CSE 574 Homework 1

Zackary Crosley

September 8, 2018

Problem 1. Two dice are rolled

Two dice are rolled simultaneously.

- Event A: The sum of two dice is 3.
- Event B: The sum of two dice is 7.
- Event C: At least one of the dice is 2.

1. Compute P(A|C)

$$P(A|C) = \frac{P(C|A) \times P(A)}{P(C} \tag{1}$$

$$P(A|C) = \frac{1 \times P(A)}{P(C)} \tag{2}$$

$$P(A|C) = \frac{\frac{2}{36}}{1 - \frac{5}{6} \times \frac{5}{6}} \tag{3}$$

$$\mathbf{P(A|C) = \frac{2}{11}}$$

2. Compute P(B|C)

$$P(B|C) = \frac{P(C|B) \times P(B)}{P(C)} \tag{4}$$

$$P(B|C) = \frac{\frac{2}{6} \times P(B)}{P(C)} \tag{5}$$

$$P(B|C) = \frac{\frac{2}{6} \times \frac{6}{36}}{1 - \left(\frac{5}{6} \times \frac{5}{6}\right)} = \frac{\frac{2}{6} \times \frac{1}{6}}{1 - \frac{25}{36}} \tag{6}$$

$$P(B|C) = \frac{\frac{2}{36}}{\frac{11}{36}} \tag{7}$$

$$\mathbf{P(B|C) = \frac{2}{11}}$$

Problem 2. To win a chess tournament you must win two of three rounds in a row. Show that the optimal strategy is to play the weakest opponent second and the order of the other two doesn't matter.

W_i indicate winning a game against player i and L_i indicate losing againt player i. Therefore in order to win the tournament one of the following conditions must be met:

$$\{W_1, W_2, W_3\}, \{W_1, W_2, L_3\}, \{L_1, W_2, W_3\} \tag{8}$$

$$P(\{W_1, W_2, W_3\}) = p_1 \times p_2 \times p_3 \tag{9}$$

$$P(\{W_1, W_2, L_3\}) = p_1 \times p_2 \times (1 - p_3) \tag{10}$$

$$P(\{L_1, W_2, W_3\}) = (1 - p_1) \times p_2 \times p_3 \tag{11}$$

$$P(\{W_1, W_2, W_3\} \vee \{L_1, W_2, W_3\} \vee \{W_1, W_2, L_3\}) = (p_1 \times p_2 \times p_3) + ((1-p_1) \times p_2 \times p_3) + (p_1 \times p_2 \times (1-p_3)) \tag{12}$$

$$P(\{W_1, W_2, W_3\} \vee \{L_1, W_2, W_3\} \vee \{W_1, W_2, L_3\}) = p_2 \times (p_1 \times p_3 + (1-p_1) \times p_3) + p_1 \times (1-p_3)) \tag{13}$$

$$P(\{W_1, W_2, W_3\} \vee \{L_1, W_2, W_3\} \vee \{W_1, W_2, L_3\}) = p_2 \times (p_1 + p_3 - p_1 \times p_3) \tag{14}$$

By Communative Property of addition and multiplication, we see that swapping the values of p_1 and p_3 would have no change to the equation outcome. Therefore the order of play against players 1 and 3 don't matter.

Playing the game with the highest probability of success as the second game is only optimal if:

$$p_2 \times (p_1 + p_3 - p_1 \times p_3) \geq p_1 \times (p_2 + p_3 + p_2 \times p_3) \tag{15}$$

$$p_2 \times p_1 + p_2 \times p_3 - p_1 \times p_2 \times p_3 \geq p_1 \times p_2 + p_1 \times p_3 - p_1 \times p_2 \times p_3 \tag{16}$$

$$p_2 \times p_3 \geq p_1 \times p_3 \tag{17}$$

Since $p_2 \geq p_1$ by definition, $p_2 \times p_3 \geq p_1 \times p_3$ must be true. Therefore, p_2 is more optimal in the second position than p_1. This property holds true for p_3 as well since the placement of p_1 and p_3 were shown to be irrelevant relative to one another.

Therefore, p_2 should be the match with the highest probability of winning.

Problem 3. The output of a factory is produced by three machines that produce 20

$$P(M_3|Defect) = \frac{P(Defect|M_3) \times P(M_3)}{P(Defect)} \tag{18}$$

$$P(M_3|Defect) = \frac{P(Defect|M_3) \times P(M_3)}{P(M_3, Defect) + P(M_2, Defect) + P(M_1, Defect)} \tag{19}$$

$$P(M_3|Defect) = \frac{P(Defect|M_3) \times P(M_3)}{P(Defect|M_3) \times P(M_3) + P(Defect|M_2) \times P(M_2) + P(Defect|M_1) \times P(M_1)} \tag{20}$$

$$P(M_3|Defect) = \frac{0.01 \times 0.50}{0.01 \times 0.50 + 0.03 \times 0.30 + 0.05 \times 0.20} \tag{21}$$

$$P(M_3|Defect) = \frac{0.005}{0.005 + 0.009 + 0.010} = \frac{0.005}{0.024} \tag{22}$$

$$\mathbf{P(M_3|Defect) = 0.208}$$

Problem 4. What is the expected product of the value rolled on two fair dies?

$$E[U] = \sum_{u \in U} u \times P(U = u) \tag{23}$$

$$E[U] = \frac{1}{36} \sum_{i,j=1}^{6} i \times j = \frac{1}{6} \sum_{i=1}^{6} \frac{1}{6} \sum_{j=1}^{6} i \times j \tag{24}$$

$$\mathbf{E[U] = 12.25}$$

Problem 5. What is the expected number of die rolls before getting a 6?

$$E[X] = \sum_{x \in X} x \times P(X) \tag{25}$$

$$E[X] = \sum_{x=1}^{\infty} x \times (1-p)^{x-1} \times p \tag{26}$$

$$E[X] = p \times \sum_{x=1}^{\infty} x \times (1-p)^{x-1} \tag{27}$$

$$E[X] = p \times \left(\sum_{x=1}^{\infty} (1-p)^{x-1} + \sum_{x=2}^{\infty} (1-p)^{x-1} + \sum_{x=3}^{\infty} (1-p)^{x-1} + \dots \right) \tag{28}$$

$$E[X] = p \times \left(\sum_{x=1}^{\infty} (1-p)^{x-1} + \sum_{x=1}^{\infty} (1-p)^{x} + \sum_{x=1}^{\infty} (1-p)^{x+1} + \dots \right) \tag{29}$$

$$E[X] = p \times \left(\sum_{x=1}^{\infty} (1-p)^{x-1} + \sum_{x=1}^{\infty} (1-p)^{x-1} \times (1-p) + \sum_{x=1}^{\infty} (1-p)^{x-1} (1-p)^2 + \dots \right) \tag{30}$$

$$E[X] = p \times \left(\frac{1}{1-(1-p)} + \frac{1-p}{1-(1-p)} + \frac{(1-p)^2}{1-(1-p)} + \dots \right) \tag{31}$$

$$E[X] = p \times \left(\frac{1}{p} + \frac{1-p}{p} + \frac{(1-p)^2}{p} + \dots \right) \tag{32}$$

$$E[X] = 1 + 1 - p + (1-p)^2 + \dots \tag{33}$$

$$E[X] = \frac{1}{1-(1-p)} = \frac{1}{p} \tag{34}$$

$$\mathbf{E[X] = \frac{1}{6}}$$

Problem 6. Let X be a random variable with probability mass function:

$$f(x) = \begin{cases} \frac{x^2}{a}, & x = -3, -2, -1, 0, 1, 2, 3 \\ 0, & \text{otherwise} \end{cases}$$

Compute the value of a and E(x).

$$\sum_{x \in X} f(x) = 1 \tag{35}$$

$$\sum_{x\in[-3,-2,-1,0,1,2,3]} f(x) = 1 \tag{36}$$

$$f(-3)+f(-2)+f(-1)+f(0)+f(1)+f(2)+f(3) = 1 \tag{37}$$

$$\frac{(-3)^2}{a}+\frac{(-2)^2}{a}+\frac{(-1)^2}{a}+\frac{(0)^2}{a}+\frac{(1)^2}{a}+\frac{(2)^2}{a}+\frac{(3)^2}{a} = 1 \tag{38}$$

$$\frac{9+4+1+0+1+4+9}{a} = \frac{28}{a} = 1 \tag{39}$$

$$\mathbf{a = 28}$$

$$E(X) = \sum_{x\in X} x \times P(X = x) \tag{40}$$

$$E(X) = \sum_{x\in[-3,-2,-1,0,1,2,3]} f(x) \times \frac{1}{7} \tag{41}$$

$$E(X) = \frac{1}{7} \times (f(-3)+f(-2)+f(-1)+f(0)+f(1)+f(2)+f(3)) \tag{42}$$

$$E(X) = \frac{1}{7} \times 28 = \frac{28}{7} = 4 \tag{43}$$

$$\mathbf{E(X) = 4}$$

Problem 7. Compute the algorithmic complexity of the following algorithm.

```
int i,j,k = 0;
for (i = n/2; i <= n; i++) // Repeats n - n/2 times, or n/2.
        for (j = 2; j <= n; j *= 2) // Repeats log_2(n) - 1 times.
                k += n/2;
```

Time Complexity is O($\frac{n}{2} \times ((\log_2 n) - 1)$)

Problem 8. Find O Complexity for the following recurrence.

```
T(n) = 2T(n-1) + 1 w/ T(0) = 0
```

Can be re-written as:

```
int val, prev_val = 0;
for(i = 1; i <= n; i++) {
        int new_val = 2 * prev_val + 1;
        prev_val = new_val;
        val += new_val;
```

Therefore this recurrence is O(n)

Problem 9. 1. Find eigenvalues an eigenvectors of A and A^2

$A = \begin{bmatrix} -1 & 3 \\ 2 & 0 \end{bmatrix}$ and $A^2 = \begin{bmatrix} 7 & -3 \\ -2 & 6 \end{bmatrix}$

$$0 = det\left(\begin{bmatrix} -1-\lambda & 3 \\ 2 & 0-\lambda \end{bmatrix}\right) \tag{44}$$

$$(-1-\lambda)(-\lambda) - (2 \times 3) = \lambda^2 + \lambda - 6 = 0 \tag{45}$$

$$\lambda_1 = -3, \lambda_2 = 2 \tag{46}$$

$$\begin{bmatrix} -1-\lambda & 3 \\ 2 & 0-\lambda \end{bmatrix} \Rightarrow \begin{bmatrix} -1+3 & 3 \\ 2 & 0+3 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 3 \\ 2 & 3 \end{bmatrix} \tag{47}$$

$$\begin{bmatrix} 2 & 3 \\ 2 & 3 \end{bmatrix} \cdot \vec{v_1} = 0 \tag{48}$$

$$\vec{v_1} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} \tag{49}$$

$$\begin{bmatrix} -1-\lambda & 3 \\ 2 & 0-\lambda \end{bmatrix} \Rightarrow \begin{bmatrix} -1-2 & 3 \\ 2 & 0-2 \end{bmatrix} \Rightarrow \begin{bmatrix} -3 & 3 \\ 2 & -2 \end{bmatrix} \tag{50}$$

$$\begin{bmatrix} -3 & 3 \\ 2 & -2 \end{bmatrix} \cdot \vec{v_2} = 0 \tag{51}$$

$$\vec{v_2} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{52}$$

$$0 = det\left(\begin{bmatrix} 7-\lambda & -3 \\ -2 & 6-\lambda \end{bmatrix}\right) \tag{53}$$

$$(7-\lambda)(6-\lambda) - (-2 \times -3) = \lambda^2 - 13\lambda + 36 = 0 \tag{54}$$

$$\lambda_1 = 9, \lambda_2 = 4 \tag{55}$$

$$\begin{bmatrix} 7-\lambda & -3 \\ -2 & 6-\lambda \end{bmatrix} \Rightarrow \begin{bmatrix} 7-9 & -3 \\ -2 & 6-9 \end{bmatrix} \Rightarrow \begin{bmatrix} -2 & -3 \\ -2 & -3 \end{bmatrix} \tag{56}$$

$$\begin{bmatrix} -2 & -3 \\ -2 & -3 \end{bmatrix} \cdot \vec{v_1} = 0 \tag{57}$$

$$\vec{v_1} = \begin{bmatrix} 3 \\ -2 \end{bmatrix} \tag{58}$$

$$\begin{bmatrix} 7-\lambda & -3 \\ -2 & 6-\lambda \end{bmatrix} \Rightarrow \begin{bmatrix} 7-4 & -3 \\ -2 & 6-4 \end{bmatrix} \Rightarrow \begin{bmatrix} 3 & -3 \\ -2 & 2 \end{bmatrix} \tag{59}$$

$$\begin{bmatrix} 3 & -3 \\ -2 & 2 \end{bmatrix} \cdot \vec{v_2} = 0 \tag{60}$$

$$\vec{v_2} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{61}$$

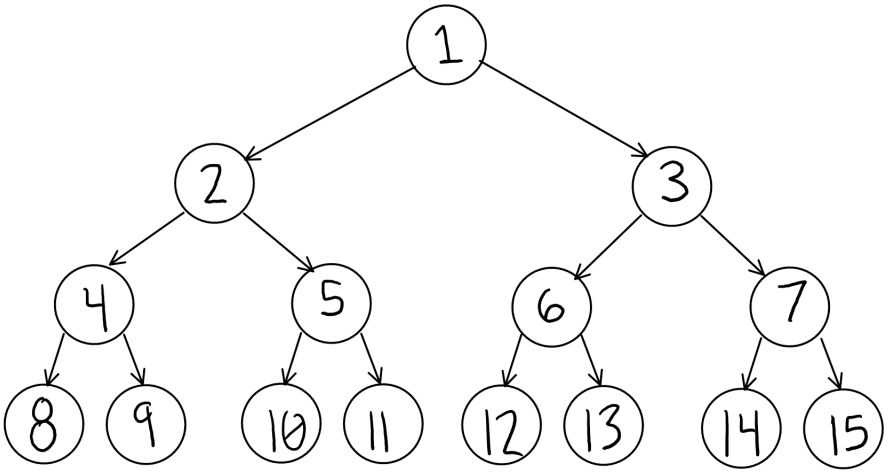

Figure 1: State space for first 15 states.

2. Derive a relationship between the eigenvalues and eigenvectors fo the corresponding powers of the matrix A.

$$A \cdot \vec{x} = \lambda \cdot \vec{x} \tag{62}$$

$$A \cdot A \cdot \vec{x} = A \cdot \lambda \cdot \vec{x} \tag{63}$$

$$A \cdot A \cdot \vec{x} = \lambda \cdot A \cdot \vec{x} \tag{64}$$

$$A \cdot A \cdot \vec{x} = \lambda \cdot (\lambda \cdot \vec{x}) \tag{65}$$

$$A^2 \cdot \vec{x} = \lambda^2 \cdot \vec{x} \tag{66}$$

Therefore the eigenvalues (λ) are squared while the eigenvectors ($\vec{x}$) remain the same.

Problem 10. Consider a state space where the start state is number 1 and the successor function for state n returns two states, numbers 2n and 2n+1

1. Draw the portion of the state space for states 1 to 15.

2. Suppose the goal state is 11. List the order in which nodes will be visited for a breadth first search, depth limited search with limit 3, and iterative deepening search.
   Breadth First Search
   Depth Limited Search
   Iterative Deepening Search

3. Would bidirectional search be appropriate for this problem? If so, describe in detail how it would work. What is the branching factor in the forward and reverse directions?

   A birectional search would be valid for this search, so long as the state transitions can be read in both directions. That is, if node two encodes both its edges to node four and five and its edge back to node one. The search would do a search of depth one from node one, followed by a search of one depth to from node eleven. This process would repeat back and forth until the target is found. In a depth first search, all fifteen nodes would have be expanded before node

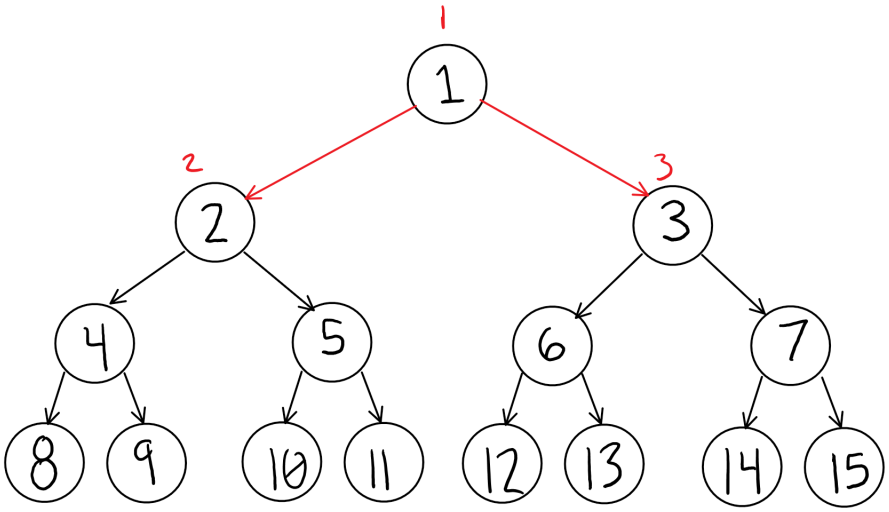

Figure 2: First stage of Breadth First Search.

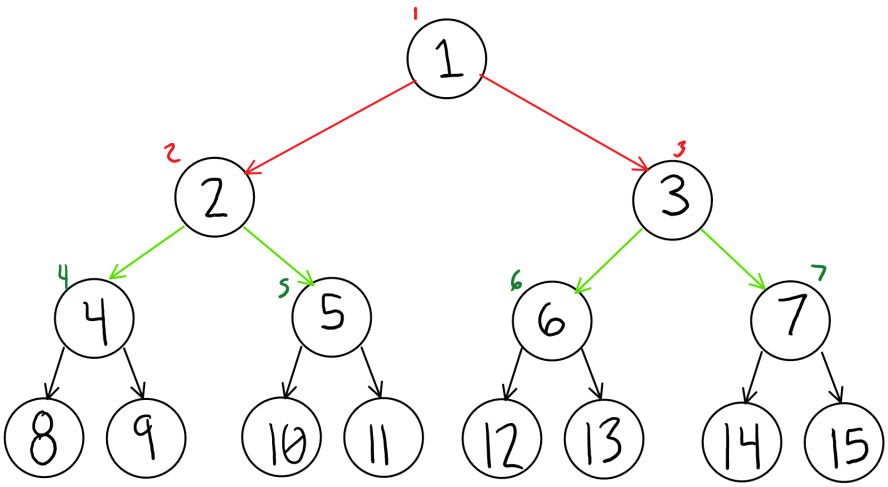

Figure 3: Second stage of Breadth First Search.

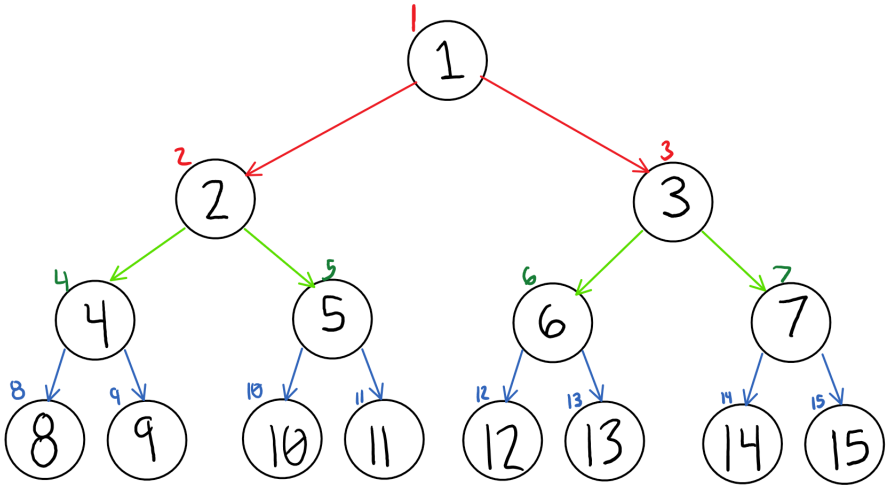

Figure 4: Third stage of Breadth First Search.

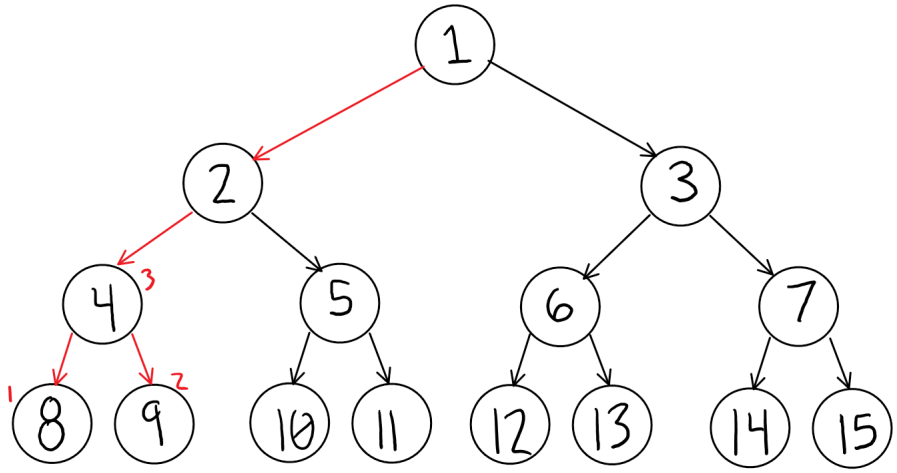

Figure 5: First stage of Depth Limited Search.

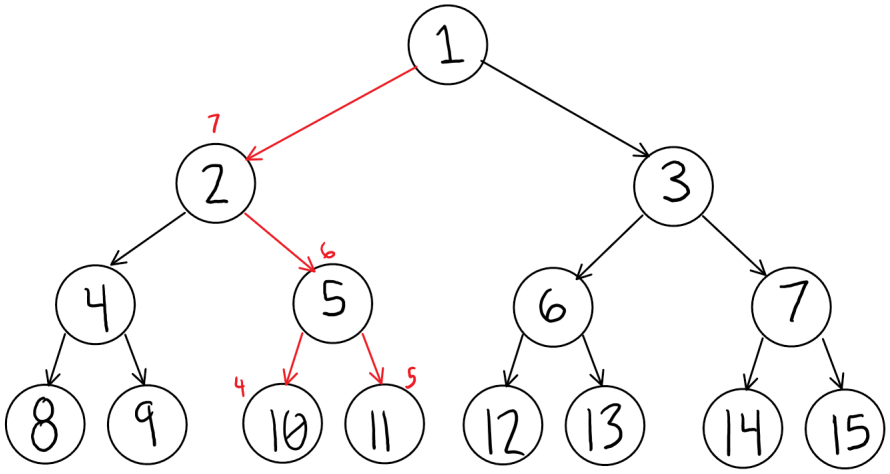

Figure 6: Second stage of Depth Limited Search.

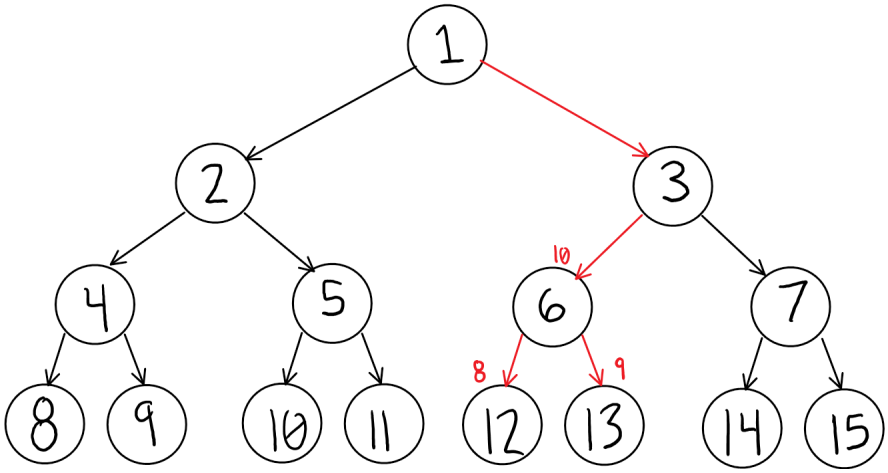

Figure 7: Third stage of Depth Limited Search.

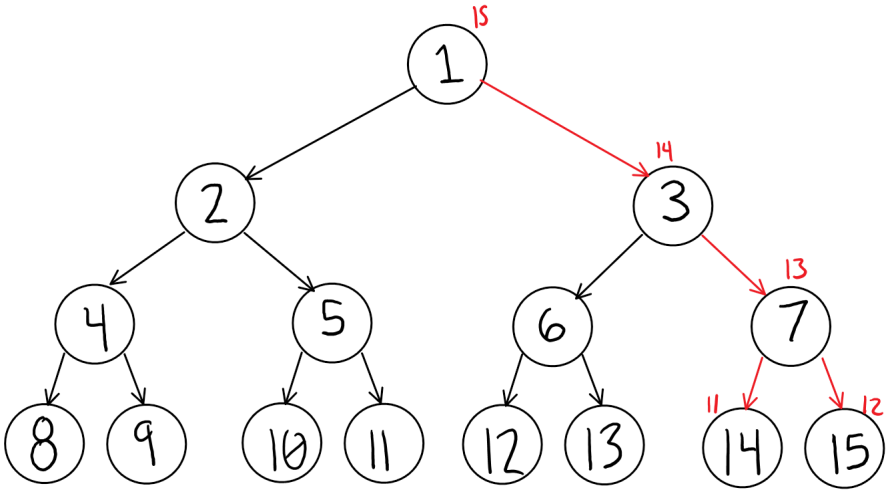

Figure 8: Final stage of Depth Limited Search.

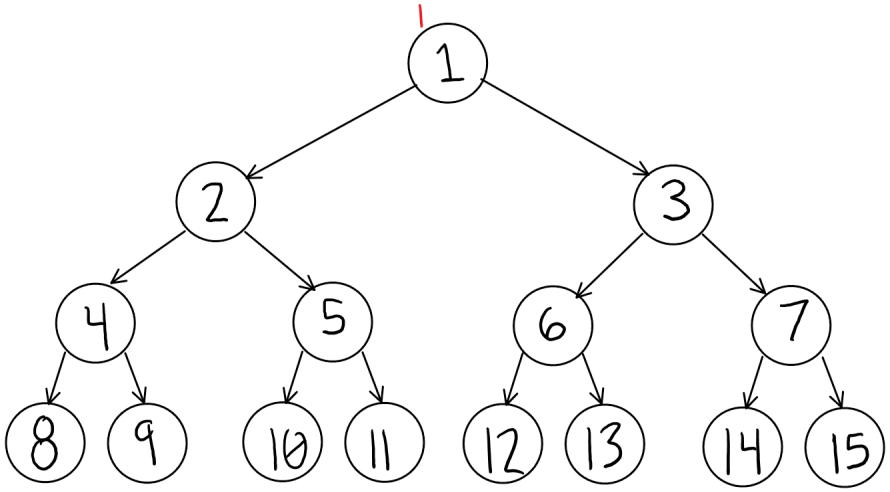

Figure 9: First stage of Iterative Deepening Search.

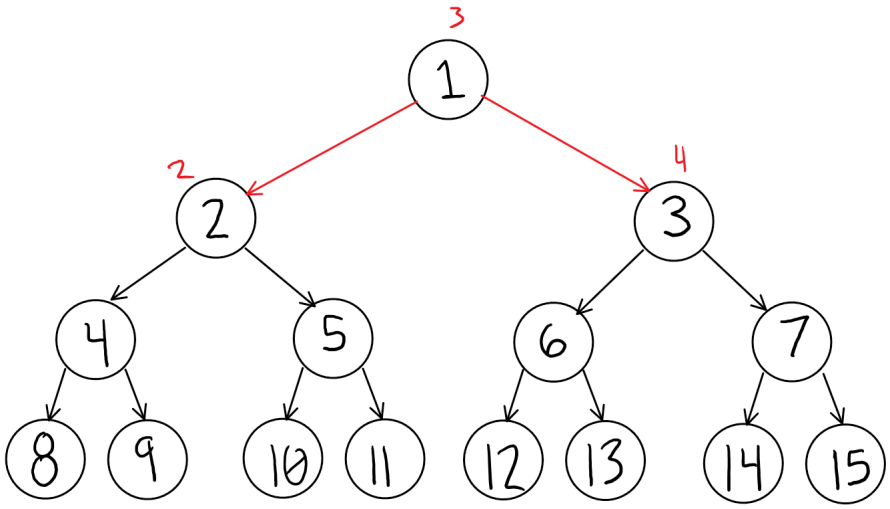

Figure 10: Second stage of Iterative Deepening Search.

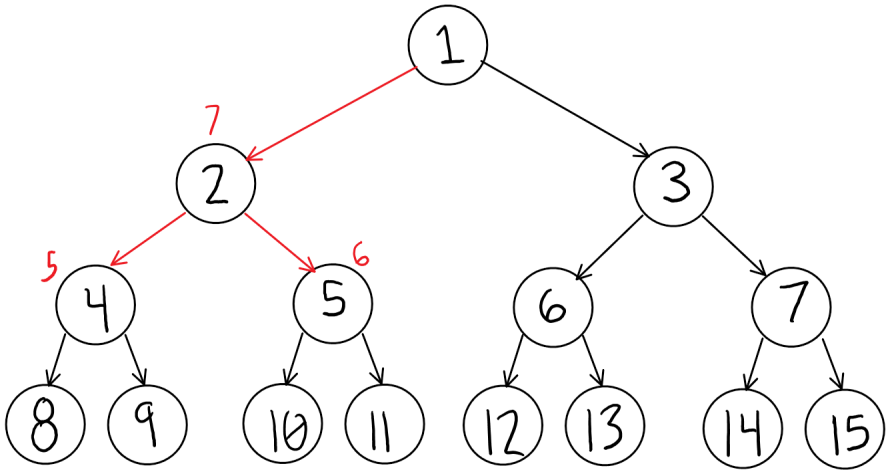

Figure 11: Third stage of Iterative Deepening Search.

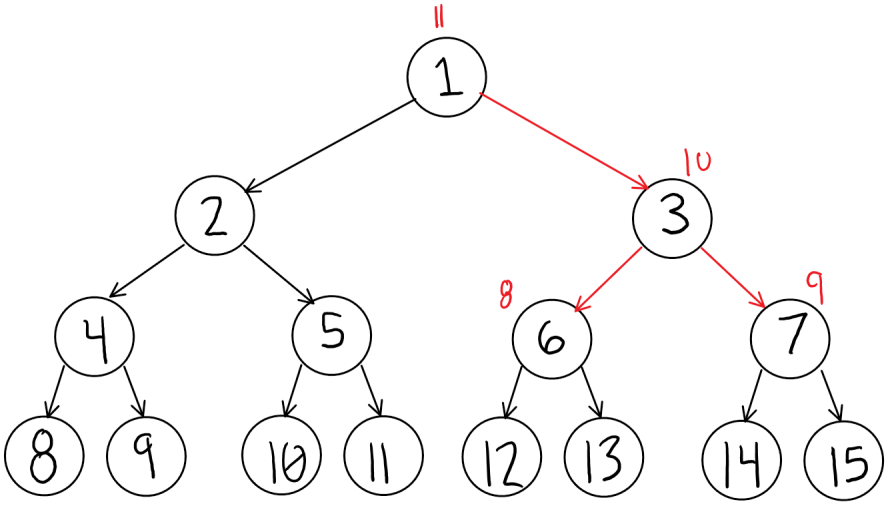

Figure 12: Fourth stage of Iterative Deepening Search.

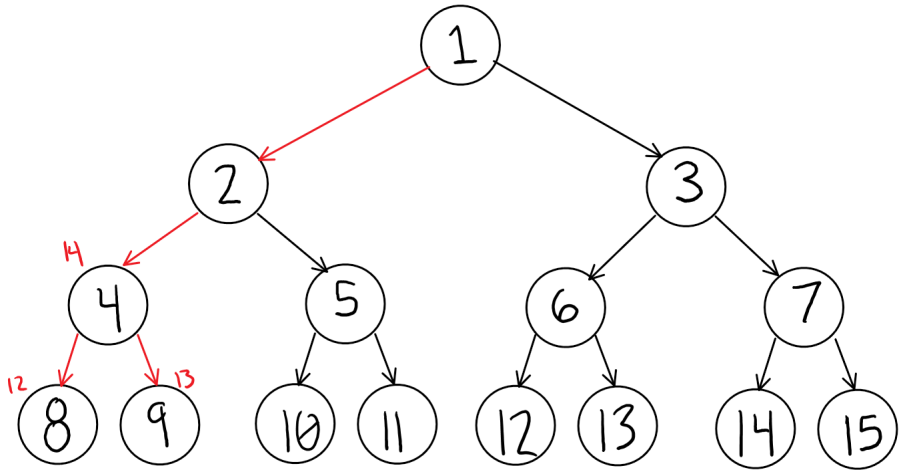

Figure 13: Fifth stage of Iterative Deepening Search.

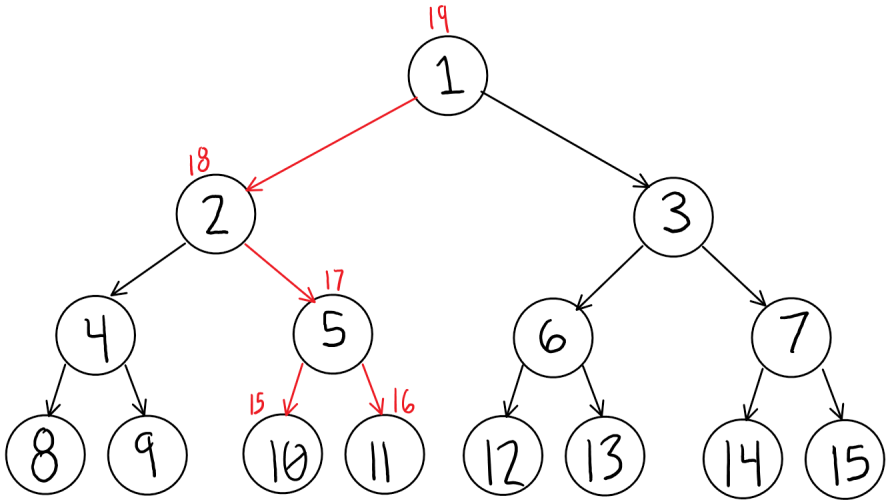

Figure 14: Sixth stage of Iterative Deepening Search.

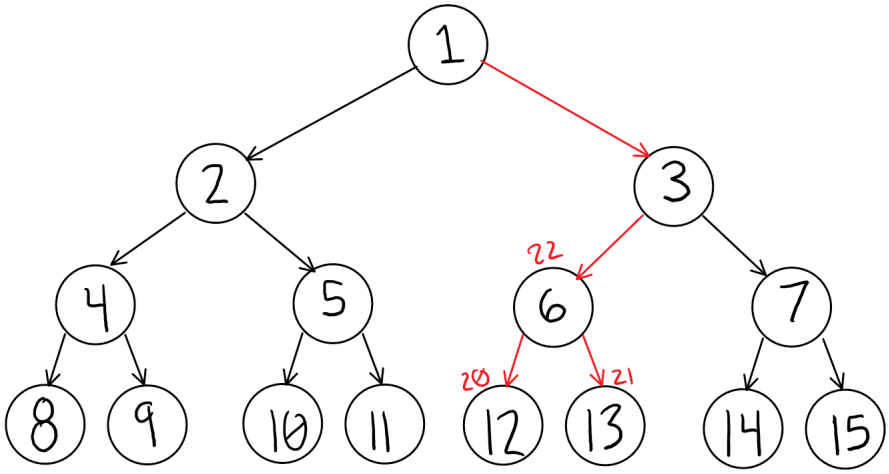

Figure 15: Seventh stage of Iterative Deepening Search.

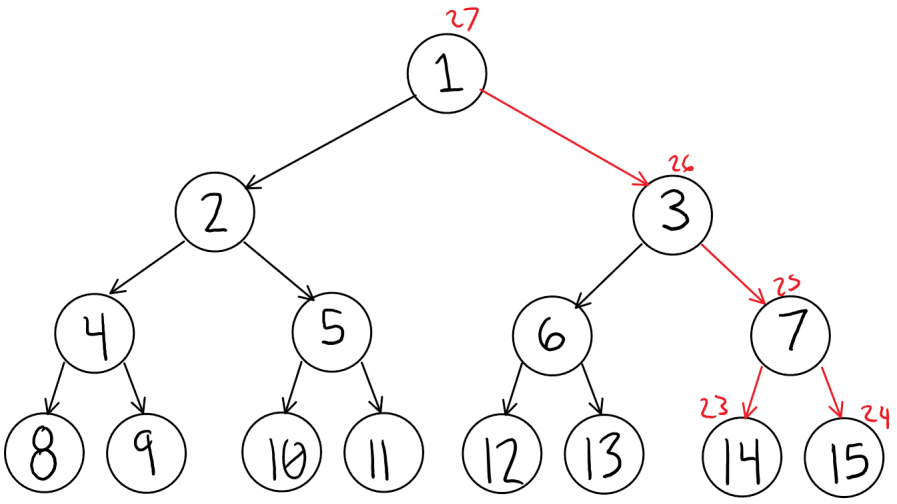

Figure 16: Final stage of Iterative Deepening Search.

eleven was found. In iterative deepening and depth limited, this would require twenty and eight nodes to be expanded respectively. For a bidirectional search, however, node one would be expanded along with nodes two and three. Node eleven would be expanded with nodes five. Finally, node two would expand nodes four and five with three expanding nodes six and seven for a total of eight nodes expanded. A standard graph search operates in $O\left(b^d\right)$, where d is the depth and b is the branching factor. A bidirectional stearch, however, reduces the depth to half for two separate paths, reducing the complexity to $O\left(2b^{\frac{d}{2}}\right)$. In this case, since our branching factor is 2, this results in a total time complexity of $O\left(b^{\frac{d}{2}+1}\right)$.

4. Does the answer to (c) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

   Performing a reverse search from the goal node will give the direct path, in reverse, from the start node. This is because each node in the tree only has one parent, so moving backwards from 11 returns node 5, whose parent is node 2, and finally node 1. For this to be possible with a standard search each node must encode not only its successors but its predecessors. Since the transitions follow a specific, defined rule set it is possible to traverse the tree backwards with only mathematical operations. By this method, starting from our goal node, each node in the path before it is $\lfloor \frac{NODE}{2} \rfloor$. In our example, our path from node eleven would be derived as $\lfloor \frac{11}{2} \rfloor = 5, \lfloor \frac{5}{2} \rfloor = 2, \lfloor \frac{2}{2} \rfloor = 1$.