**HOMEWORK ASSIGNMENT 6**

Problem 1 and 2

Homework Assignment 6

1) BFS Algorithm : Waterjug Problem

* The waterjug object is initially in the state (0,0,0)
* Add this object (the one representing the initial state of the problem) to the queue.
* Do – while the queue is not empty.
* Begin.
* Check if the object at the start of the queue is the required goal.
* If yes,
* Display or print the search path ie; the visited nodes
* Exit.
* Else
* Create all possible valid states of child to current state.
* Add them to the queue.
* Add this current state to the visited nodes array.
* This array stores all the traversed nodes.
* Deque the current object
* End - if
* End.

2) Correctness of the Algorithm -

* Since the Algorithm explores all possible nodes, we will get finite count on the number of nodes.

Problem 2, 3 and 4 ( 4th problem implementation is inside the zipped folder named waterjug.docx)

* The states of the Algorithm will be finite
* Since the States is finite, a graph can be plotted.
* The Application of BFS is to visit all nodes in the graph. Hence, we will get the correct graph.
* In case of impossible state (where the ~~traversed~~ traversal stops abruptly) it is not possible ~~tha~~ to plot the graph.

3) Time Required for the Algorithm

The worst case time Complexity is all the number of nodes visited. If the nodes exists, then time required will be the last node visited

∴ The Time Complexity is $O(c_1 c_2 c_3)$, where $c_1 c_2 c_3$ represents the waterjug limit

4) The Implemented Solution is inside the ~~source~~ main folder named (waterjug.docx).

Problem 5

5.

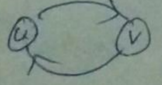
(a). Proof.

Let P be a Euler circuit. It is tested as $v_1, e_1, v_2, e_2 \dots V$. Every time a vertex $V_k$ is listed, two edges $e_{k-1}$ and $e_k$ are listed before & After the vertex. Since the circuit can only visit each edge once, edges are not repeated. Thus each vertex should have even degrees.

A connected graph, with even degree of vertices has Euler tour. (Euler cycle)

Proof by Induction

Base case: Consider graph G with 2 vertices and two edges between them.




Assumption: Assume that all connected graphs with m edges, where each vertex has even degree, has Euler tour.

Proof:- Consider a connected graph G with $K > m$, and each vertex has even degree. We shall start at vertex V and keep following edges arbitarily selecting one after another until we return to V. Consider the trail as W and E be edges of W.

Graph $G-E$ has components $C_1, C_2, C_3 \ldots C_k$ where each component will clearly has less than $m$ edges and every degree is even because when we removed $W$, we removed even edges from each vertex listed in the circuit.

By induction each component has an Eulerian circuit $E_1, E_2 \ldots E_k$. Since $G$ is connected, there is an vertex $a_i$ in each component $C_i$ on both $W$ + $E_i$.

An Euler circuit can be defined in $G$ by starting at $V$ in $W$ until reaching $a_i$, following the edges in $E_k$ to come back to $a_i$ and then adding edges in $W$ to reach a vertex $a_i$ in another component and so on until we reach $V$.

If $G$ is semi-Eulerian, then there is an open Euler trial $v_1$ + ends at $P$ in $G$. Suppose the trail begins at $v_1$ the last listing $v_n$. Except for the first listing of $v_1$ + the last listing of $v_n$, every time a vertex is listed, that accounts for two edges adjacent to that vertex, the one before it in the list and the one after it in the list. This circuit uses every edge exactly once. So every degree must be even, except for $v_1$ + $v_n$ which must be odd.

Suppose u and v are the vertices of odd degree. Consider G + uv. The graph has all even degrees. from the previous proof, G has an Eulerian circuit. The circuit uses the edge uv. Thus we have Euler path in G, when we omit the edge uv. If only one vertex has odd degree there will not be an Eulerian path, since the path for Graph G will not be complete.

(b) Algorithm

Let G be a graph with V vertices and E edges.

1. If graph has any odd vertex, return that Euler path does not exist.
2. Choose any vertex V at random
3. Follow edge one at a time, If there is a choice between a bridge & non bridge, choose the non bridge [Removing single edge from connected graph make it disconnected. such an edge is called bridge]
4. Stop, when we cut the edge

The Running time of the above algorithm is $O(|V| + |E|)$, since it involves travelling each vertex and each edge.