Lab2 Solution

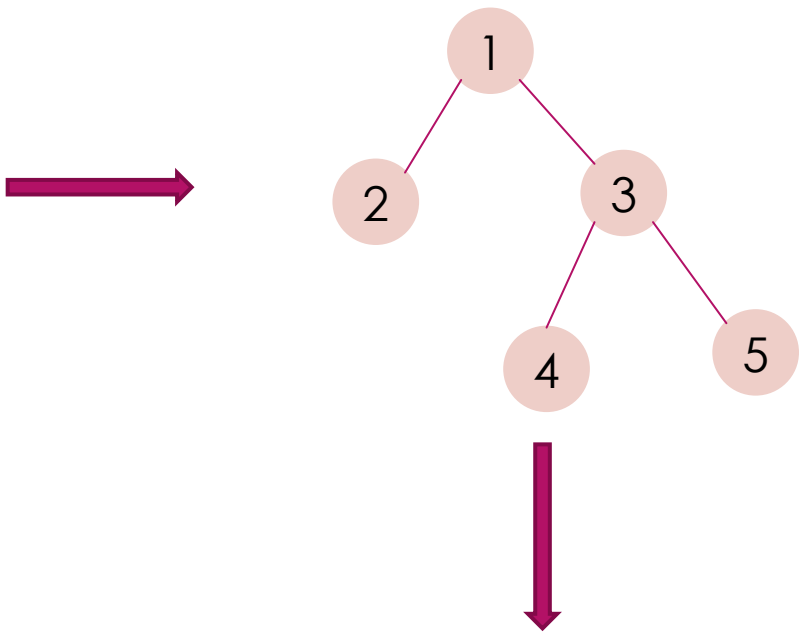
YAO ZHAO

Lab2.A: A Bunny at Large

- Bunnies are large, rabbit-like animals which are found in SUSTech. When reports came into Satori's bunny shop that a wild bunny had been spotted forty-five miles south of SUSTech, they were not taken seriously. However, as the evidence began to accumulate, Satori from the shop felt obliged to investigate, for the descriptions given by people who claimed to have seen the bunny were extraordinarily similar.
- Satori sketched a map in her mind about the places her bunny would possibly stay. The map can be seen as a connected undirected graph with N nodes and $N-1$ edges. Then she would begin a **breadth-first search(BFS)** around the whole map, starting from node 1.
- Can you tell her the number of possible BFS orderings? As the answer might be extremely large, you should output the answer module 998244353.
- More details about BFS: https://en.wikipedia.org/wiki/Breadth-first_search

Sample Input 1

```
5
1 2
1 3
3 4
3 5
```



All possible BFS orderings:

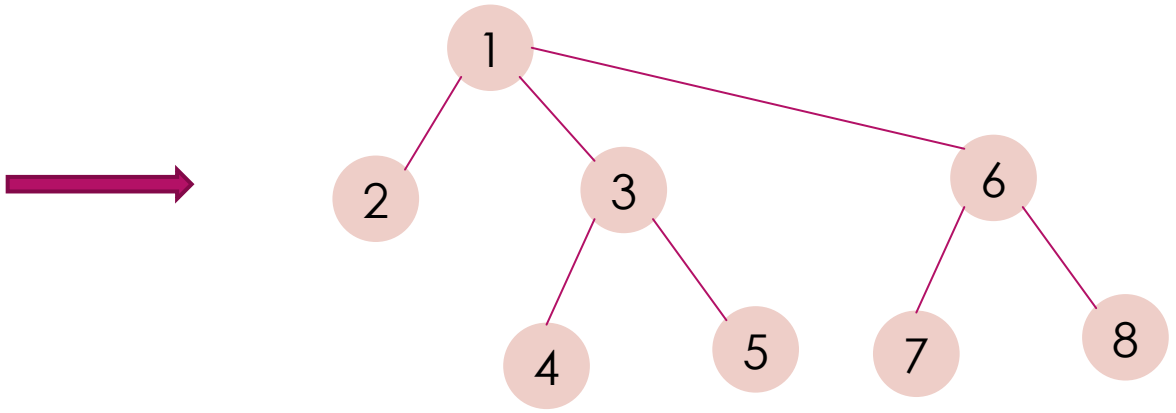
1->2->3->4->5
1->3->2->4->5
1->2->3->5->4
1->3->2->5->4

Sample Output 1

4

Sample Input 2

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



First level

1 -> 2 3 6 ->

2->3->6
2->6->3
3->6->2
3->2->6
6->2->3
6->3->2

6 possible order

Second level

4 5

7 8

4->5 **7->8**

5->4 **8->7**

4 possible order

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

1 6 3 2 8 7 5 4

Sample Output 2

24

Solution：

```
Boolean BFS(Node S){
    queue.init();
    queue.add(S);
    result = 1;
    while(!queue.empty()){
        e = queue.remove();
        e.setVisited();
        count = 0;
        while(e.hasNextChild){
            child = e.getNextChild();
            if (!child.isVisited()&&!queue.exist(child)){
                queue.add(child);
                count = count + 1;
                result = result * count;
             }
        }//end while
    }//end while
} //end BFS
```

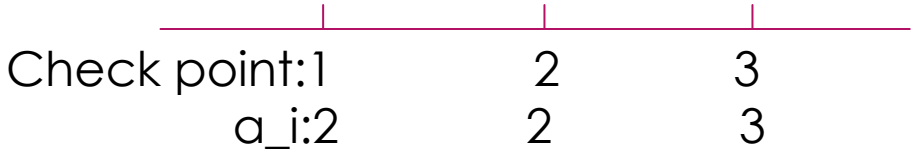
Lab2.B:The Beginning of My Great Adventure

- FluffyT, the super bunny, successfully escaped from Satori's bunny shop. She then ran into an alley with N check points.
- As FluffyT is a super bunny, she can build a portal between check point i and a_i using 1 second and travel through within no time. However, she cannot travel in the opposite direction (from a_i to i). Also, she can run from check point i to check point $i-1$ and $i+1$ using 1 second.
- FluffyT is currently at check point 1. Can you tell her the minimum time to get to each check point?

minimum time

Sample Input 1

3
2 2 3



Check point 1: 1->1 0

Check point 2:
1->2 1
1->a_1=2 1

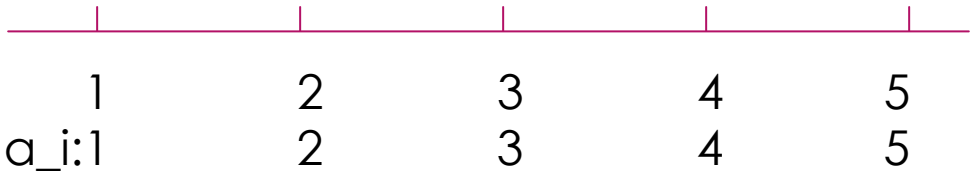
Check point 3:
1->2 ->3 2
1->a_1=2->3 2

Sample Output
0 1 2

minimum time

Sample Input 2

3
1 2 3 4 5



Check point 1: 1 0

Check point 2: 1->2 1

Check point 3: 1->2 ->3 2

Check point 4: 1->2->3->4 3

Check point 5: 1->2->3->4->5 4

Sample Output 2
0 1 2 3 4

Sample Input 3

7
4 4 4 4 7 7 7

minimum time

Check point 1: 1 0

Check point 2: 1->2 1

Check point 3: 1->2 ->3 2

Check point 4: 1->a_1:4 1

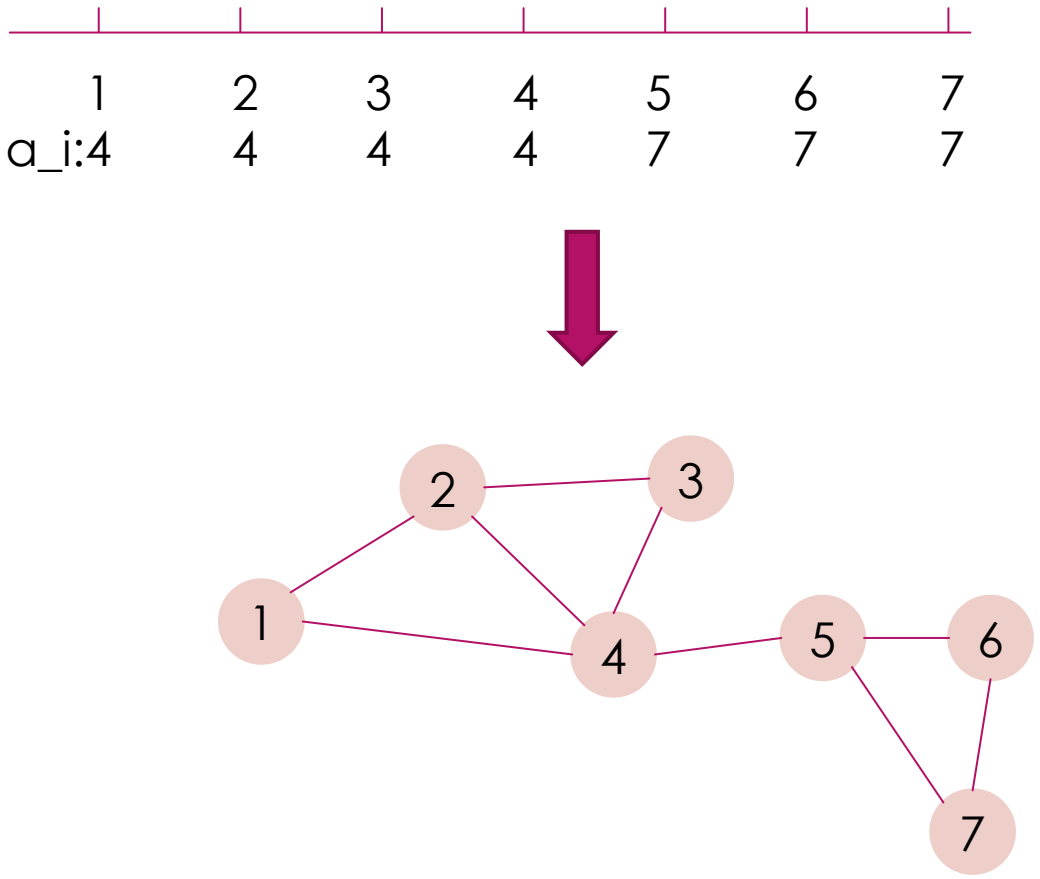
Check point 5: 1-> a_1:4 ->5 2

Check point 6: 1-> a_1:4 ->5 ->6 3

Check point 7: 1-> a_1:4 ->5 ->a_5:7 3

Sample Output 3
0 1 2 1 2 3 3

Hint:



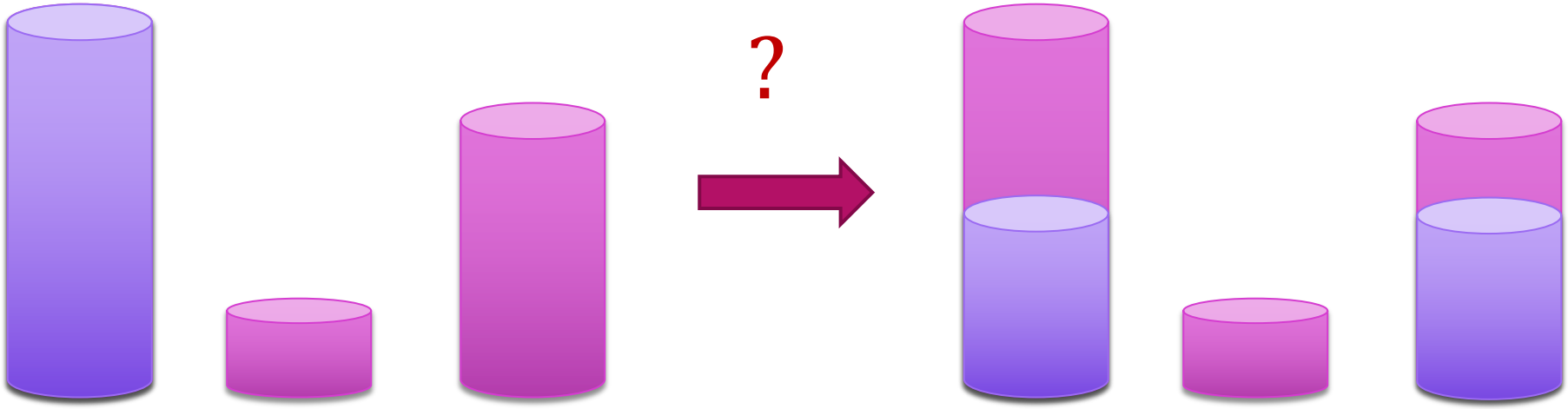
More questions

Dividing water

- Lanran has a large bottle of water with s ml, and 2 cups with capacity n, m ml. He wants to divide all his water equally into 2 containers to give to his 2 friends. Because there is no scales on the cups, during one operation Lanran can only pour a full cup of water. Now he wants to know what is the minimum operations he should do to divide these water equally. If Lanran can not do that, please output 'impossible'(without quotes)

Dividing water

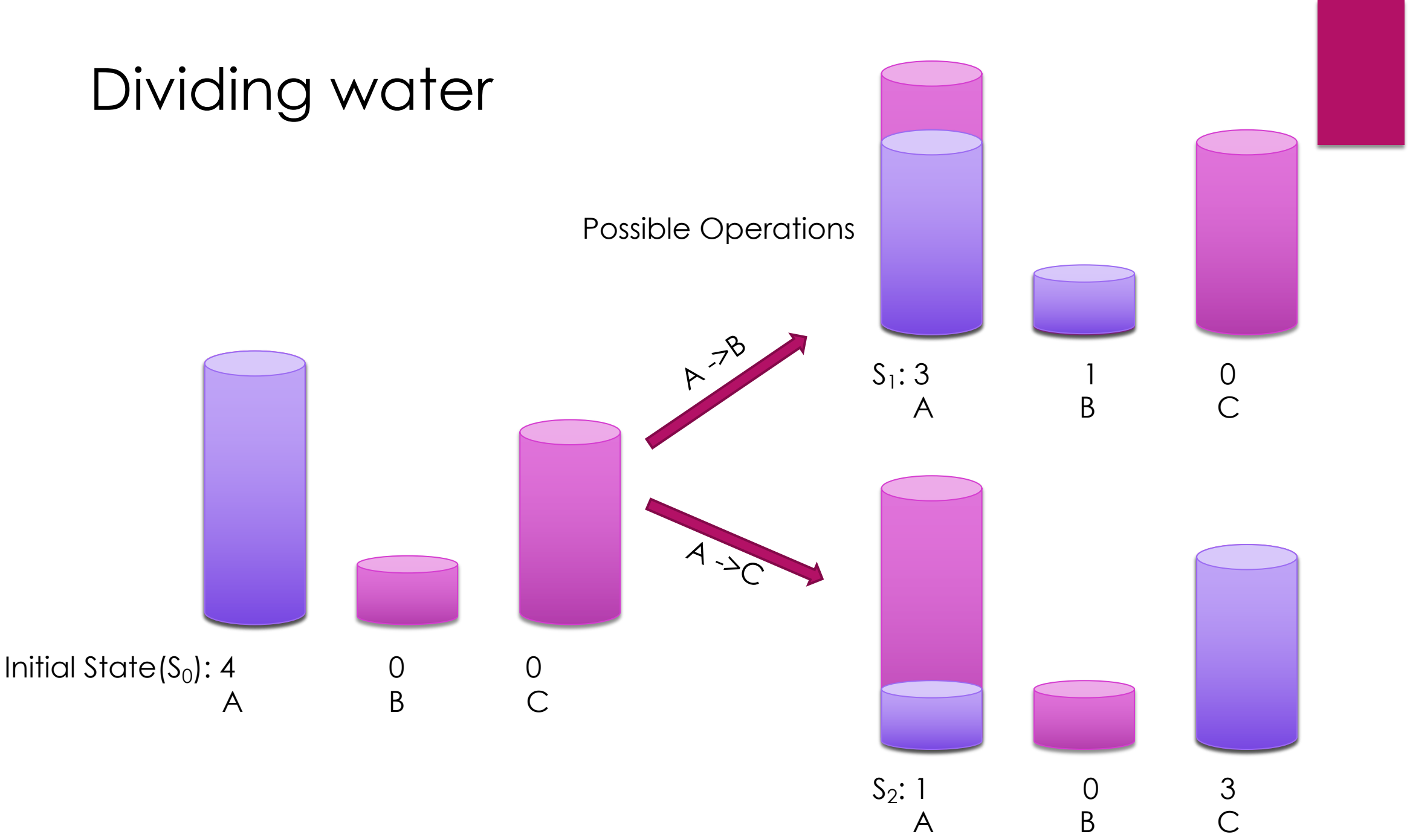
Container Capacity: 4 1 3



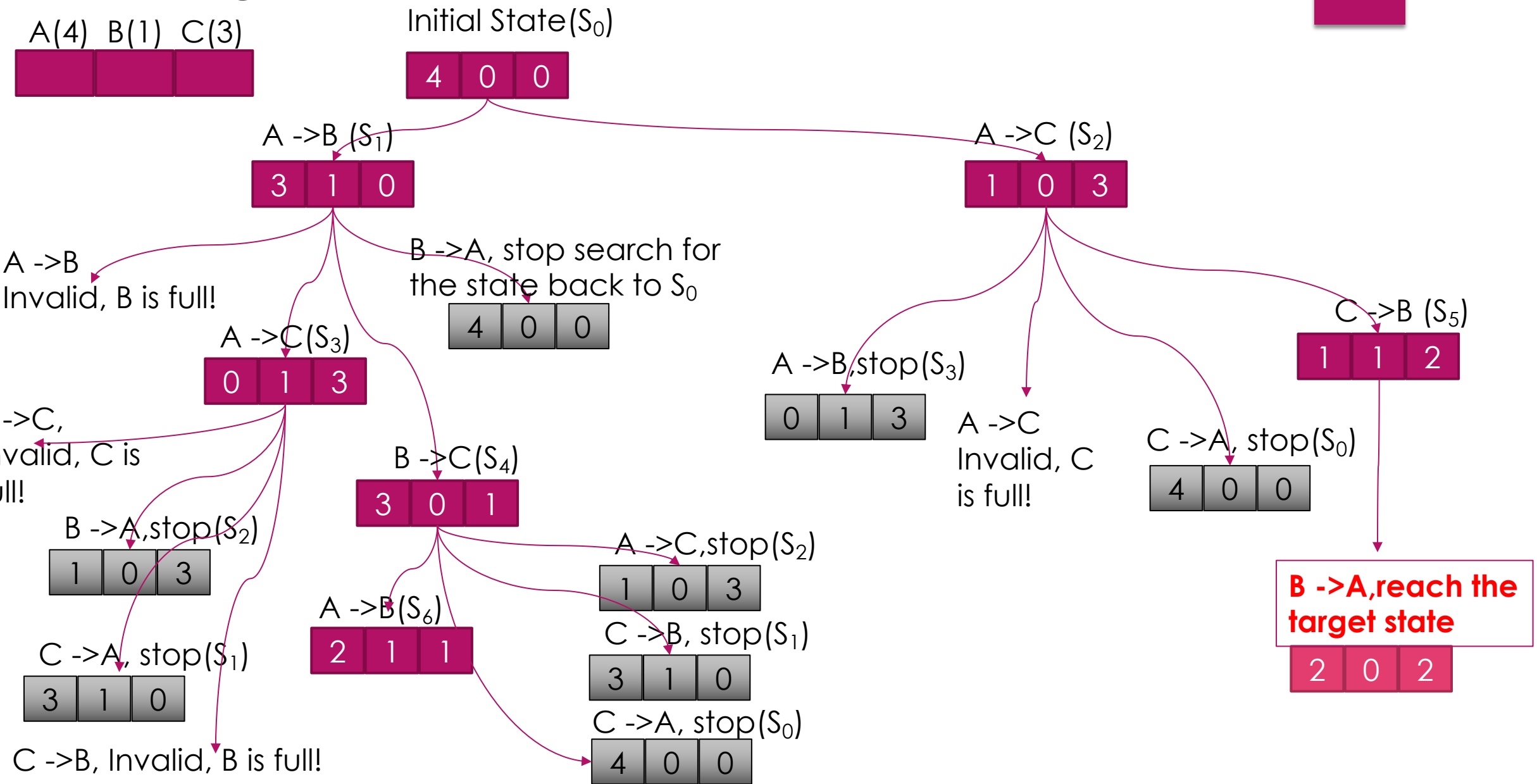
Initial State(S_0): 4 0 0

Target State(S_t): 2 0 2

Dividing water



Dividing water



Dividing water Solution

How to find the shortest distance?

BFS

BFS pseudo-code

```
Boolean BFS(Node S, Node goal){
    queue.init();
    queue.add(S);
    while(!queue.empty()){
        e = queue.remove();
        e.setVisited();
        while(e.hasNextChild){
            child = e.getNextChild();
             if (child.equals(goal)){
                return true;
            }
            if (!child.isVisited()&&!queue.exist(child))
                    queue.add(child);
        }//end while
    }//end while
     return false;
} //end BFS
```