## Problem Formulation of Missionaries & Cannibals

State Space: All combinations between $(3,3,1)$ and $(0,0,0)$ where $(m, c, b)$
$(missionaries\ left\ to\ be\ transported, cannibals\ left\ to\ be\ transported, boolean\ for\ position\ of\ boat)$

Initial State: $(3,3,1)$ which means all 3 cannibals and missionaries are not yet transported and the boat is on the starting side of the river (where all 6 passengers are located)

Goal Test: $(0,0,0)$ meaning all 6 passengers are transported and the boat is on the end side.

Actions: Alternating arithmetic subtraction and addition with the following combination:

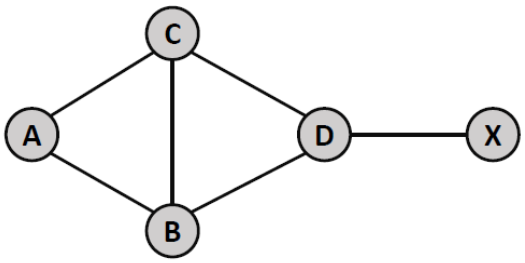
- Subtraction of form $(m, c. 1)$ from the current state of form $(m^*, c^*, 1)$. i.e., the boat crosses the river from start to end side with $m, c$ passengers where $1 \leq m + c \leq 2$
- Addition of form $(m, c. 1)$ to the current state of form $(m^*, c^*, 0)$. i.e., the boat crosses the river from ending side back to start side with $m, c$ passengers where $1 \leq m + c \leq 2$.
- Invalid actions are those that result in the resulting form $(m^*, c^*, b^*)$ where $m^* < c^*$

Path Cost: Number of actions taken. Each arithmetic operation counts as 1 cost.

## Tree Search vs Graph Search

a) Graph Search keeps track of explored nodes. Tree Search does not keep track of explored nodes.
b) State are the representations of the physical configurations of the search problem. Whereas nodes are the data structure that are part of the search tree/graph and it contains, amongst others: the state, parent node, child node action, etc.
c) It keeps track of nodes that have been expanded and not states. It keeps track of expanded/explored nodes because a node can be reach by more than 1 path, keeping track of these expanded nodes prevent infinite loops.

## BFS & DFS



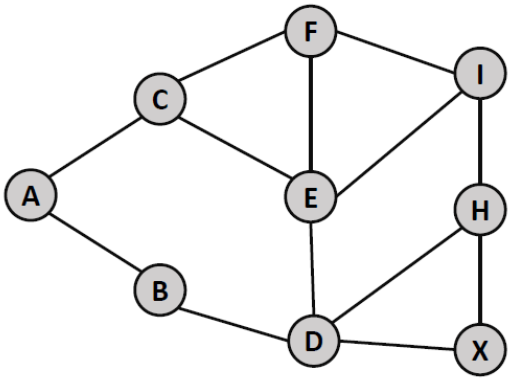

a) Run BFS Graph Search:
    1. Frontier: A (Explored: )
    2. Frontier: AB, AC (Explored: A)
    3. Frontier: AC, ABC, ABD (Explored: A, B)
    4. Frontier: ABC, ABD, ACD (Explored: A, B, C)
        a. Note: C is in explored, ABC skipped. So, expand ABD instead.
    5. Frontier: ACD, ABDX (Explored: A, B, C, D)
        a. Note: ABDX is chosen when goal-tested because X is the solution.
    6. Solution: ABDX

b) Run DFS Graph Search:
   1. Frontier: A (Explored: )
   2. Frontier: AB, AC (Explored: A)
   3. Frontier: AB, ACB, ACD (Explored: A, C)
   4. Frontier: AB, ACB, ACDB, ACDX (Explored: A, C, D)
      a. Note: ACDX is chosen when goal-tested because X is the solution.
   5. Solution: ACDX

c) BFS Tree Search's additional nodes:
   1. ABA when expanding B
   2. ACA when expanding C
   3. ACB when expanding C

d) DFS Tree Search's additional nodes:
   1. ACA when expanding C
   2. ACDC when expanding D
   3. Nothing else, only the 2 above.

## More BFS/DFS

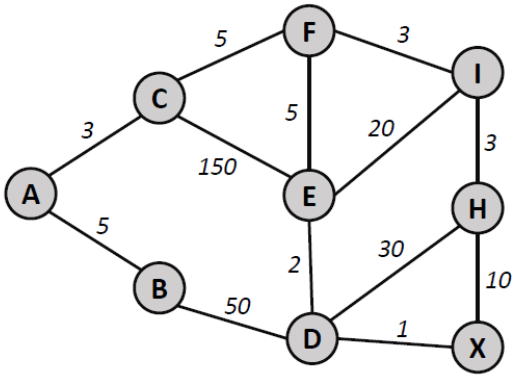



a) Run BFS Graph Search:
   1. Frontier: A (Explored: )
   2. Frontier: AB, AC (Explored: A)
   3. Frontier: AC, ABD (Explored: A, B)
   4. Frontier: ABD, ACE, ACF (Explored: A, B, C)
   5. Frontier: ACE, ACF, ABDE, ABDH, ABDX (Explored: A, B, C, D)
      a. Note: ACDX is chosen when goal-tested because X is the solution.
   6. Solution: ABDX

b) Run DFS Graph Search:
   1. Frontier: A (Explored: )
   2. Frontier: AB, AC (Explored: A)
   3. Frontier: AB, ACE, ACF (Explored: A, C)
   4. Frontier: AB, ACE, ACFE, ACFI (Explored: A, C, F)

5. Frontier: AB, ACE, ACFE, ACFIE, ACFIH (Explored: A, C, F, I)
6. Frontier: AB, ACE, ACFE, ACFIE, ACFIHD, ACFIHX (Explored: A, C, F, I, H)
    a. Note: ACFIHX is when goal-tested chosen because X is the solution.
7. Solution: ACFIHX

## Uniform Cost Search




Run DFS Graph Search:

1. Frontier: A (Explored: )
2. Frontier: AC (3), AB (5) (Explored: A)
3. Frontier: AB (5), ACF (8), ACE (153) (Explored: A, C)
4. Frontier: ACF (8), ABD (55), ACE (153) (Explored: A, C, B)
5. Frontier: ACFI (11), ACFE (13), ABD (55), ACE (153) (Explored: A, C, B, F)
6. Frontier: ACFE (13), ACFIH (14), ACFIE (31), ABD (55), ACE (153) (Explored: A, C, B, F, I)
7. Frontier: ACFIH (14), ACFED (15), ACFIE (31), ABD (55), ACE (153) (Explored: A, C, B, F, I, E)
8. Frontier: ACFED (15), ACFIHX (24), ACFIE (31), ACFIHD (44), ABD (55), ACE (153) (Explored: A, C, B, F, I, E, H)
    - Note: although ACFIHX is solution, it is not the lowest cost in Priority Queue, so it will not be popped and goal tested. Continue checking.
9. Frontier: ACFEDX (16), ACFIHX (24), ACFIE (31), ACFIHD (44), ABD (55), ACE (153) (Explored: A, C, B, F, I, E, H. D)
    - Note: ACFEDX is lowest cost in Priority Queue, it will be popped in next iteration and goal-tested to be true because X is goal. Thus, making it the optimal solution.
10. Solution: ACFEDX