [15]

Answers in part 1 should be succinct. Erroneous "extra" comments may decrease your score.

Part 2: [30]

When designing algorithms in part 2, the use of pseudocode is greatly encouraged! Feel free to quote algorithms we described in class, but make sure you describe how the algorithm is used. (e.g. to quote Dijkstra, you should specify your source node and where the results are stored, say an array. Then, go on to explain what you do with those results.) You can also quote the complexities of the algorithms discussed in class

NOTE: feel VERY free to use the backside of the exam papers.

Stats:

	A1	A2	A3	B1	B2	B3	A total	B total	total
High	5.0	4.5	5.0	10.0	8.0	7.0	14.5	22.5	35.5
Low	3.0	2.0	3.0	3.0	3.0	0.0	8.5	6.0	20.5
Median	5.0	2.5	5.0	5.0	7.0	5.0	12.5	16.0	27.5
Average	4.5	2.9	4.7	5.0	6.4	3.6	12.1	14.9	27.1

Everyone gets plus 5 to the final written score due to B1.

Part 1 - Short Answers [5 points each]:

1. State an advantage and a disadvantage of iterative deepening (or equivalently, a “gain” and a “loss”). Why would we want to use iterative deepening?

   [3] Allows us to do BFS without storing the all the states with the same level in memory.
   [2] Takes more time than regular BFS (re-compute shallow nodes in each iteration)
   Used when we have a huge graph (with high branching factor) that won’t fit in memory

2. Given a tree with n nodes, describe an algorithm that will return the maximum matching, as well as provide an example match that achieves the maximum matching. Analysis the time complexity. (Recall: a matching is a set of edges such that no vertex is touched by two edges in the set. The maximum matching is the maximum cardinality of a matching.) O(n^2) algorithm is enough for full marks (O(n) is possible).

   [3] Make tree rooted with DFS/BFS O(n), and separate even and odd depth nodes.
   [2] This gives bipartite graph, and bipartite matching is O(n^2), since O(n) to find augmenting path, and at most n augmenting paths.
   ----
   [5] Run DP. Again root tree with DFS/BFS O(n). At each node, we get two states:
      It is matched with parent -> all its children won’t be matched with their parent
      It isn’t matched with parent -> pick a children to match to.
   O(n) since there are n nodes, and DP for each node is O(edges out of the node), and there are n edges in total.
   ----
   [5] Pick any leaf and match to parent. Repeat (O(n) to pick a leaf, O(n) possible max matching). Note: can be O(n) if done smartly.

3. Given the map of his kingdom as a graph (vertices are towns, edges are roads), and two towns, A and B, King Donkey would like to know if he can send two armies at once from town A to town B. The two armies have to travel disjoint routes: i.e. they may not use the same road (edge) for traveling; otherwise the ill-tempered men may fight among themselves. Armies can however, share a town (towns are big enough, unlike the narrow roads). Given the graph, A, and B, describe an algorithm that solves King Donkey’s problem.

   [3] Run flow from A to B.
   [2] If flow >= 2 , then yes. Otherwise no.

Part 2 [10 points each]:

1. There’s a WarCraft tournament in town that consist of numerous matches. Each game is played between two players, and there’s always a winner (i.e. no ties). At the end of the tournament, the player who wins the largest number of games win the tournament (we may have people tie for champion). Note: due to time constraints, not everyone gets to play everyone else in the tournament, there’s around 50 players and 100 games.

   The matches are announced up front, and Ernie would like to know if he still has a chance of winning halfway through the tournament. Ernie knows the results of all the games that have been played. He also knows what games are coming up. Give an algorithm to determine if Ernie can still somehow win.

   (E.g. if Bert has beaten Elmo and Oscar before, and only two games are left: Bert vs. Ernie, Elmo vs. Oscar. Then, Ernie can’t hope to win)

   [3] Assume Ernie wins all games that he plays afterwards. And then calculate how many games can the other people win without getting more points then Ernie. If someone already has more points, break.

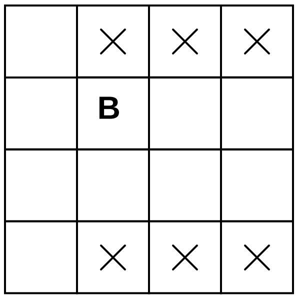
   [2] Mention flow or matching
   [3] Nodes: a start and end node, each player (except Ernie) is a node, and each game is a node. Edges: start node goes to players with edge weight equal to the number of games they can at most win. Players connect to games they participate in with edge weight 1. Each game then connect to the end node with edge weight 1.

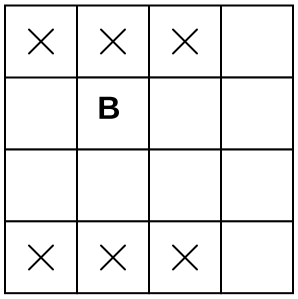
   [2] Flow = number of games played ⇔ there’s a way to pick a winner for all future games so that no one wins more games than Ernie.

   Flow < number of games played ⇔ the restriction that people can’t have more wins than Ernie stops us from assigning a winner to each game.

   Backtracking solutions get up to [5] marks, depending on how much speed ups are used.

2. Given a N by N chess board, we can place at most 2N-2 Bishops on it, e.g.





And that is the best we can do, since there are 2N-1 forward diagonals, but the first and last forward diagonals are opposite squares, and can only contain one bishop between the two.

Given N (up to 14), we can count how many different ways can we put down the 2N-2 bishops so that they don't attack each other? To simplify things, reflections and rotations count as different ways. (e.g. the two boards on the left are different ways.)

To make things interesting though, let us mark one of the positions BAD (**B**), which prevent us from putting a bishop on it. Given N and a bad square, how many ways can we put down 2N-2 non-attacking bishops?

Describe a general approach and any speed up you can think of (with enough detail so that someone can follow it and implement it). (Hint: Expect to do more than the n-Queens problem, since bishops don't attack vertically and horizontally, giving us less bounding. Also, there's more bishops involved.)

[3] Backtracking
[2] Try to place one bishop per forward diagonal.
[0.5] Need to pick the first or the last forward diagonal to NOT place a bishop.
(Or use number of bishops placed as terminating condition)
[2] Keep track of which backward diagonals are occupied during backtrack. Don't leave checking till the end.
[0.5] Never put things on the BAD square
[2] Do white bishops and black bishops separately, and just multiply their answers.

[up to 2 bonus] Any other useful bounds, speed ups. (e.g. don't place bishops on the four corners, but assume the two main diagonals are used. Then, multiply answer by 4)

OR

Places bishops on the short diagonals before long diagonals. Complexity is O(2^N) (N can go up to twenty something)

OR

Give analytically solution – discuss in class.

3. Think you got the right answer to 2? Try this one on for size!
   You are analyzing a communication grid of size N (up to 18). An N by N matrix of booleans describing the grid:

   element (i, j) = true ⇔ node i and j can communicate with each other

   Since a node cannot communicate with 2 or more nodes at the same time, describe an algorithm that return the maximum number of simultaneously communicating nodes, as well as an example of nodes communicating to each other that achieves the maximum.

[10] Give general matching algorithm.

OR

Combine Brute Force with other techniques!

[5] Try all splitting into two sets of vertices, and
[5] Run bipartite matching (remove edges between vertices of the same set)

OR

[5] For each edge both YES and NO as a match, and recurse of the subgraph.
[5] Memoize of the set of vertices left.

OR (partial marks)

[5] Try to match a pair of nodes. Erase the nodes, and recurse and backtrack.
O(N^2 * (N-2)^2 * …) = O(2^N * ((N/2)! ^2))
for N = 18, it is around 3*10^16..
[up to 2] Speed ups