Given a problem **P** with **3** empty beakers **b1, b2 & b3**; with capacities **A, B & C** respectively, the task is to have **t** liters of water in any beaker.

Reduce it to a graph problem.

**● How to construct a graph G?**

1. **What do vertices represent? How many vertices are there?**

Each vertex represents the state of all 3 beakers, i.e. how much water each beaker holds. Let there be a starting point or root vertex **[R]**, where all the 3 beakers are initially empty.

Number of vertices **|V| =** **(A+1) \* (B+1) \* (C+1)** where  **A, B and C** represent the maximum capacities of beakers **b1, b2** & **b3** respectively.

1. **What do edges represent (when would there be an edge from u -> v ?**

An edge from **u** to **v** represents that from one node **u,** one of the following operations can be used to move to another node **v:** **fill\_\_, empty\_\_, pour\_to\_.**

1. **How many edges are there?**

Between **every two vertices,** there exists an edge**.**

This number of edges **|E|** = |**V**|**C2 ;** which is O(V2) or **.**

However, not all edges can be traversed through as we can partially un-fill a beaker until it's empty. But we cannot partially refill it again to its previous quantity.

1. **What is the time necessary to construct the graph (in terms of the problem input size)?**

The time required to create the graph equals the time required to traverse all vertices and connect the edges to all possible neighboring vertices, which is **O(V2)** or **O( (ABC)2).**

* **What problem should be solved on the graph?**

The graph search problem is to be solved with the objective of finding a node where one of the 3 capacities has become **t** i.e. we aim to find one node among a set of nodes where any one of the beakers has **t** liters of water.

Then we have to find the shortest path/sequence to that node from the starting node.

1. **What algorithm should be used?**

**ALGO1: Breadth First Search** is used on root vertex **[R]** as the starting node. We are always concerned with minimizing the number of visited edges while working with unweighted graphs. As a result, we can be confident that every direct neighbor of the source node has a distance of exactly one. The next thing we know for sure is that all of the source node's second neighbors have a distance of two units, and so on until we find a node that satisfies our constraint, i.e., the node must have one beaker containing **t** units of water. BFS will stop when such a node is found.

Let's call it the **[Final]** node. Since BFS traverses level-wise, the path from **[R]** to **[Final]** must have required the fewest operations among all the paths from **[R]** to any qualified node.

**In other words, through BFS we check the reachability of any such [Final] node.**

**When will the algo report that the task is impossible:**

* When **t > maximum (A,B,C)**, then such a node will not exist in the first place as the graph has nodes up to A, B, or C only.
* When **t** cannot be reached through any possible operations (*as defined above*), then that node will remain unvisited in BFS.

**E.g.:** If A, B & C are all even & t is odd, then the task is impossible.

**ALGO2:** Once the **[Final]** node is found, we will run the Single-Source Shortest Path algorithm to get the shortest path/sequence from node **[R]** to node **[Final]**.

**Complexity analysis**

1. **What is the complexity of the above 2 algorithms in terms of G?**

BFS takes **O( V+E )** & *SS Shortest Path takes*  **O( E+VlogV )**

1. **What is the complexity of the overall problem in terms of P ?**

Total time complexity, **T(n)** = time complexity to form the graph

+

time taken to find **[Final]** node

+

time taken to find the shortest path/sequence **[R]** to node **[Final]**

**T(n)** = **O(V2)** + **O(V+E)** [*for BFS*] + **O( E+VlogV )** [*for SS Shortest Path*]

Since, **|E| = O(V2), hence T(n)=O(V2)**

**Space Complexity = number of vertices =**