Homework 1
CS4100/5100, Fall 2014

Your Name: **SHAKTI PRASAD PATRO**

1. In what order do the following search methods (assume you use the graph search version) expand the nodes in the graph below (assume that nodes are added to the stack/queue in alphabetical order)? The agent starts at node a and must reach node g.

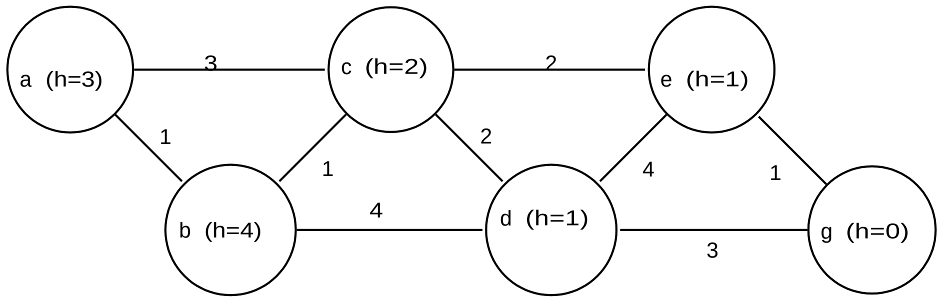
a. BFS (5 pts)? **a -> b ->c-> d-> e-> g (soln: a,b,d,g)**

b. UCS (5 pts)? **a -> b-> c-> d-> e-> g(soln: a, b, c, e, g)**

c. DFS (5 pts)? **a->b-> c-> d -> e-> g (soln: a, b, c, d, e, g)**

d. Greedy search (5 pts)? **a -> c-> d -> g (soln : a, c, d, g)**

e. A* (5 pts)? **a -> b-> c-> d-> e->g (soln: a, b, d, g)**



2. There are two players playing a game (Max and Min). Below is the minimax and $\alpha - \beta$ search tree for this game. The leaves are labeled with their static evaluation values.

a) (5 pts) Use Minimax to evaluate the game tree below. Fill in the blanks in the figure below with each nodes Minimax value.

1) **Simple MinMax Problem**

From Bottom to up:

Max: **12** **15** **9** 2 11 10 **12** 9 15 **14**

Min: **9** **2** **10** **9**

Max: **10**

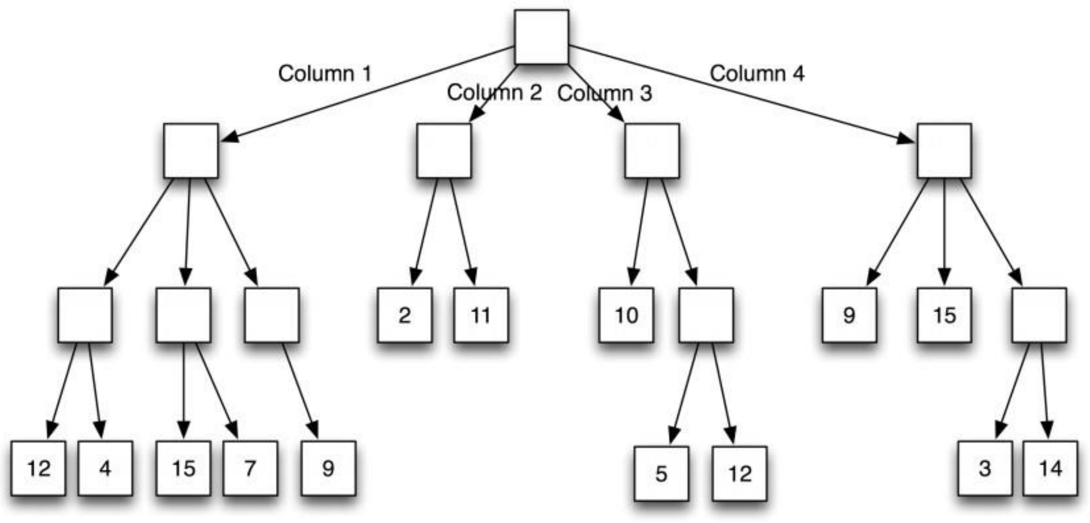
2) **Using $\alpha - \beta$ pruning**

From Bottom to up:

Max: **12** **>=15** **9** 2 11 10 **12** 9 15 **14**

Min: **9** **2** **10** **9**

Max: **10**



b) (5 pts) What moves does Minimax say the maximizer should make? Is it column 1, column 2, column 3, or column 4?

Ans: column 3

3. (Russel and Norvig, Exercise 3.3) Suppose two friends live in different cities on a map, such as the Romania map shown in Figure 3.2. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance d(i, j) between

the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

a. (5 pts) Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)

ANS:

1) **STATE:** a state contains the coordinates of the two friends f1(i,j) and f2(i,j) and the path cost.
2) **ACTIONS:** the actions can be any neighboring city the friends move to.
3) **PATH COST:** the road distance travelled by each friend (more specifically the max of their distances travelled) ie: if c1(i,j) is cost for friend1 and
4) **TRANSITION FUNCTION:** the functions tells the friends to move to the next city.
5) **GOAL STATE:** the state where both friends have same coordinates. That is, (f1 == f2)

b. (5 pts) Let D(i, j) be the straight-line distance between cities i and j. Which of the following heuristic functions are admissible?

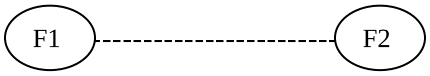
(i) D(i, j):
If f1 and f2 are on the same straight line then the path cost between them is equal to the straight line distance. Otherwise it is always greater according to triangle inequality. SO this heuristic is **admissible.**

(ii) 2D(i, j):
This heuristic is **not admissible** as twice the straight line distance can be more than the actual path cost.
Ex: **f1**--------c1------------**G**------------c2-------------**f2**
In this case the straight line distance = c1+c2
And actual path cost = c1 + c2
So 2D = 2(c1+c2) >> Actual Path cost(c1+c2)

(iii) D(i, j)/2:
Since D(i,j) < actual path cost so is D(i,j)/2. Hence this is **admissible.**

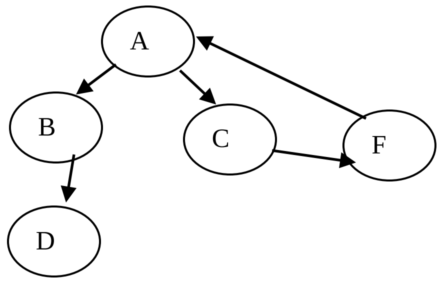
c. (5 pts) Are there completely connected maps for which no solution exists?

If friend1 f1 and friend2 f2 are on two adjacent cities and there are no paths other than crossing



In this case when f1 reaches f2 node, f2 will reach f1. There is no solution for this graph.

d. (5 pts) Are there maps in which all solutions require one friend to visit the same city twice?



In the above map, A, B are the two friends. The friend at A has to complete the loop and travel back to A to meet the other friend. There is no other solution in this graph other than travelling one node twice.

4. (Russell and Norvig, Exercise 5.7) Prove the following assertion: For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will be never be lower than the utility obtained playing against an optimal MIN (10 pts). Can you come up with a game tree in which MAX can do still better using a suboptimal strategy against a suboptimal MIN (5 pts)?

ANS:

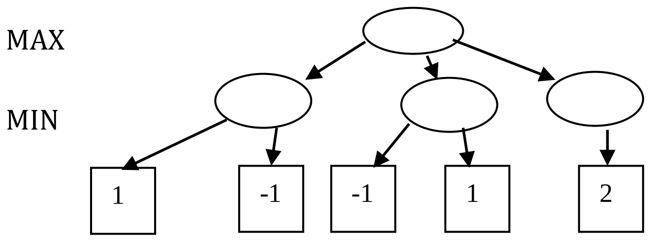
Suppose at an ith layer is a MAX position. Then (i+1)th layer is a MIN position.
Now optimal min will take value = min{m1,m2,m3…….mn} = optMin
Where m1,m2…mn are the values of terminal nodes below MIN.

If min takes suboptimal values then subOptMin is always greater than optMin
Ie. (subOptMin > optMin)

Now when MAX selects , value = max {minValues}
= max(subOptValues)
>= max(minOpt Values)

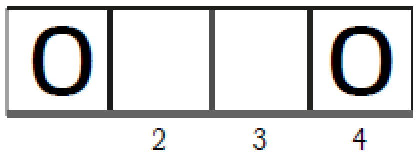
So MAX always takes greater than or equal to the SubOptimal MIN value.

Therоretically speaking, if MIN takes a suboptimal choice, then its node value can be greater than its optimal value. So when it comes to MAX node, it value can only increase not decrease. Now again when MIN takes a suboptimal decision, this may be equal or higher than its current value and so on. This proves that for a suboptimal MIN, utility by MAX will always be greater.

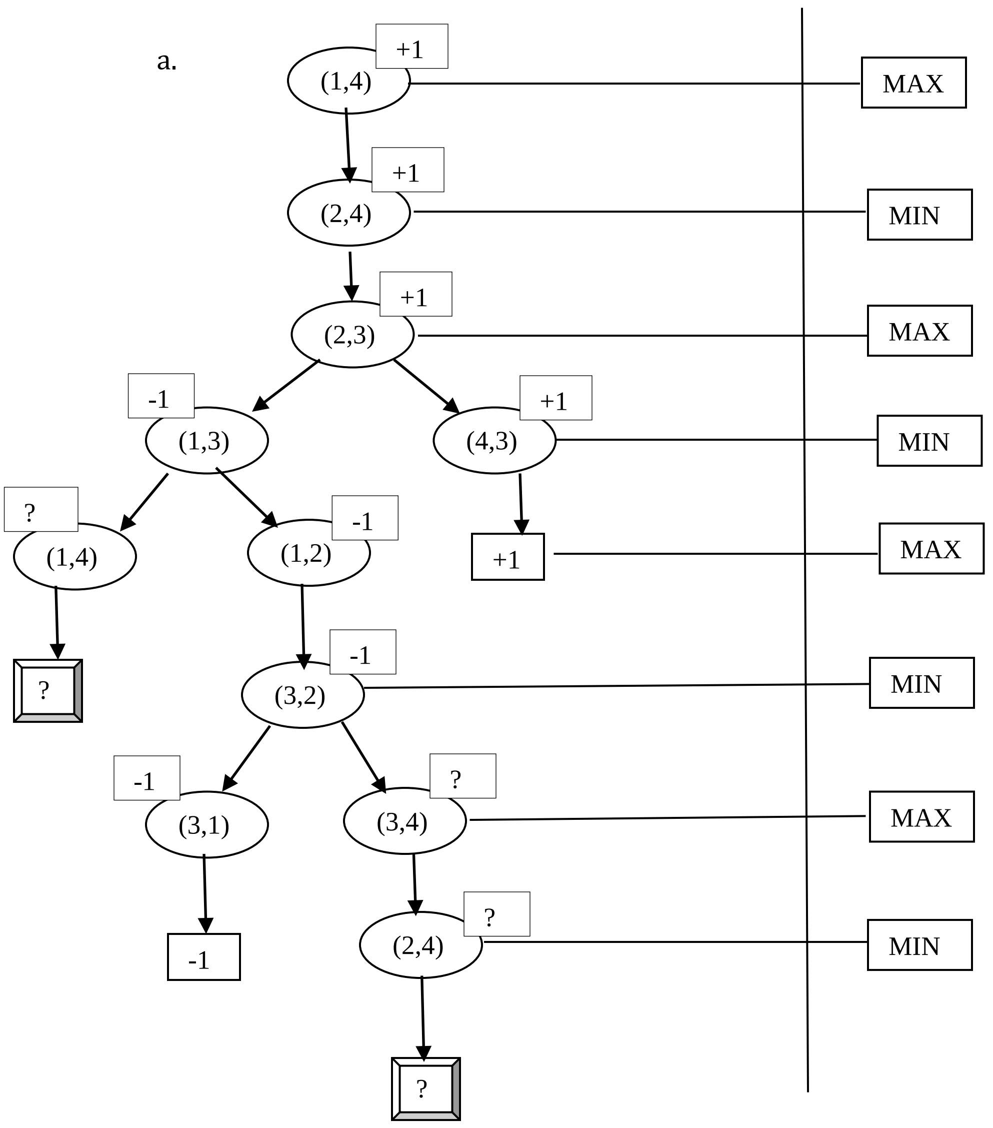


In this case if min takes suboptimal values, then it will have – (1, 1, 2). So even if MAX takes any suboptimal choice it wins as MIN will always take the suboptimal choice after MAX.

5. Do Exercise 5.8 in Russel and Norvig (5 pts for each part).



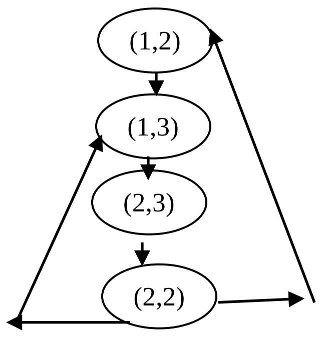
a.



b. Handling '?' -> For min : min(?, -1) = -1, for max: max(?, +1)= +1

c. Standard algorithm would fail as it would go into a loop. We can fix this by adding a '?' for the looping scenario. Then we can verify if state is '?', if yes then return the next state on the basis of the other state it is compared to. Like min/max (state1, '?') = state1. Now it won't work where we compare two unknown quantities.

Ex: See below graph: here the state of 2,2 is undefined by the algorithm



d. Assuming Both A,B take optimal choices.

For n=3, A always loses
For n=4, A always wins

Now for n=5, for first move each player has one choice.
So after first move there is another game of 3. So we can deduce that if B wins for game three, then it will reach the 2nd node first than A reaches the 4th node. Eventually B will win.

Now for n=6, after taking 1st compulsory move, we are left with a 4 square problem, which A always wins. ie. A reaches 5th node earlier than B reaches 2nd node. This confirms that A wins.

This can be extended to n boxes to prove that A wins every even game for n>2 and B wins every odd game for n>2.