ECE 479/579 **Name:**
Fall 2015 **(please print)**

Sample Examination
October 5, 2015

1. (10 points) Design an AI production system to convert positive integer decimal numbers into octal ones. Illustrate its operation by converting 189.

Answer:
1. State-set: { (x,octal_string) | x is decimal. Octal string is octal # }
2. Initial State : {189,{}} {}-Empty String
3. Goal state: {0, octal_string}

4. Operators:
   a) div – yields the integral part of the result of O1 divided by O2, i.e., O1 div O2
   b) mod – yields the integer remainder of the division, i.e, O1 mod O2

5. Rules
   R1: if x>0
   then
   x ← x div 8
   octal_string ←attach(x mod 8) to octal_string from the left

   R2: if x=0, STOP

   R3: if x<0, then output "invalid input", STOP

Example: (189, {}) ← R1
(23, {5}) ← R1
(2, {75}) ← R1
(0, {275}) ← R2, STOP!

2. (10 points). Consider an instance of the Towers of Hanoi game. The objective of the game is to move both disks from the left pin to the right one (please see figure), subject to the following constraints:

- Only one disk can be moved at a time,
- The large disk cannot be placed on top of the small disk.

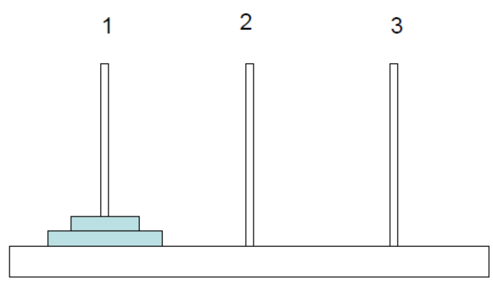
The state of the game can be indicated by a pair (x, y), where x is the pin number for the large disk and y is the pin number for the small disk. For example, (2, 1) is a configuration in which the large disk is on pin 2 and the small one on pin 1.

You are to:

a) Using this notation, list all possible states for our instance for the game (see figure).

b) Create a table indicating which transitions between states are possible in one move.

c) Find a solution using breadth-first search. Ensure that no path visits the same node twice.



Answer:

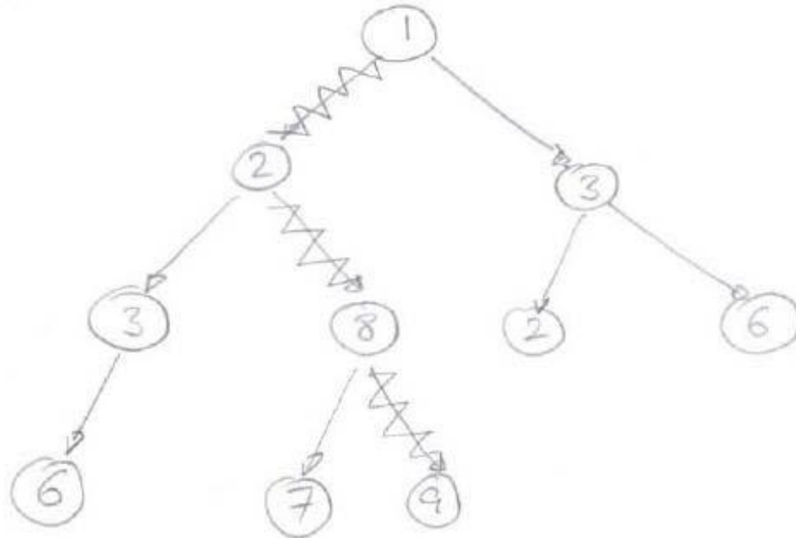
a) 1. (1,1) 4. (2 ,1) 7. (3,1)
2. (1 ,2) 5. (2,2) 8. (3,2)
3. (1,3) 6 . (2,3) 9. (3,3)

b)

b) State transition table; x denotes a legal transition, for example, from 1 (1,1) to 3. (1,3) (big disc is on peg, small on peg 3)

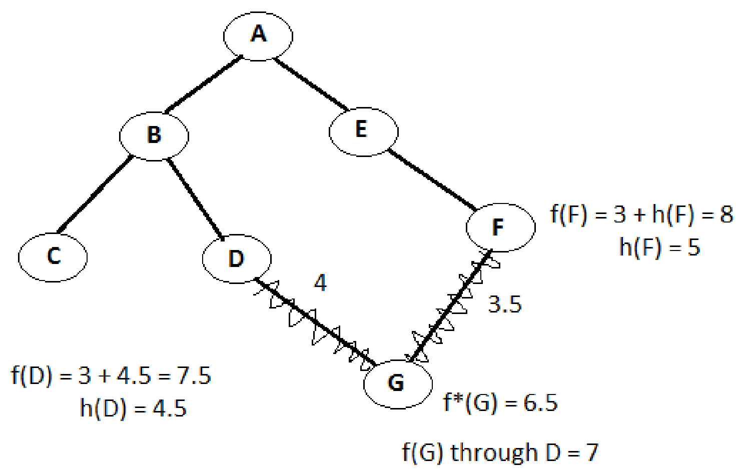
	1	2	3	4	5	6	7	8	9
1		x	x						
2	x		x					x	
3	x	x				x			
4					x	x	x		
5				x		x			
6			x	x	x				
7				x				x	x
8		x					x		x
9							x	x	

c)



3. (8 points) Consider the A* algorithm. Devise and explain an example illustrating that using h(n) function which is NOT an underestimate of the actual *cost* may not guarantee optimality of the solution.

Answer:



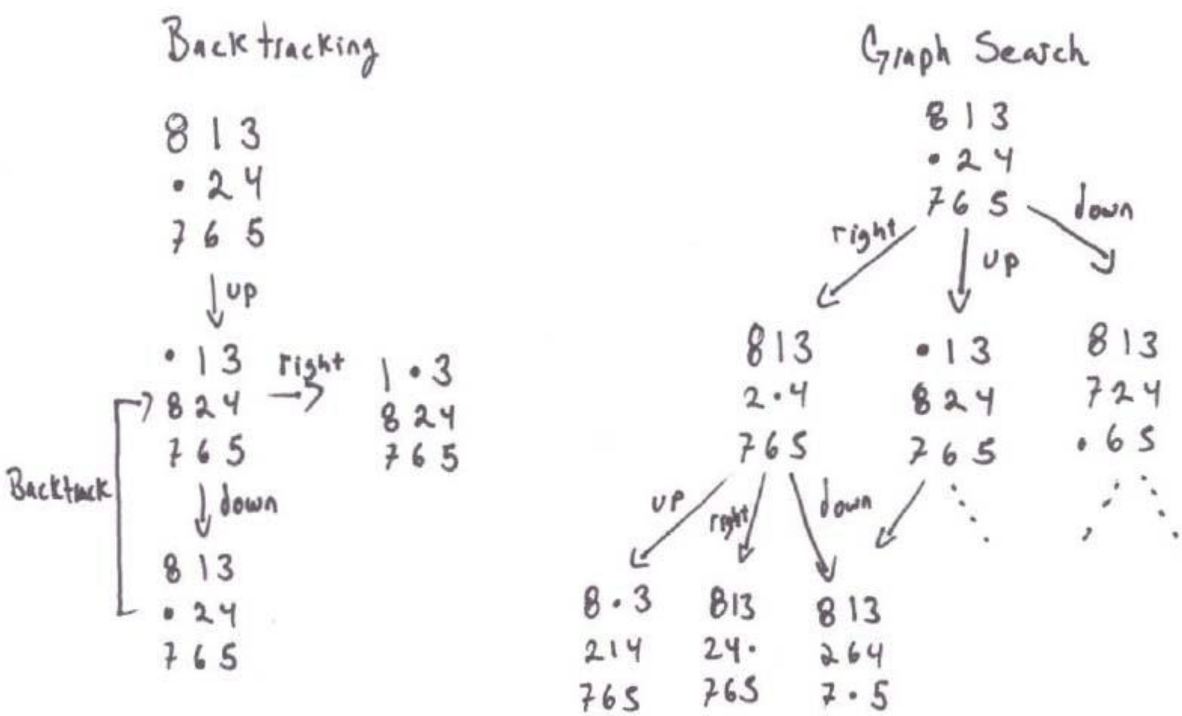
Assume nodes D and F are open, we would select D as f (D) = 7.5. However the optimal path as we can see above is A-E-F-G. h(F) was an over-estimate.

4. (7 points) Explain the difference between the backtracking control strategy and the general graph search. Use the 8-puzzle example discussed in class. Use

Initial State	Goal State
8 1 3	1 2 3
- 2 4	- 8 4
7 6 5	7 6 5

You need not solve the problem! Rather than generating the entire state space, carry out a few steps that clearly show how the strategies differ.

Answer:
In Backtracking, the control strategy forgets any trial paths that result in failures. Only the path being currently extended is stored explicitly. In graph search, provision is made for keeping track of the effects of all applicable rules simultaneously.



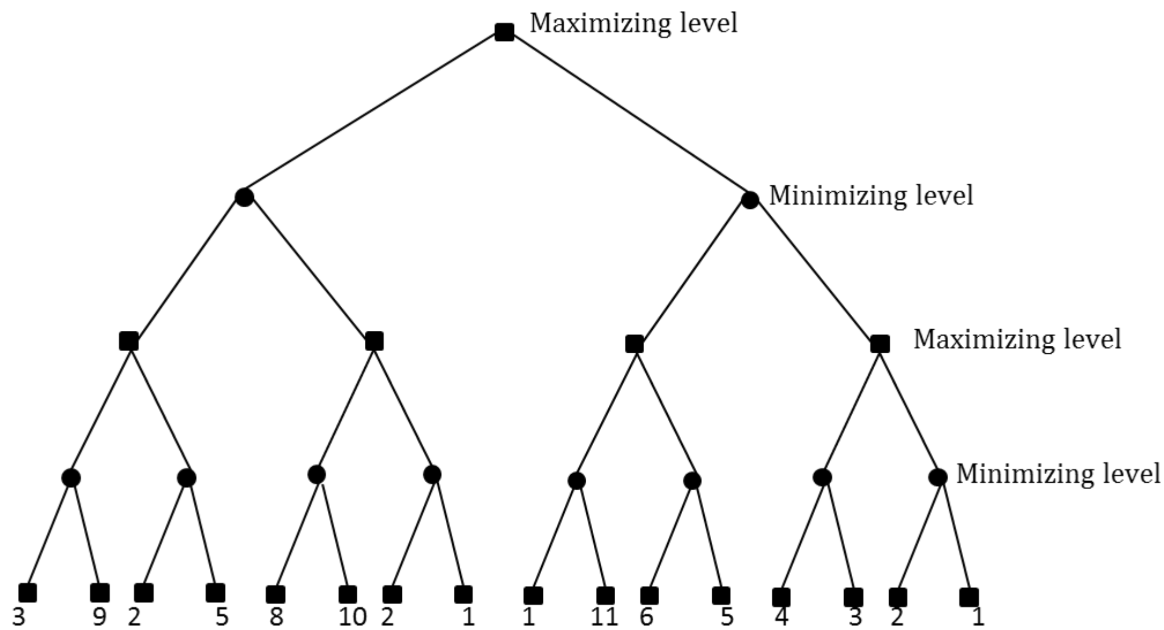
5. (5 points) Would the A* algorithm work properly on graphs containing cycles? Please justify your answer.

Answer: When a new path is generated to an existing node, this path is discarded. If costs are nonnegative, a cyclic path can never be better than the same path with the cycle omitted. Further details can be provided by an illustration.

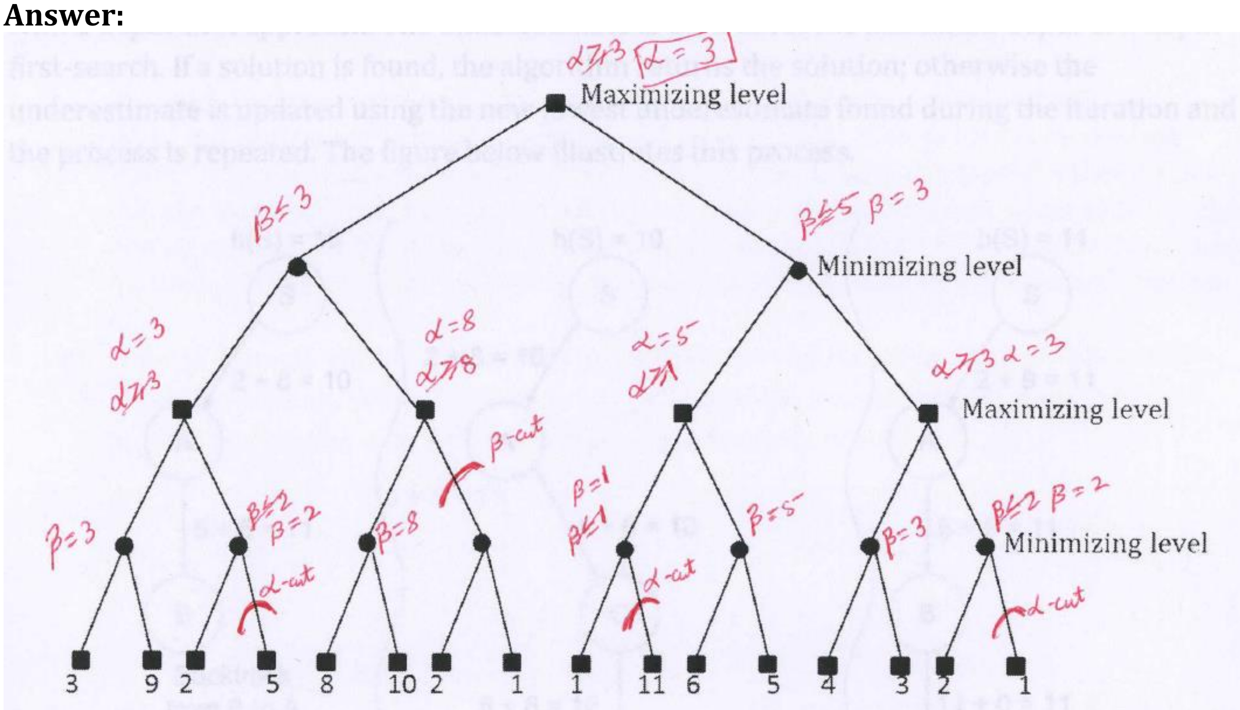
6. (10 points) Given the game tree below:

a) Apply alpha-beta pruning, searching the tree in the left-to-right direction

b) Apply alpha-beta pruning, searching the tree in the left-to-right direction



Answer:



7. (Graduate Students Only – 10 points) Critically discuss the following statement: given that we now have massively parallel computing clusters, there is no point in using heuristics to speed up search when solving exponentially complex problems. Please provide <u>three</u> reasons/aspects when discussing the statement.

Heuristics are useful even in the face of massive parallelism. Consider this:

1) Heuristics save memory by expanding fewer nodes. Thus larger problems could be tackled with the use of heuristics.
2) Parallelism had overhead (networking and synchronization). Bandwidth may be limited.
3) Each core only provides a (sub)linear speed up, but these problems are exponentially hard.
4) Heuristics still help with the speed even when parallelized.
5) Consider deploying an AI, then you must consider: weight, size, cost, power etc.
6) Heuristics can provide optimality guarantees, whereas massive blind search only provides guarantee if it is exhaustive.
7) Some search trees may not parallelize efficiently (thin, deep trees)
8) Portability and bugs. There are more things that can go wrong with parallelism (race conditions, deadlock, etc.) than with using a heuristic.

Counter points:

1) Given small enough problems where each potential solution can be easily enumerated, all solutions could be tested simultaneously.
2) Heuristics may be more difficult to determine than to parallelize an uninformed search.