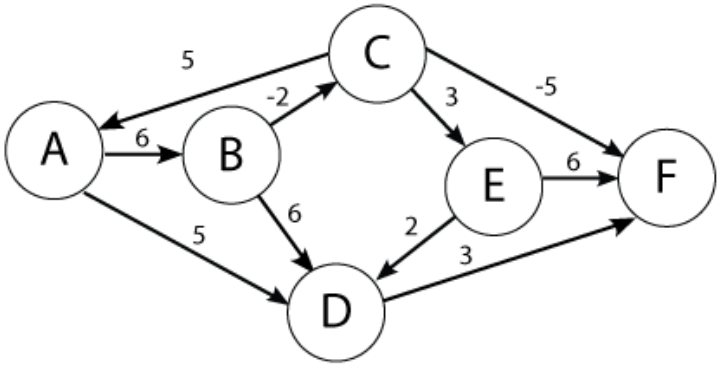
CSYE 7374
Autonomous Behavior and Learning in Games
Practice Exam One Solutions

Student Name: _____
Professor: Nik Bear Brown

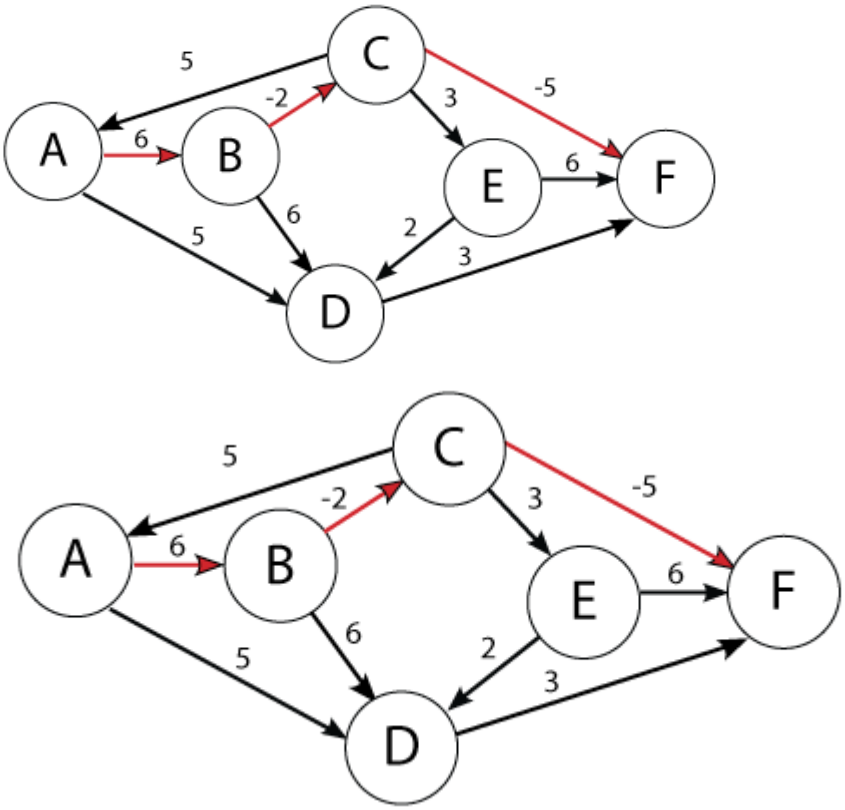
Rules:
1. NO COMPUTER, NO PHONE, NO DISCUSSION or SHARING.
2. Ask if you don't understand a question.
3. You may use one 8½"×11" sheets of notes (you may use both sides, written or printed as small as you like).
4. Time allowed. Until end of class.
5. Bring pen/pencil. The exam will be written on paper.

Q1 (10 Points)

Use the Bellman-Ford algorithm to find the shortest path from node A to F in the weighted directed graph above. *Show your work.*



Solution:



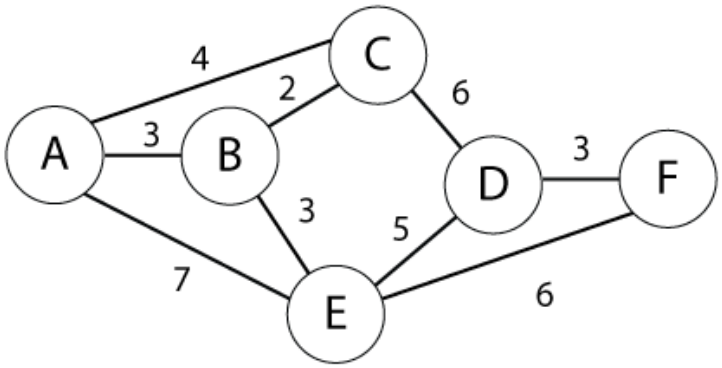
Shortest path: A->B->C-->F at cost -1

	A	B	C	D	E	F
0	0	INF	INF	INF	INF	INF
1	0	6	INF	5	INF	INF
2	0	6	4	5	INF	8 D
3	0	6	4	5	7	-1 C
4	0	6	4	5	7	-1
5	0	6	4	5	7	-1
6	0	6	4	5	7	-1

Many students calculate from F to A which is fine but the numbers in the table will be different even though the final path will be the same.

Q2 (10 Points)

Use Kruskal's algorithm to find a minimum spanning tree for the connected weighted graph below:



Solution:

Kruskal's algorithm pseudocode:

A.Create a forest T (a set of trees), where each vertex in the graph is a separate tree
B.Create a set S containing all the edges in the graph
C.Sort edges by weight

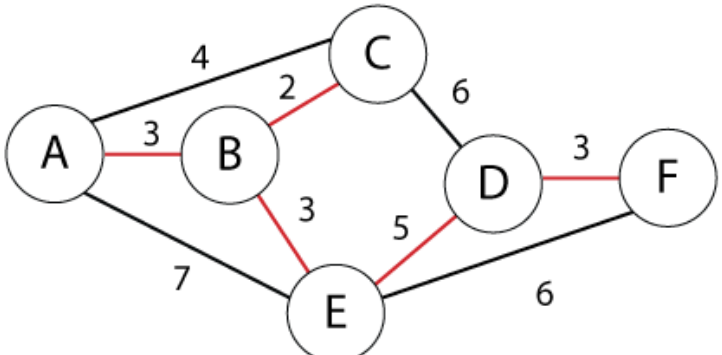
While S is nonempty and T is not yet spanning:

I. remove an edge with minimum weight from S
II. if that edge connects two different trees, then add it to the forest, combining two trees into a single tree, otherwise discard that edge.

Steps:

III. Connect B-C (2)
IV. Connect A-B (3)
V. Connect D-F (3)
VI. Connect B-E (3)
VII. Skip A-C (4) (Forms cycle)
VIII. Connect E-D (4)

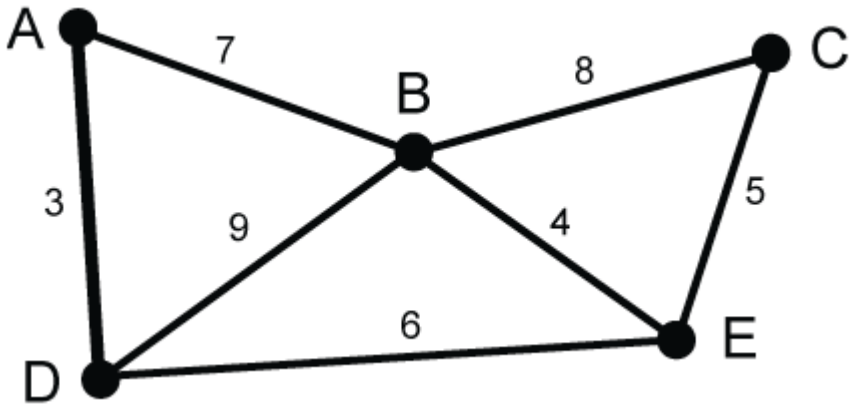
Stop. MST formed (5 edges, 6 vertices)



MST = {A-B , B-C , B-E, E-D, D-F}
Kruskal's algorithm is O(E log V) time. Where E is the set of edges and V is the set of vertices.

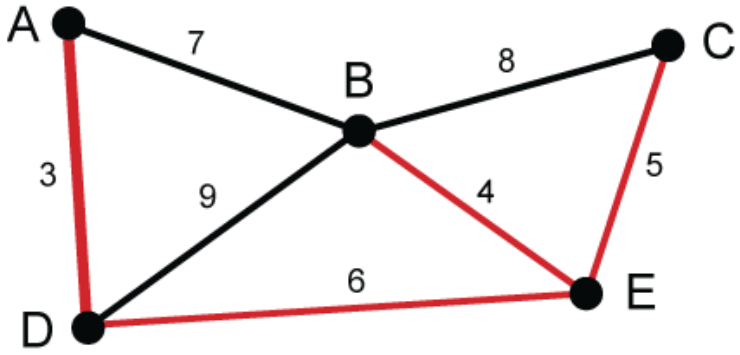
Q3 (10 Points)

Use Prim's algorithm to find a minimum spanning tree for the connected weighted graph below. *Show your work.*



What is the Time Complexity of Prim's algorithm?

Solution:



*Note: Using Kruskal's algorithm is NOT correct. It says: "*Use Prim's algorithm to find a minimum spanning tree*".*
Prim's algorithm (Queue edges for Minimum Spanning Tree) – is a greedy algorithm that finds a minimum spanning tree for a connected weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. Runs in $O(n^2)$ with an array; O(m log n) with a binary heap

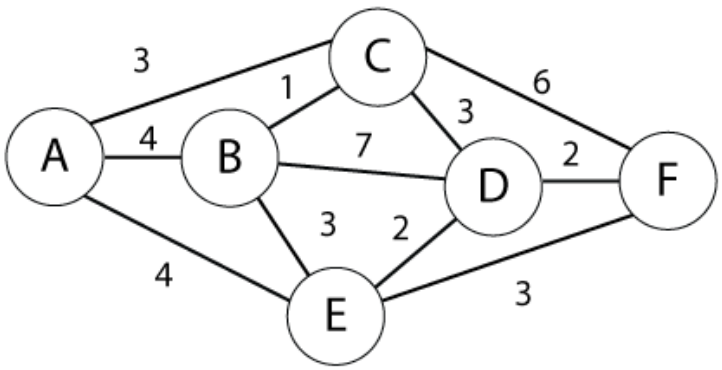
Step 1: Choose any starting vertex. Look at all edges connecting to the vertex and choose the one with the lowest weight and add this to the tree.
Step 2: Look at all edges connected to the tree. Choose the one with the lowest weight and add to the tree.
Step 3: Repeat step 2 until all vertices are in the tree (n-1 edges).
Prim(G, c) {
foreach (v in V) a[v] <- ∞
Initialize an empty priority queue Q
foreach (v in V) insert v onto Q
Initialize set of explored nodes S = {}

while (Q is not empty) {
u ⍰ delete min element from Q
S ⍰ S (union symbol){u}
foreach (edge e = (u, v) incident to u)
if ((v (is not an element symbol) (S) and (ce <a[v]))
decrease priority a[v] to ce
}

In other words
Take the minimum edge of the cut-set each time.
0: A S = {A}
1: A-D < A-B take A-D S = {A,D}
2: D-E < A-B or D-B take D-E S = {A,D,E}
3: B-E < A-B or D-B or C-E take B-E S = {A,D,E,B}
4: C-E < A-B or D-B or B-C- take C-E S = {A,D,E,B,C}
Done n-1 edges.
MST = { A-D , D-E , B-E, C-E }

Q4 (10 Points)

Find shortest path from A to F in the graph below using Dijkstra's algorithm. *Show your steps.*



Solution:

Given a graph, G, with edges E of the form (v1, v2) and vertices V, and a source vertex, s

dist : array of distances from the source to each vertex
prev : array of pointers to preceding vertices
i : loop index
F : list of finished vertices
U : list or heap unfinished vertices

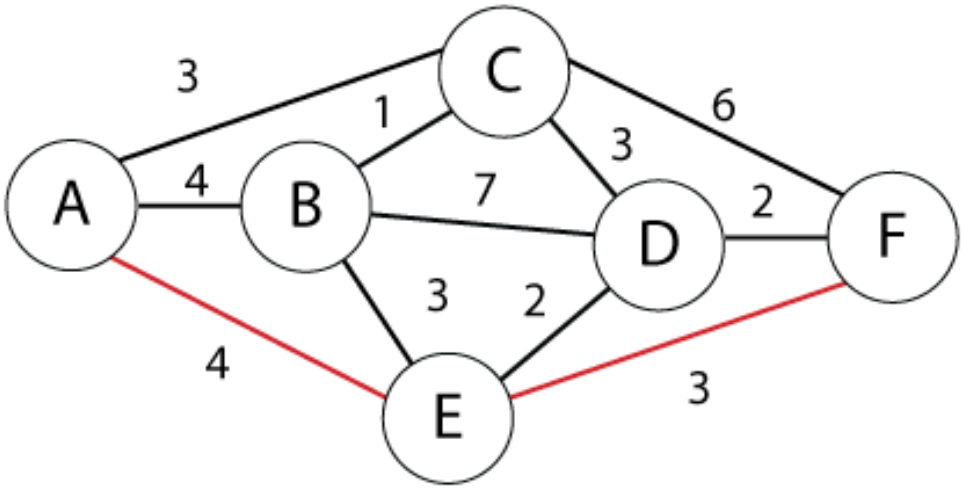
/* Initialization: set every distance to INFINITY until we discover a path */
for i = 0 to |V| - 1
dist[i] = INFINITY
prev[i] = NULL
end

while(F is missing a vertex)
pick the vertex, v, in U with the shortest path to s
add v to F
for each edge of v, (v1, v2)
/* The next step is sometimes given the confusing name "relaxation"
if(dist[v1] + length(v1, v2) < dist[v2])
dist[v2] = dist[v1] + length(v1, v2)
prev[v2] = v1
possibly update U, depending on implementation
end if
end for
end while

Source A

		A	B	C	D	E	F
1: A {A}	A	0	(4,A)	(3,A)**	INF	(4, A)	INF
2: C {A,C}	C	0	(4,A)	(3,A)	(6,C)	(4, A)**	(9,C)
3: E {A,C,E}	E	0	(4,A) **	(3,A)	(6,C)	(4, A)	(7,E)
4: B {A,C,E,B}	B	0	(4,A)	(3,A)	(6,C) **	(4, A)	(7,E)
5: C {A,C,E,B,D}	D	0	(4,A)	(3,A)	(6,C)	(4, A)	(7,E) **
4: F {A,C,E,B,D, F}	F	0	(4,A)	(3,A)	(6,C)	(4, A)	(7,E)

** U with the shortest path to s
Note: Can take in A, C, E, B, D, F or A, C, B, E, D, F as E and B both have shortest path cost 4.
Red indicates a vertex has been moved from U to F

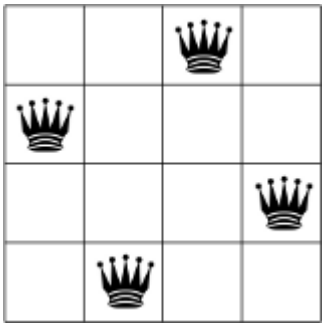


Now we return our final shortest path, which is: A à E à F Cost (4+3 = 7)

Q5 (15 Points)

The N Queen Problem. In chess, a queen can attack horizontally, vertically, and diagonally. The N-queens problem asks:

How can N queens be placed on an NxN chessboard so that no two of them attack each other?



No two queens are on the same row, column, or diagonal.

Solution:

All state-space search implementations must represent the current state of the problem to allow the relevant evaluations, operator effects, etc. to be calculated efficiently.
Here, maybe the most obvious candidate is an N x N binary array with 0's for empty squares and 1's for squares with queens. This might be good for (sighted) humans, but a little thought should reveal it is too big (redundant), and not good for discovering conflicts (attacks) quickly via computer. My choice would be a Scheme vector (I like that better than a list -- why?). It is N long for the N-Queens problem and its i^{th} element represents the i^{th} column of the board. The value of that element is the row on which sits a queen, or -1 if the column is empty. This representation only allows 0 or 1 queens per column, which already enforces 1/4 of the non-attack constraints -- neat, eh? So a 5x5 board with three queens placed (illegally) diagonally in the first three columns would have the state vector [0 1 2 -1 -1], where -1 means 'nothing placed here'.

More Here : https://www.cs.rochester.edu/u/brown/173/readings/NQueens.html

Q6 (15 Points)

Give a brief definition for **all of** the following:
1. Game Tree (include example)
2. Minimax algorithm (include example)
3. Alpha–beta pruning (include example)

Solution:

1. A game tree is a directed graph whose nodes are positions in a game and whose edges are moves. The

complete game tree for a game is the game tree starting at the initial position and containing all possible moves from each position; the complete tree is the same tree as that obtained from the extensive-form game representation.

2. Minimax algorithm
Minimax (sometimes MinMax, MM or saddle point) is a decision rule used in artificial intelligence, decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario. When dealing with gains, it is referred to as "maximin"—to maximize the
minimum gain. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision-making in the presence of uncertainty.

3. Alpha–beta pruning

Alpha–beta pruning is a search algorithm that seeks to decrease the number of nodes that are evaluated by the minimax algorithm in its search tree. It is an adversarial search algorithm used commonly for machine playing of two-player games (Tic-tac-toe, Chess, Go, etc.). It stops evaluating a move when at least one possibility has been found that proves the move to be worse than a previously examined move. Such moves need not be evaluated further. When applied to a standard minimax tree, it returns

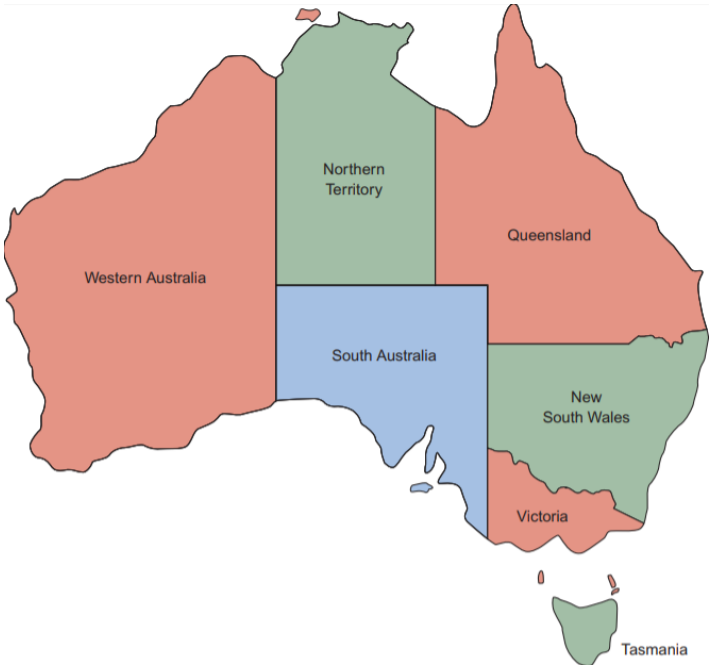
the same move as minimax would, but prunes away branches that cannot possibly influence the final Decision

Q7 (15 Points)

Map Coloring Problem. Given a map of Australia, color it using three colors such that no neighboring territories have the same color.

Express this problem as Constraint satisfaction problem**.**

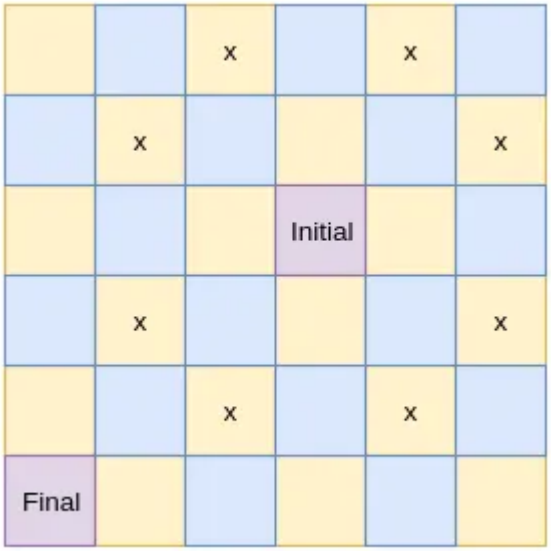
Solution:



A CSP is composed of: q A set of variables X1,X2,…,Xn with domains (possible values) D1,D2,…,Dn q A set of constraints C1,C2, …,Cm q Each constraint Ci limits the values that a subset of variables can take, e.g., V1 ≠ V2 In our example: n Variables: WA, NT, Q, NSW, V, SA, T n Domains: Di ={red,green,blue} n Constraints: adjacent regions must have different colors. q E.g. WA ≠ NT (if the language allows this) or q (WA,NT) in {(red,green),(red,blue),(green,red),(green,blue),(blue,red), (blue,green)}

Q8 (15 Points)

Knight Tour Problem.
Given a chess board of size n x n, initial position of knight and final position of knight. We need to find the minimum number of steps required to reach final position, If it is impossible to reach final position then return -1. Knight moves according to following rules and is not allowed to leave chess board.
Consider the diagram below



In the above diagram initially the knight is at (2,3) and has to reach (5,0). The knight can reach to the final cell in two steps using the following paths (2,3) -> (3,1) -> (5,0).
Give an algorithm to solve Knight Tour Problem.

Solution:

We need to Perform a BFS for this. n this Knight Tour problem, knight is allowed to move in 8 directions and each move has unit cost associated with it. We can visualize this situation as a graph where the edges will represent possible moves and the vertices will represent possible positions of knight. This reduces the problem to standard problem of shortest path in unweighted graph. Hence we will be using Breadth First Search (BFS) to solve this problem.