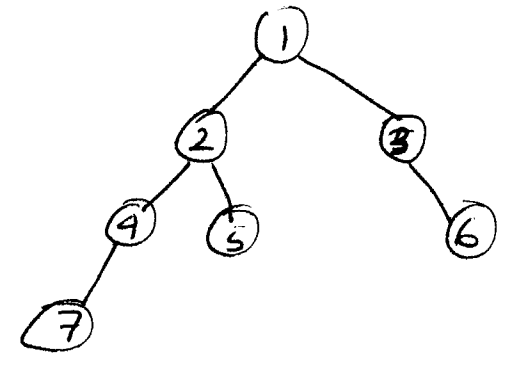


Q1: Stack-print-tree(x)

```

Push(s, x)
while (!stack-empty(s))
  a ← pop(s)
  print key[a]
  if right[a] != NIL
    push(s, right[a])
  if left[a] != NIL
    Push(s, left[a])

```



Output: 1 2 4 7 5 3 6

Q2: Queue-print-tree(x)

```

Enqueue(Q, x)
while (!Queue-empty(Q))
  a ← Dequeue(Q)
  print key[a]
  if left[a] != NIL
    Enqueue(Q, left[a])
  if right[a] != NIL
    Enqueue(Q, right[a])

```

for the same example above

output: 1 2 3 4 5 6 7

Q3: Worst case: When the inputs are sorted, resulting in a tree of single branch. Hence the complexity of insertion is $O(n)$.

Best case: When the binary tree is balanced. Thus an order of $O(\lg(n))$

Q4:

Bipartite \Leftrightarrow no odd cycle

Necessary \Rightarrow

start from v_1 , after odd number of traversals on edges in E , we will only end up in a vertex in V_2 and thus no way we could form a cycle.

\therefore If bipartite \Rightarrow No odd cycles.

Sufficient \Leftarrow

use v_0 in V as a starting point and apply DFS. For Disconnected part, choose unlabeled vertex and apply again. It is easily seen that disconnected part will not affect each other. Therefore we will examine only the connected graph.

Once DFS is applied put all odd number vertices in V_1 and even numbered ones in V_2 . Now we show that $(u,v) \notin E$ if u,v are in the same partition.

Without losing generality, we assume that u,v belongs to V_1 .

So $d[u]$ and $d[v]$ are both odd. If (u,v) belongs to E , then $u \rightarrow u_0 \rightarrow v \rightarrow (u,v) \rightarrow u$ forms a cycle whose length is $d[u] + d[v] + 1$ which is again odd.

\therefore Contradiction.

Hence our assumption that (u,v) is in E is wrong

Thus proved.

Q5:

Algorithm

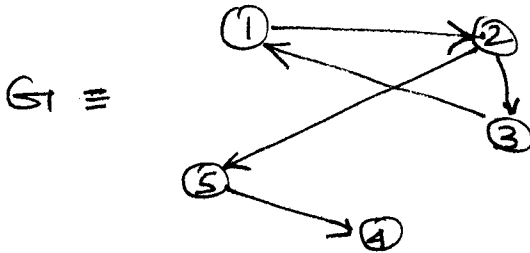
```

for u in V do {
    BFS(G, u);
    for (color(v) == black and u != v) do {
        E' = E' + (u, v);
    }
}

```

Complexity = $O(|V| + |E| + |V| * |V|) * |V| = O(|V| * (|V| + |E|))$

Example:



$G' \equiv$

