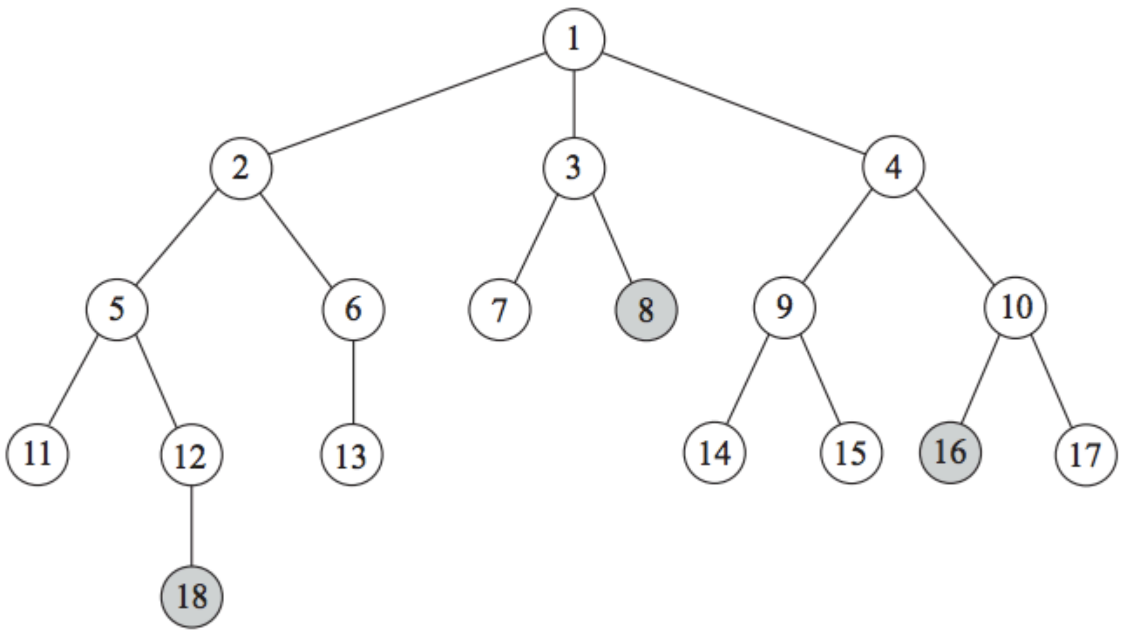


Original Search Tree

**1. Breadth-First Search**

1 2 3 4 5 6 7 **8**

Frontier:

1) ~~1~~
2) ~~2~~ 3 4
3) ~~3~~ 4 5 6
4) ~~4~~ 5 6 7 8
5) ~~5~~ 6 7 8 9 10
6) ~~6~~ 7 8 9 10 11 12
7) ~~7~~ 8 9 10 11 12 13
8) **8** 9 10 11 12 13

**2. Depth-First Search**

1 2 5 11 12 **18**

Frontier:

1) ~~1~~
2) ~~2~~ 3 4
3) ~~5~~ 6 3 4
4) ~~11~~ 12 6 3 4
5) ~~12~~ 6 3 4
6) **18** 6 3 4

<u>Alternative (based on order to visit child nodes):</u>

1 4 10 17 **16**
Frontier:

1) ~~1~~
2) ~~4~~ 3 2
3) ~~10~~ 9 3 2
4) ~~17~~ 16 9 3 2
5) **16** 9 3 2

**3. Iterative Deepening Depth-First Search**

**Iteration 1**
Visited Node: 1
Frontier:

1) ~~1~~

**Iteration 2**
Visited Node: 1 2 3 4
Frontier:

1) ~~1~~
2) ~~2~~ 3 4
3) ~~3~~ 4
4) ~~4~~

**Iteration 3**
Visited Node: 1 2 5 6 3 7 **8**
Frontier:

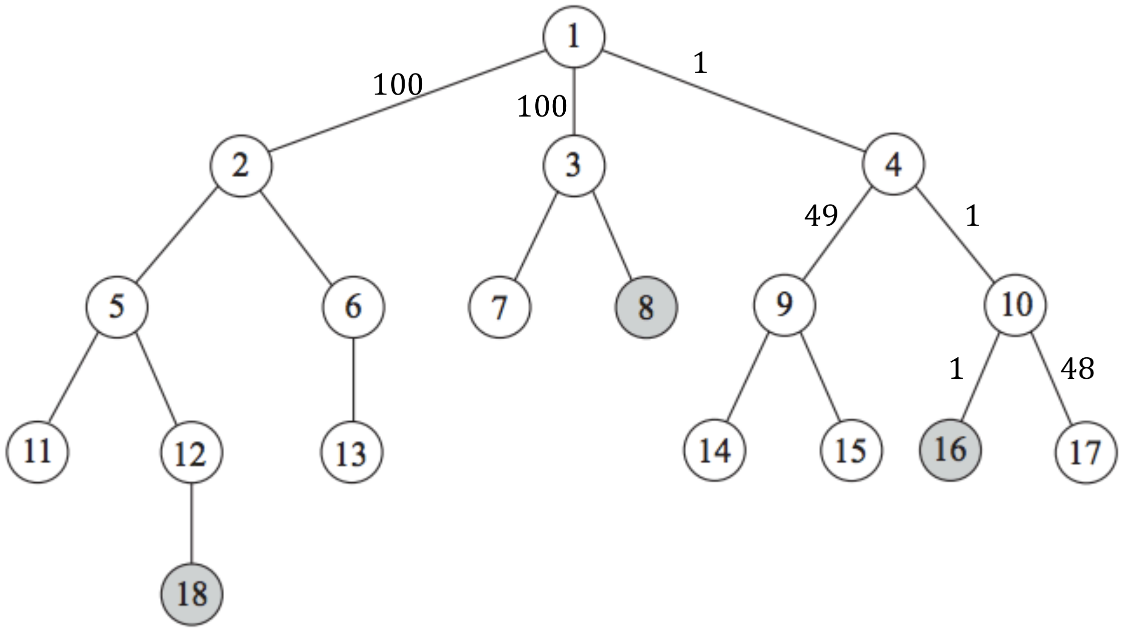
1) ~~1~~
2) ~~2~~ 3 4
3) ~~5~~ 6 3 4
4) ~~6~~ 3 4
5) ~~3~~ 4
6) ~~7~~ 8 4
7) **8** 4

4. Select a search algorithm with the property that node 16 will be selected as the solution (you may make any appropriate modification of the search tree as needed).

Use uniform-cost search algorithm and assign cost to the search tree edges to ensure 16 will be visited first. For example:
1 4 10 **16**
Frontier:

1) ~~1(0)~~
2) ~~4(1)~~ 2(100) 3(100)
3) ~~10(2)~~ 9(50) 2(100) 3(100)

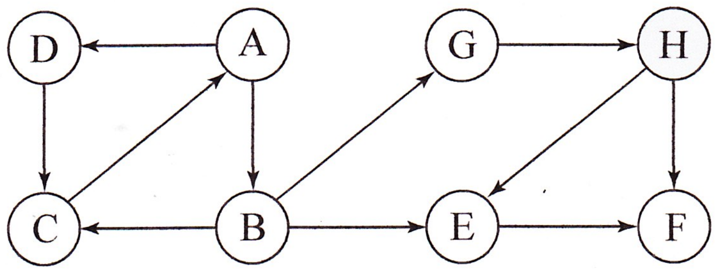
4) **16**(3) 17(50) 9(50) 2(100) 3(100)

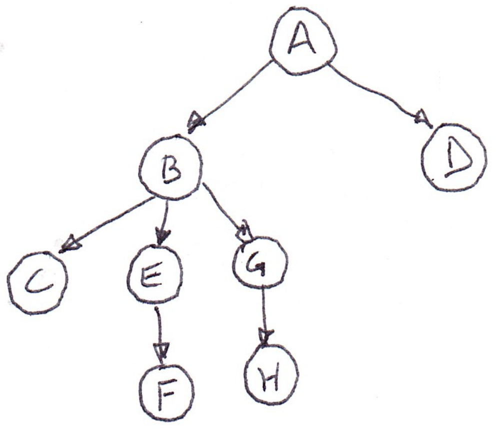


Modified Search Tree: un-assigned edges can be any non-negative cost

**Question 2**. Consider the following graph. Assume we start the search at state "A" with goal state "H." At any point during search, if ties need to be broken, the node which is closest to the start of the alphabet (closest to A) will be selected. List the order in which nodes will be visited for the following two search methods:

1. Breadth-First Search
2. Depth-First Search





BFS:

FRONTIER

~~A~~
~~B~~ D
~~D~~ C E G
~~C~~ E G
~~E~~ G
~~G~~ F

VISITED: A B D C E G H

DFS:

FRONTIER

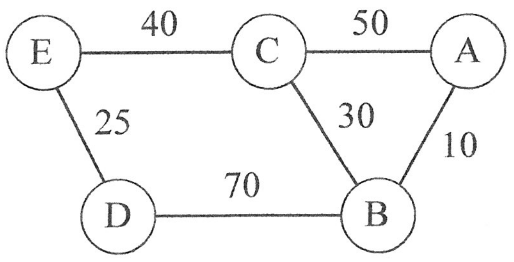
~~A~~
~~B~~ D
~~C~~ E G D
~~E~~ G D
~~F~~ G D
~~G~~ D
~~H~~ D

VISITED: A B C E F G H

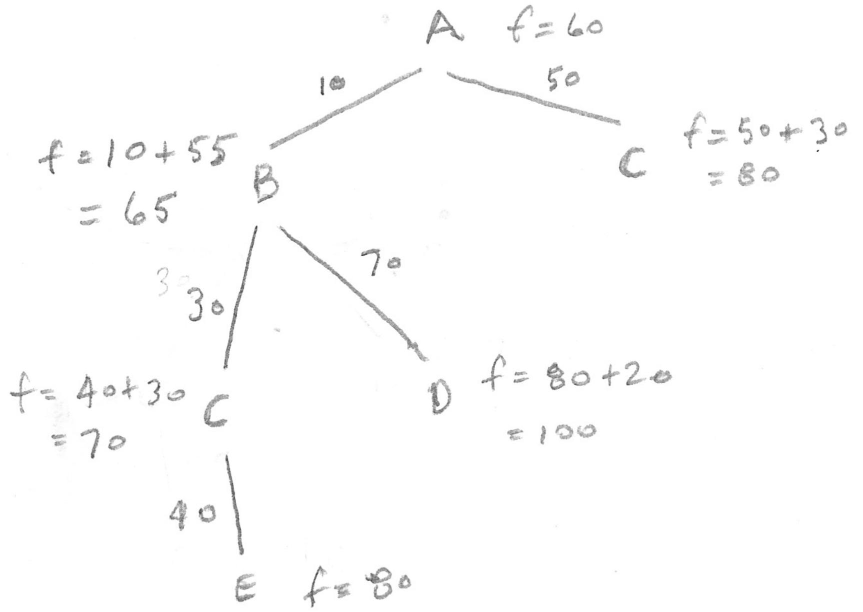
**Question 1.** Use **A* Search** to solve the map problem below, where the numbers give the distances between states.

- **Initial State** = A
- **Goal State** = E
- The value of the heuristic function for each node is given in the table on the left.

| $n$ | $h(n)$ |
|---|---|
| A | 60 |
| B | 55 |
| C | 30 |
| D | 20 |
| E | 0 |



**(a)** Draw the search tree for A*, showing the evolving frontier.



FRONTIER

A(~~60~~)

B(~~65~~) C(80)

C(~~70~~) D(100)

E(~~80~~) D(100)

Note: This node replaced by C(70) in frontier once B is expanded.

**(b)** Indicate below the order in which nodes will be expanded (i.e. put in the explored set)

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| A | B | C | E | | | | | | |

**(c)** For each of the possible node expansion orders shown below, indicate which *uninformed search algorithms* could have produced the given node order when applied to the graph above. Note: Be explicit about how you are breaking ties for nodes at the same depth.

| 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|
| A | C | E | | | BFS A↑ & DFS A↑ |
| A | B | C | D | E | UNIFORM COST SEARCH |
| A | B | C | E | | BFS A↓ |
| A | C | B | E | | ITERATIVE DEEP DFS A↑ |

Notes: Let A↓ = BREAK TIES LOWEST LETTER
A↑ = BREAK TIES HIGHEST LETTER

We follow convention from book: BFS checks for goal before insertion into frontier, while DFS, ITERATIVE DEEPENING DFS, and COST SENSITIVE all check for goal node on popping from frontier.

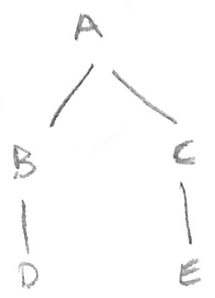
See next page for work.

**(d)** Show that the heuristic function for this problem satisfies the *consistency* criteria.

$h(n) \leq c(n,a,n') + h(n') \quad \forall n, n'$

| $n'$ | $n$ | $c+h(n')$ | | $h(n)$ | |
|---|---|---|---|---|---|
| E | C | 40 | | 30 | |
| E | D | 25 | | 20 | |
| D | B | 90 | ≥ | 55 | ✓ |
| C | A | 80 | | 60 | |
| C | B | 60 | | 55 | |
| B | A | 65 | | 60 | |

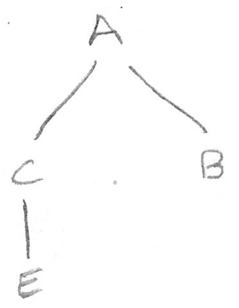
BFS A↓ VISITED: A B C E



FRONTIER

~~A~~
~~B~~ C
~~C~~ D

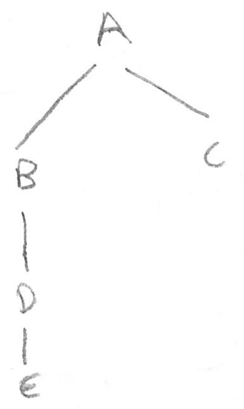
BFS A↑ VISITED: A C E



FRONTIER

~~A~~
~~C~~ B

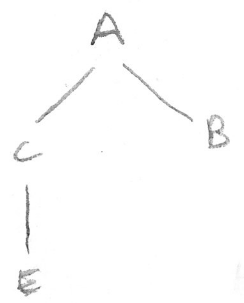
DFS A↓ VISITED: A B D E



FRONTIER

~~A~~
~~B~~ C
~~D~~ C
~~E~~ C

DFS A↑ VISITED: A C E

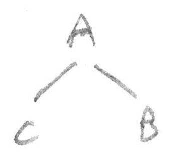


FRONTIER

~~A~~
~~C~~ B
~~E~~ B

ITER-DEEP DFS A↑ V: A C B E

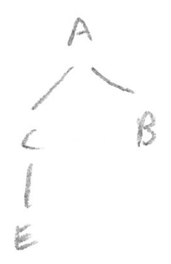
ITER #1:



FRONTIER

~~A~~
~~C~~ B
~~B~~

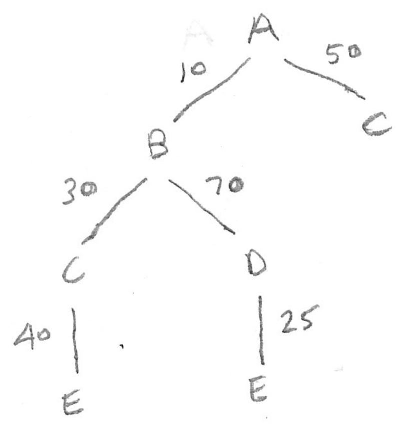
ITER #2:



FRONTIER

~~A~~
~~C~~ B
~~E~~ B

UNIFORM COST SEARCH V: A B C D E

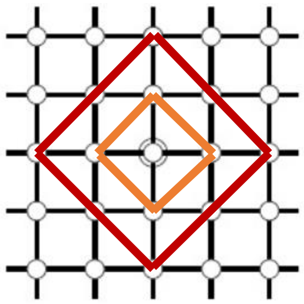


FRONTIER

A(~~0~~)
B(~~10~~) C(50)
C(~~40~~) D(80)
D(~~80~~) E(80)
E(~~80~~) E(105)

FINAL PATH:
A B C E

**4. (3.26 in textbook) Consider the unbounded version of the regular 2D graph shown in figure 3.9. The start state is at the origin, (0, 0), and the goal state is (x, y).**



a. *What is the branching factor b in this state space?*
   - Branching factor: maximum number of successors of any node
   - The branching factor for this graph is 4

b. *How many distinct states are there at depth k (for k > 0)?*
   - This question is asking, "how many distinct states exist at depth k, meaning k steps away from the origin in some combination of x and y directions?"
   - See the figure above right, the orange lines denote the "frontier" of states reachable in 1 step, and the red lines donate the frontier reachable in 2 steps. You can see that all states on the frontier or inside the frontier will be reachable, since you are allowed to loop back to interior states.
   - The number of reachable states is $1 + 2k(k + 1)$

   You can derive this answer as follows: First, you can see that the "length" of the square defined by the frontier is $2k$. This is the number of reachable states which are the farthest from the origin in terms of steps. Second, since all of the frontiers at sizes below k are also reachable, the total reachable states are given by:

$$1 + \sum_{j=1}^{k} 4j = 1 + 4\sum_{j=1}^{k} j = 1 + \frac{4k(k+1)}{2} = 1 + 2k(k+1),$$

   where the 2nd equality follows from a standard formula for the sums of the positive integers. Note that the added 1 comes from including the state at the origin.
   Also note that your textbook says that there are approximately $2d^2$states within a distance d of a point on an infinite grid. Here we have derived the exact answer.

c. What is the maximum number of nodes expanded by breadth-first tree search?
   - $O(b^d)$, or the branching factor raised to the depth
   - This equals $O(4^{x+y})$ in this example

d. What is the maximum number of nodes expanded by breadth-first graph search?
   - To solve this, we use the same formula from part (b), but for a point (x, y), instead of a depth k. Since all the points on a diagonal are at the same depth, this depth equals x + y.
   - Substituting (x + y) for k, we get $1 + 2(x + y)(x + y + 1)$.

e. Is $h = |u - x| + |v - y|$ an admissible heuristic for a state at (u, v)? Explain.
   - Yes. This heuristic is the length of the path from the state to the goal.

f. How many nodes are expanded by A* graph search using h?

    - x + y, since the heuristic in (e) is perfect.

g. Does h remain admissible if some links are removed?
    - Yes. Removing links makes the path longer, so h becomes an underestimate.

h. Does h remain admissible if some links are added between adjacent states?
    - No. Adding some links could decrease the optimal path length by allowing shortcuts. In this case our heuristic is pessimistic.