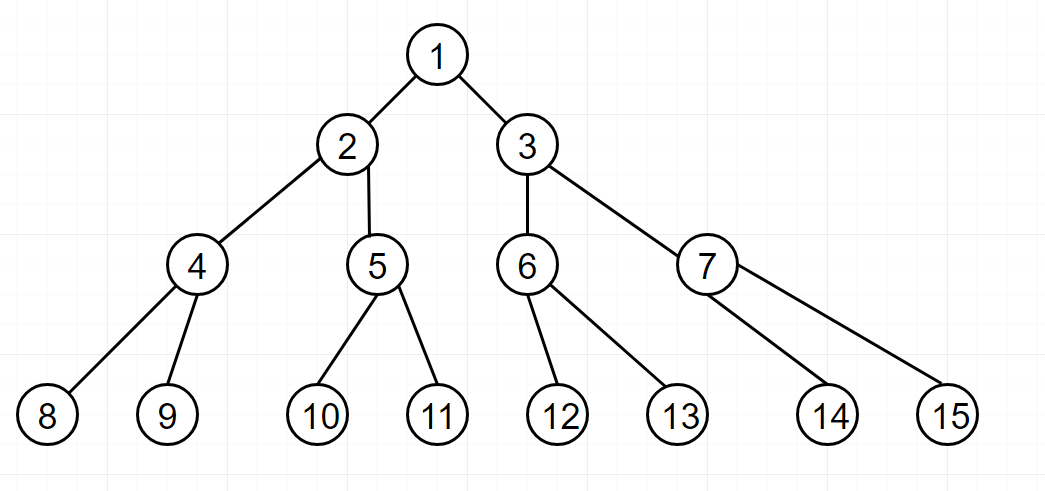
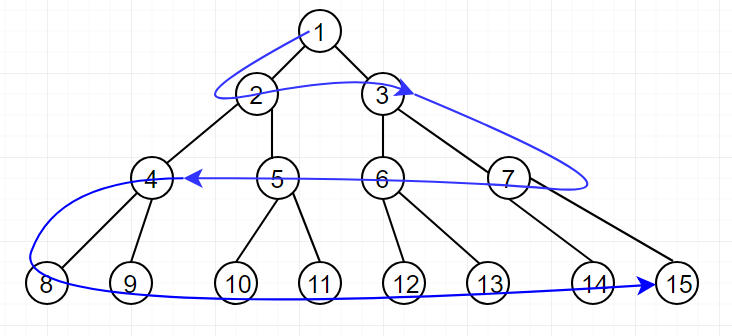
**Chapters 4 & 5 Problem Set**

Your *performance* on this assignment will *not* count towards your homework grade. However, completion of this assignment by the due date *is* required as part of your participation grade. It is primarily for study and preparation for the Midterm (Oct 8).

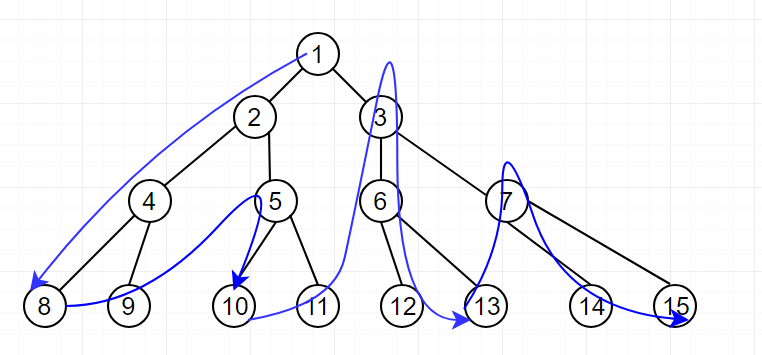
**Q1. Consider a state space (i.e. search tree) where the start state is number 1 and each state k has two successor states: numbers 2k and 2k+1. Draw the portion of the state space for states 1 to 15.**

**Q2. Suppose that the goal state of the decision tree is Question #1 is 15. List the order in which nodes will be visited for**

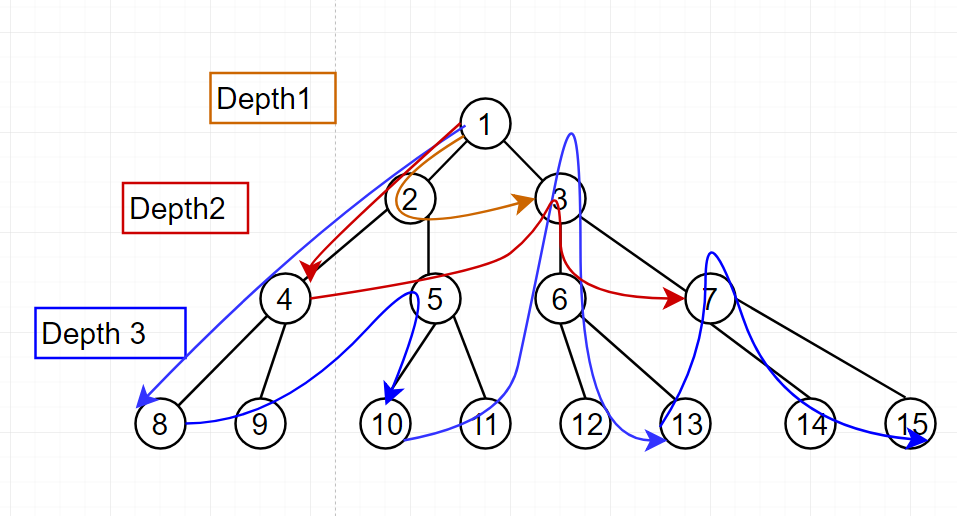
1. Breadth-first Search (BFS)
   1. [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]



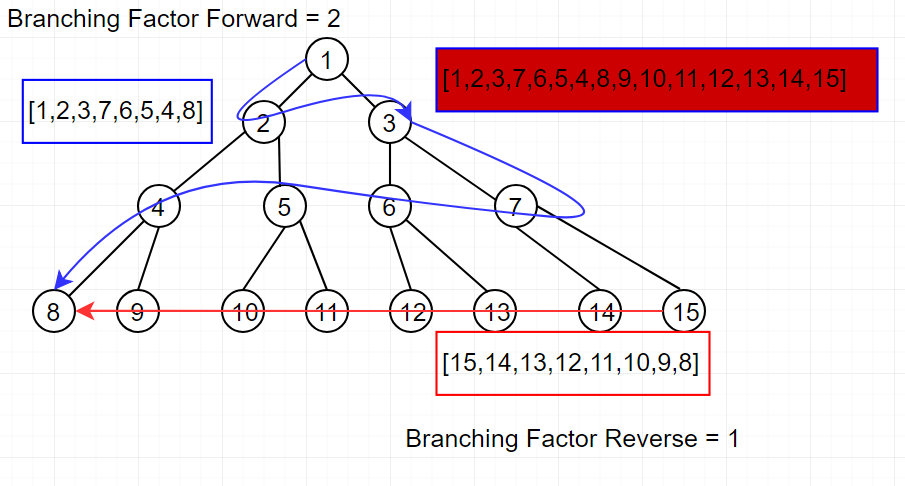
1. Depth-limited Search
   1. [1, 2, 4, 8, 9, 5, 10, 11, 3, 6, 12, 13, 7, 14, 15]



1. Iterative Deepening Search.
   1. [1 / 1, 2, 3 / 1, 2, 4, 5, 3, 6, 7 / 1, 2, 4, 8, 9, 5, 10, 11, 3, 6, 12, 13, 7, 14, 15]



**Q3. How well would Bidirectional Search work on the problem in Question #2? What is the branching factor in each direction of the bidirectional search?**



Good, because the branching factor is linear in the reverse direction

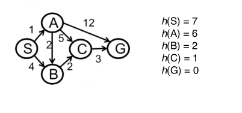
Bforward = 2

Breverse = 1

**Q4. Does the answer to the search from Question #3 suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a goal state with no (or almost no) search?**

Yes. A recursive algorithm could generate a path from 1 to n for any n in the tree.

**Q5,Q6,Q7 Consider the following search graph with start node S and goal node G and its associated heuristic values.**



**Greedy Best First**

{source node, dest node} = HueristicCost{dest}﻿﻿

(n) = h(n)

Step 0

{S} f = 7 // expand this node

Step 1

{S} f = 6

{S,A} f = 6

{S,B} f = 2 // expand this node

Step 2

{S,A} f = 6

{S,B} f = 2

{S,B,C} f = 1 // expand this node

Step 3

{S,A} f = 6

{S,B,C} f = 1

{S,B,C,G} f = 0 // Goal reached

**A\***

{source node, dest node} = PathCost{dest} + HueristicCost{dest}﻿﻿

f(n) = g(n) + h(n)

Step 0

{S} f = (0) + 7 = 7 // expand this node

Step 1

{S} f = (0) + 6 = 6

{S,A} f = (1) + 6 = 7

{S,B} f = (4) + 2 = 6 // expand this node

Step 2

{S,A} f = (1) + 6 = 7 // expand this node

{S,B} f = (4) + 2 = 6

{S,B,C} f = (4 + 2) + 1 = 7

Step 3

{S,A} f = (1) + 6 = 7

{S,A,B} f = (1+ 2) + 2 = 5 // expand this node

{S,A,C} f = (1 + 5) + 1 = 7

{S,B,C} f = (4 + 2) + 1 = 7

Step 4

{S,A,B} f = (1+ 2) + 2 = 5

{S,A,B,C} f = (1+ 2 + 2) + 1 = 6 // expand this node

{S,A,C} f = (1 + 5) + 1 = 7

{S,B,C} f = (4 + 2) + 1 = 7

Step 5 (Goal reached, but better candidates remain)

{S,A,B,C} f = (1+ 2 + 2) + 1 = 6

{S,A,B,C,G} f = (1+ 2 + 2 +3) + 0 = 8

{S,A,C} f = (1 + 5) + 1 = 7 // expand this node

{S,B,C} f = (4 + 2) + 1 = 7

Step 6

{S,A,B,C,G} f = (1+ 2 + 2 +3) + 0 = 8

{S,A,C,G} f = (1 + 5 + 3) + 0 = 9

{S,B,C,G} f = (4 + 2) + 1 = 7 // expand this node

Step 7

{S,A,B,C,G} f = (1+ 2 + 2 +3) + 0 = 8 // GOAL and lowest candidate

{S,A,C,G} f = (1 + 5 + 3) + 0 = 9

{S,B,C} f = (4 + 2 + 3) + 0 = 9

**RBFS**

{source node, dest node} = PathCost{dest} + HueristicCost{dest}﻿﻿

Correct

Step 0

{S} f = (0) + 7 = 7 // expand this node

Step 1

{S} f = (0) + 6 = 6

{S,A} f = (1) + 6 = 7

{S,B} f = (4) + 2 = 6 // expand this node

Step 2

{S,A} f = (1) + 6 = 7 // expand this node

{S,B} f = (4) + 2 = 6

{S,B,C} f = (4 + 2) + 1 = 7 // contract this node

Step 3

{S,A,B} f = (1+ 2) + 2 = 5 // expand this node

{S,A,C} f = (1 + 5) + 1 = 7

{S,B} f = 7\* // \* contracted node w/ best child f-value

Step 4

{S,A,B,C} f = (1+ 2 + 2) + 1 = 6 // expand this node

{S,A,C} f = (1 + 5) + 1 = 7

{S,B} f = 7\*

Step 5

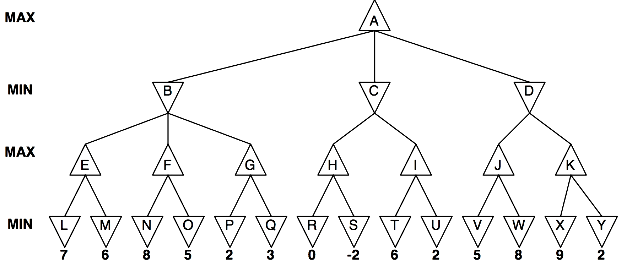
{S,A,B,C,G} f = (1+ 2 + 2 +3) + 0 = 8 // GOAL and lowest candidate

{S,A,C} f = (1 + 5) + 1 = 7

{S,B} f = 7

**Q8. Consider the following search tree using the Minimax algorithm. Values for leaf nodes L through Y are given.**

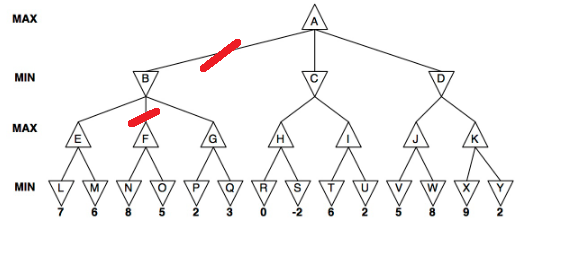
**Compute the values for nodes A through K and write them in the Answer space below formatted one node per line, e.g.**



A = 8  
B = 3  
C = 0  
D = 8  
E = 7  
F = 8  
G = 3  
H = 0  
I  = 6  
J = 8  
K = 9

**Q9. Using the same game tree as the Minimax question above, determine which states will *not be explored* if alpha-beta pruning is used.**

**On an attached file (PDF, JPG, or GIF)**, draw the tree and circle all unvisited (i.e. pruned) subtrees, add (min,max) pairs at each appropriate node (nodes in pruned subtrees do not have to be labeled), and indicate whether **alpha-pruning** or



A: Node **I** will be pruned because MAX (i.e. alpha) will decide that **C** will choose no higher node than **H** (i.e. 0) and MAX already has a better choice at **B**=3.