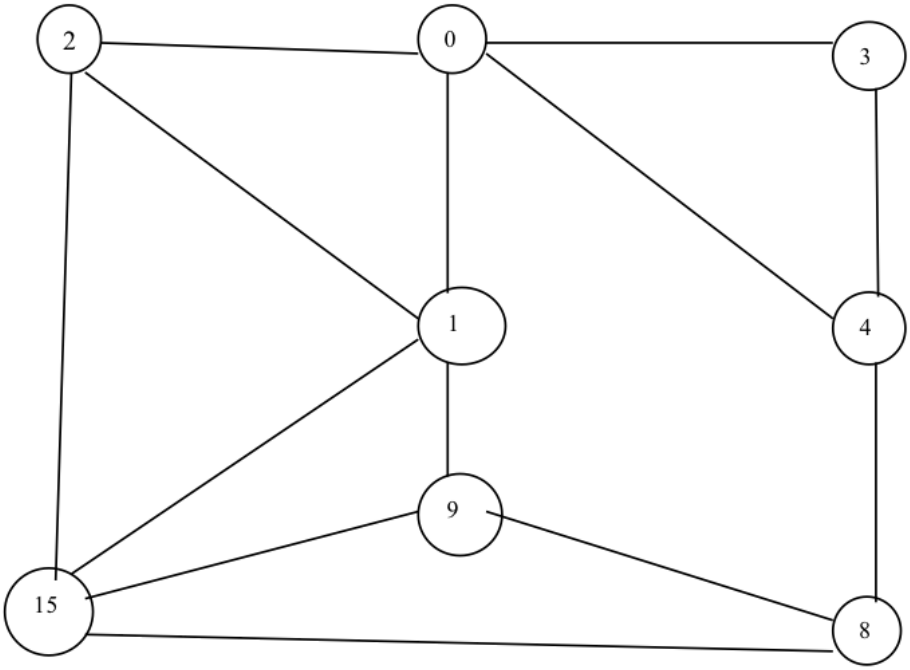
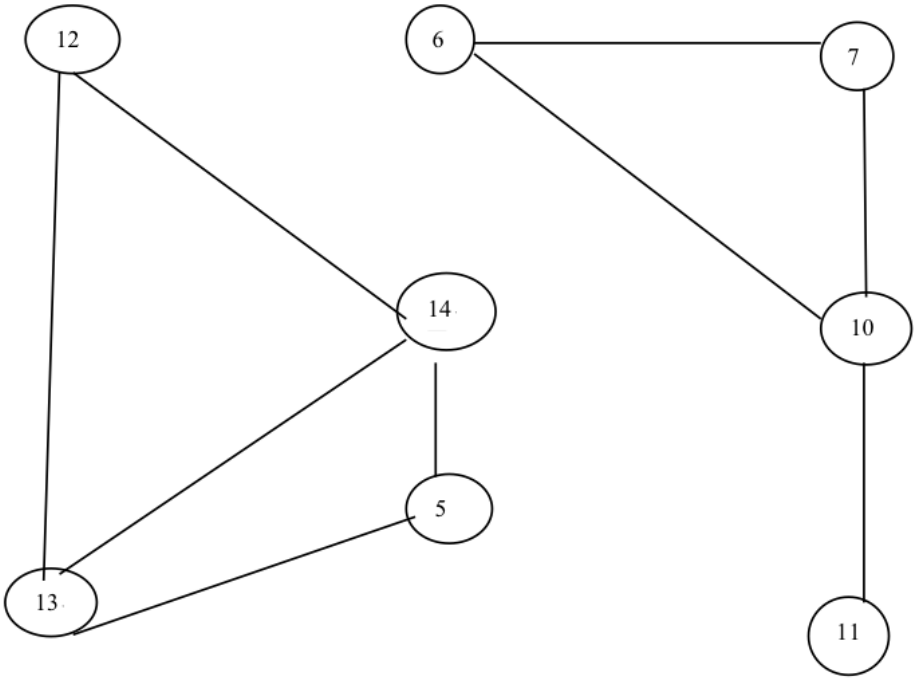
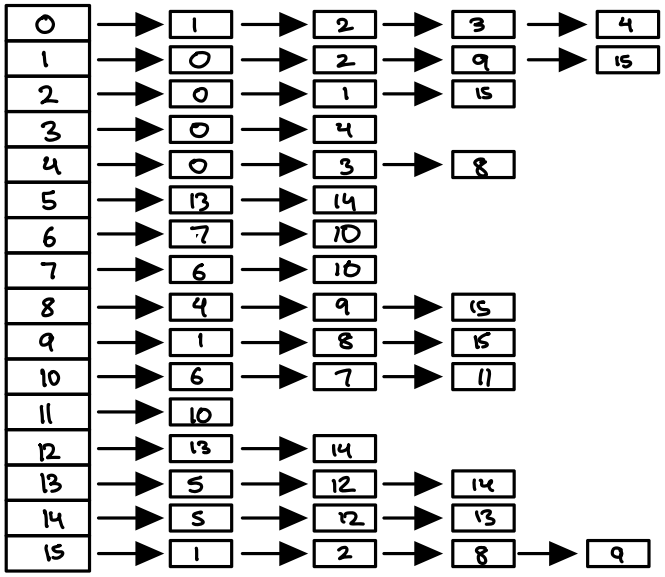
OCS 25 - Data Structures and Algorithms
Programming Assignment - 2
Undirected graphs

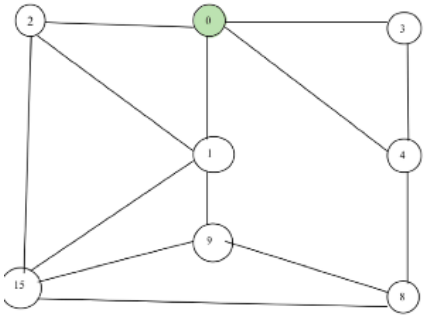
1) Draw, in the style of the figure in the text (page 524), the adjacency lists built by Graph's input stream constructor for the graph presented above.



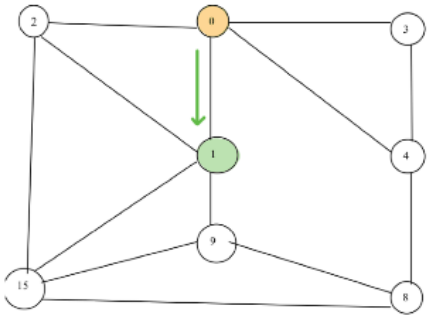




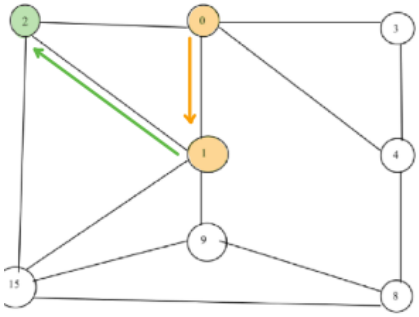
2) Show, in the style of the figure on page 533, a detailed trace of the call dfs(0) for the above graphs. Also, draw the tree represented by edgeTo[].



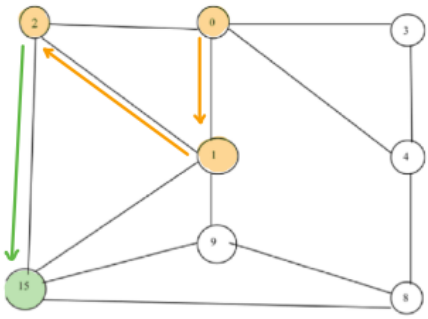
	Marked	Adjacent	EdgeTo
0	T	1,2,3,4	null
1		0,2,9,15	
2		0,1,15	
3		0,4	
4		3,8	
8		4,9,15	
9		1,8,15	
15		1,2,8,9	



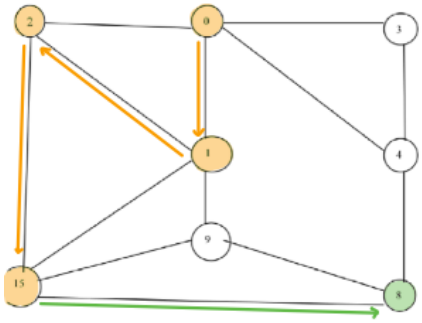
	Marked	Adjacent	EdgeTo
0	T	~~1~~,2,3,4	null
1	T	0,2,9,15	0
2		0,1,15	
3		0,4	
4		3,8	
8		4,9,15	
9		1,8,15	
15		1,2,8,9	



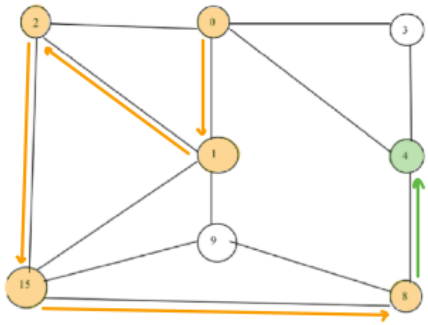
	Marked	Adjacent	EdgeTo
0	T	~~1~~,2,3,4	null
1	T	~~0~~,2,9,15	0
2	T	0,1,15	1
3		0,4	
4		3,8	
8		4,9,15	
9		1,8,15	
15		1,2,8,9	



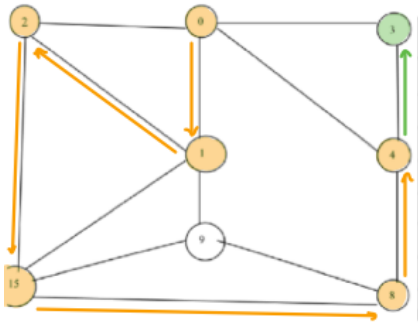
	Marked	Adjacent	EdgeTo
0	T	~~1~~,2,3,4	null
1	T	~~0~~,2,9,15	0
2	T	~~0~~,~~1~~,15	1
3		0,4	
4		3,8	
8		4,9,15	
9		1,8,15	
15	T	1,2,8,9	2



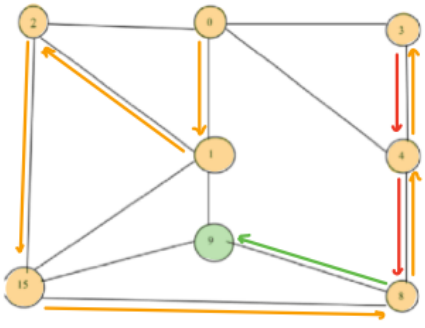
	Marked	Adjacent	EdgeTo
0	T	~~1~~,2,3,4	null
1	T	~~0~~,2,9,15	0
2	T	~~0~~,~~1~~,15	1
3		0,4	
4		3,8	
8	T	4,9,15	15
9		1,8,15	
15	T	~~1~~,2,~~8~~,9	2



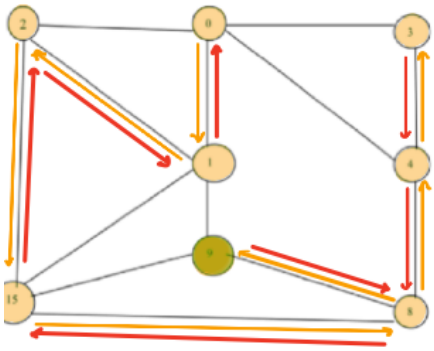
	Marked	Adjacent	EdgeTo
0	T	~~1~~,2,3,4	null
1	T	~~0~~,2,9,15	0
2	T	~~0~~,~~1~~,15	1
3		0,4	
4	T	3,8	8
8	T	~~4~~,9,15	15
9		1,8,15	
15	T	~~1~~,~~2~~,~~8~~,9	2



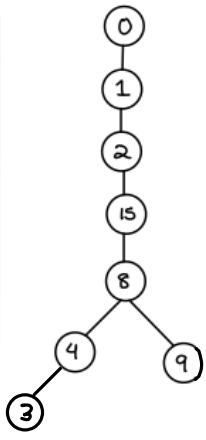
	Marked	Adjacent	EdgeTo
0	T	~~1~~,2,3,4	null
1	T	~~0~~,2,9,15	0
2	T	~~0~~,~~1~~,15	1
3	T	0,4	4
4	T	~~3~~,8	8
8	T	~~4~~,9,15	15
9		1,8,15	
15	T	~~1~~,2,8,9	2



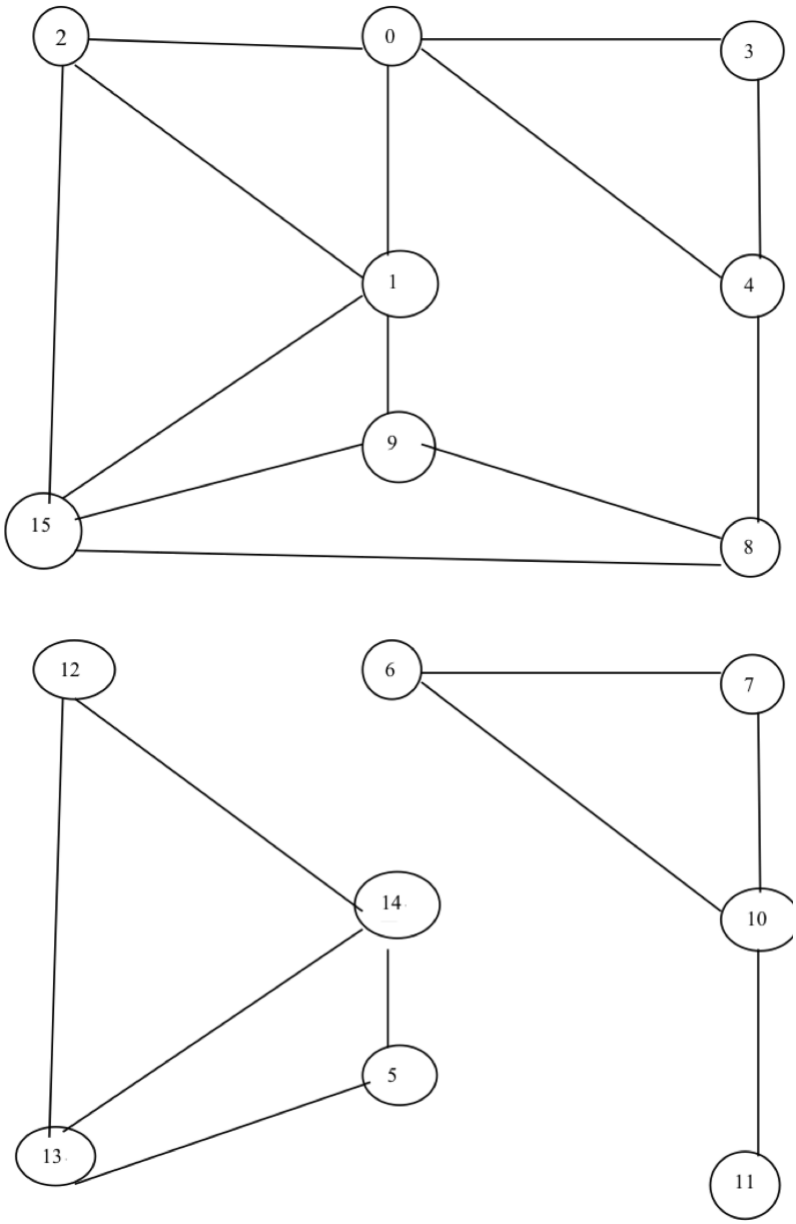
	Marked	Adjacent	EdgeTo
0	T	~~1~~,2,3,4	null
1	T	~~0~~,2,9,15	0
2	T	~~0~~,~~1~~,15	1
3	T	~~0~~,4	4
4	T	~~3~~,~~8~~	8
8	T	~~4~~,~~9~~,15	15
9	T	1,8,15	8
15	T	~~1~~,2,8,9	2

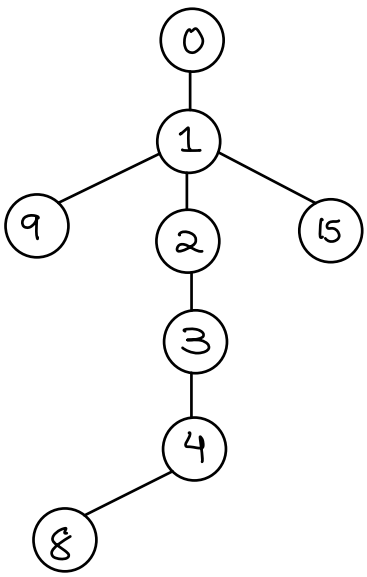


	Marked	Adjacent	EdgeTo
0	T	~~1~~,2,3,4	null
1	T	~~0~~,2,9,15	0
2	T	~~0~~,~~1~~,15	1
3	T	~~0~~,4	4
4	T	~~3~~,8	8
8	T	~~4~~,~~9~~,15	15
9	T	~~1~~,8,15	8
15	T	~~1~~,2,8,9	2

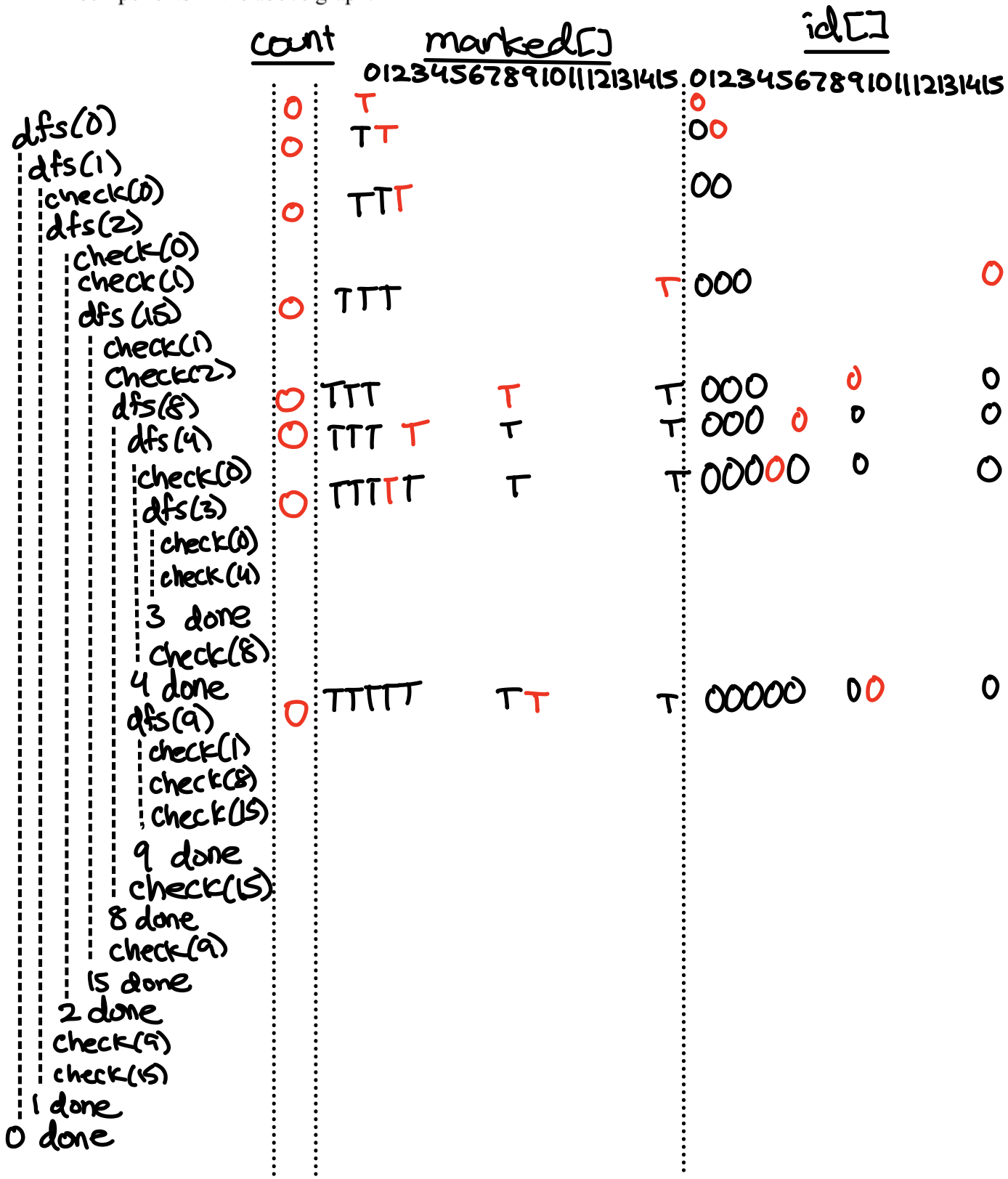


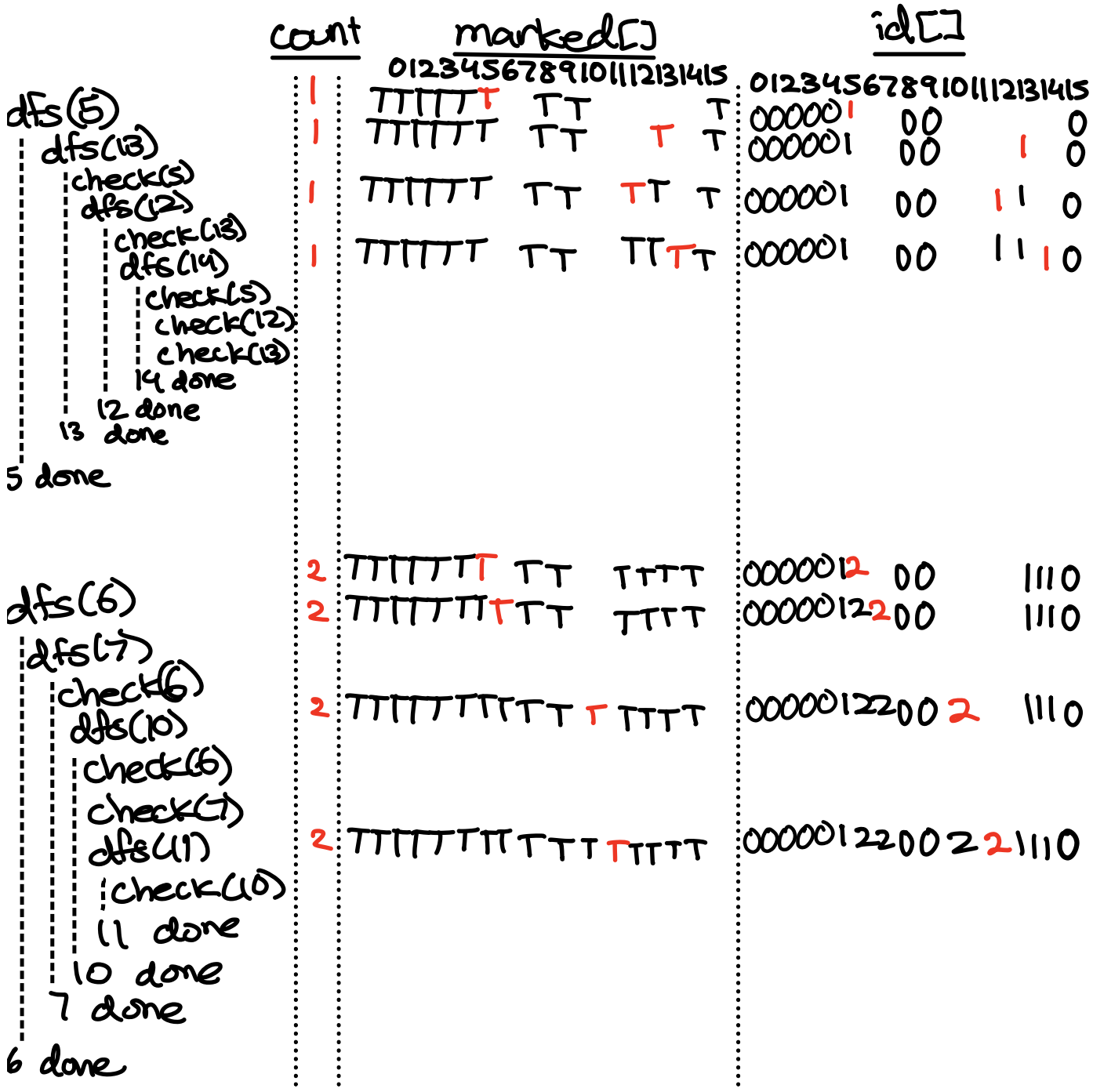
3) Draw the tree represented by edgeTo[] after the call bfs(G, 0) in Algorithm 4.2 for the above graph.





4) Show, in the style of the figure on page 545, a detailed trace of CC for finding the connected components in the above graph.




count
marked[]
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
id[]
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
dfs(5)
dfs(13)
check(5)
dfs(12)
check(13)
dfs(14)
check(5)
check(12)
check(13)
14 done
12 done
13 done
5 done
dfs(6)
dfs(7)
check(6)
dfs(10)
check(6)
check(7)
dfs(11)
check(10)
11 done
10 done
7 done
6 done

5) Show, in the style of the figures in this section, a detailed trace of Cycle for finding a cycle in the above graph.

```
public class Cycle
{
  private boolean hasCycle;
  public boolean hasCycle()
  {
     return hasCycle;
  }
  public static void dfs(Graph G, int v, int u)
  {
    marked[v]=true;
    for(int w: G.adj(v))
    {
      if (!marked(w))
      {
        dfs(G,w,v);
      }
      else if (w!=u)
      {
        hasCycle=true;
      }
    }
  }
}
```

dfs(G, 0, 15)
↳ w=1, v=15
!marked(w) = true
↳ dfs(G, 1, 0)
↳ w=0, v=0
!marked(w) = false;
w != v = false;

↳ w=2, v=0
!marked(w) = true
↳ dfs(G, 2, 1)
↳ w=0, v=1
!marked(w) = false
w != v = true
↳ hasCycle = true;