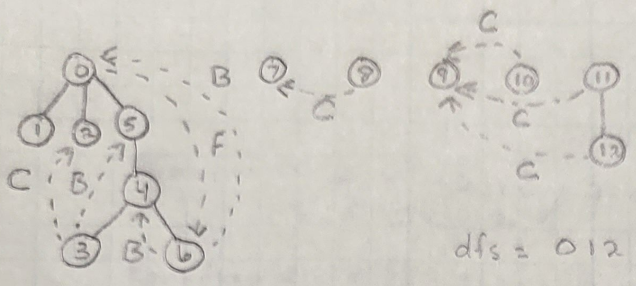
Data Structures : Lab 11

Edwin (Trey) Sparks
11-3-21

1)



DFS Traversal Tree

dfs = 0 1 2 5 4 3 6 7 8 9 10 11 12 ✓

Cross Traversal
(F - forward, B - Backward, C - Cross)

2) Coding Screen Shot @ the End. ◡̈

3) dfs = 0 1 2 5 4 3 6 7 8 9 10 11 12 ✓

Notice, Vertex 12 → mark 13.
This occures because of the counter attached to mark.

4) Coding Screen Shot @ the End. ◡̈

5) $A = \overset{L}{\begin{bmatrix} 1 & 0 & 0 \\ x & 1 & 0 \\ x & x & 1 \end{bmatrix}} \overset{u}{\begin{bmatrix} x & x & x \\ 0 & x & x \\ 0 & 0 & x \end{bmatrix}}$ $\Rightarrow$ $\begin{matrix} R_2 - \frac{5}{2}R_1 \\ R_3 - 2R_1 \end{matrix} \begin{bmatrix} 2 & 2 & 3 \\ 0 & 4 & 5/2 \\ 0 & \underbrace{-3 \; -4}_{} \end{bmatrix}$ $\begin{matrix} k = 5/2 \\ k = 2 \end{matrix}$

$10 - \frac{5}{2}(3)$
$= \frac{20}{2} - \frac{15}{2}$
$= \frac{5}{2}$

Replace Ri by Ri + kRj

L: Records k Values;

$R_3 + \frac{3}{4}R_2 = \begin{bmatrix} 2 & 2 & 3 \\ 0 & 4 & 5/2 \\ 0 & 0 & -17/8 \end{bmatrix} \Rightarrow u$ $k = -3/4$

$-4 + \frac{3}{4}(\frac{5}{2})$
$= -\frac{32}{8} + \frac{15}{8}$
$= -\frac{17}{8}$

$\begin{bmatrix} 1 & 0 & 0 \\ 5/2 & 1 & 0 \\ 2 & -3/4 & 1 \end{bmatrix} \Rightarrow L$ $A = LU = \begin{bmatrix} 1 & 0 & 0 \\ 5/2 & 1 & 0 \\ 2 & -3/4 & 1 \end{bmatrix} \begin{bmatrix} 2 & 2 & 3 \\ 0 & 4 & 5/2 \\ 0 & -3 & -4 \end{bmatrix}$

6)

y

Equation 1: $1 \cdot y_1 + 0 \cdot y_2 + 0 \cdot y_3 = 2$
$\Rightarrow$ $y_1 = 2$

Equation 2: $\frac{5}{2} \cdot y_1 + 1 \cdot y_2 + 0 \cdot y_3 = 4$
$\Rightarrow$ $\frac{5}{2} y_1 + y_2 = 4$
$\Rightarrow$ $5 + y_2 = 4$
$\Rightarrow$ $y_2 = -1$

Equation 3: $2 \cdot y_1 + -\frac{3}{4} y_2 + 1 \cdot y_3 = 3$
$\Rightarrow$ $2y_1 - \frac{3}{4} y_2 + y_3 = 3$
$\Rightarrow$ $4 + \frac{3}{4} + y_3 = 3$
$\Rightarrow$ $y_3 = -\frac{7}{4}$

x

Equation 3: $-\frac{17}{8} x_3 = -\frac{7}{4}$
$\Rightarrow$ $x_3 = 0.8235$ or $\frac{14}{17}$

Equation 2: $4x_2 + \frac{5}{2} \cdot 0.8235 = -1$
$\Rightarrow$ $x_2 = -0.7647$ or $-\frac{13}{17}$

Equation 1: $2x_1 + 2 \cdot -0.7647 + 3 \cdot 0.82 = -\frac{7}{4}$
$\Rightarrow$ $x_3 = 0.5294$ or $\frac{9}{17}$

7) Check

Equation 1: $2 \cdot x_1 + 2 \cdot x_2 + 3 \cdot x_3 = 2$

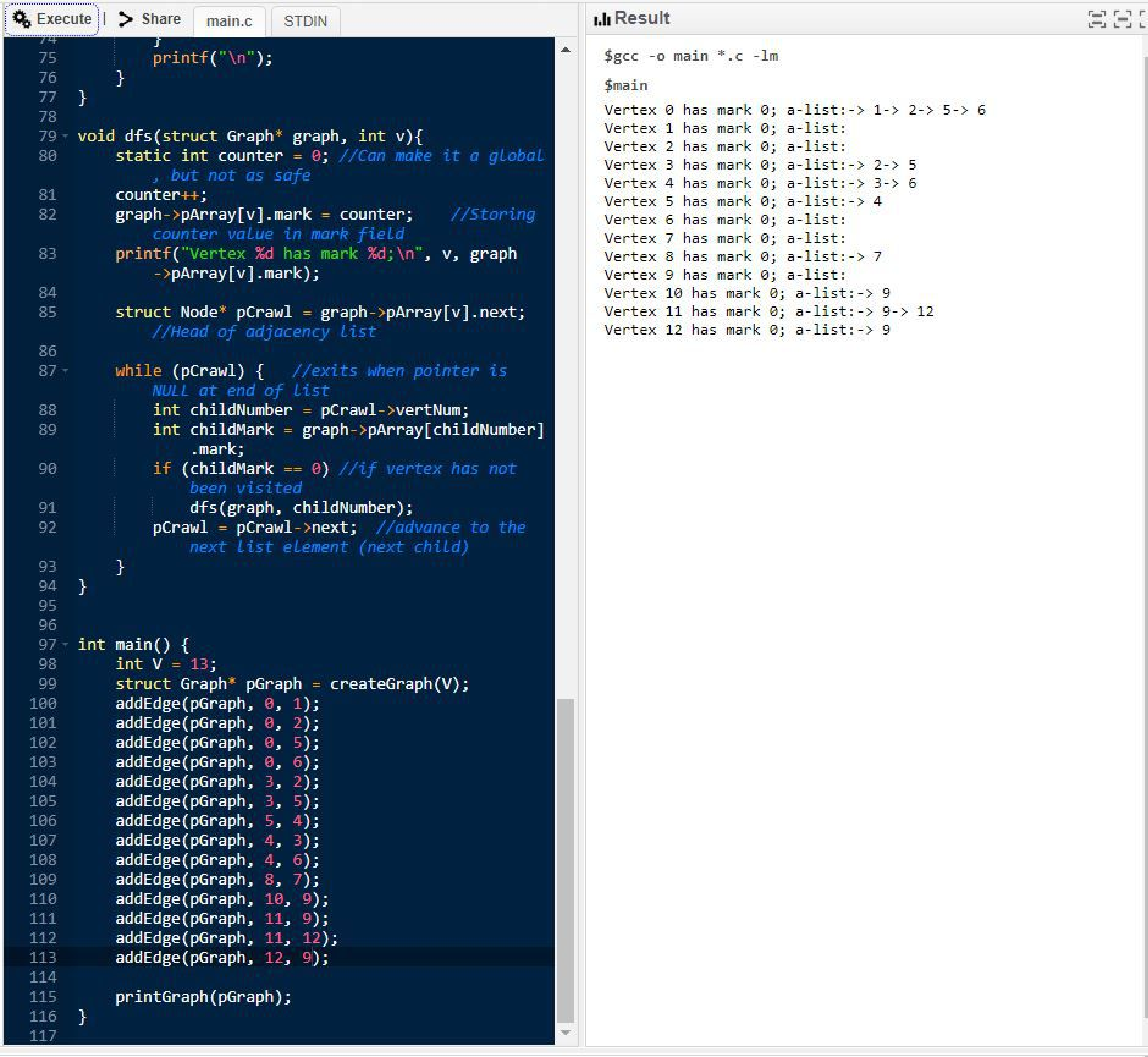
$\Rightarrow 34/17 = 2$ ✓

Equation 2: $5 \cdot x_1 + 9 \cdot x_2 + 10 \cdot x_3 = 4$

$\Rightarrow 68/17 = 4$ ✓

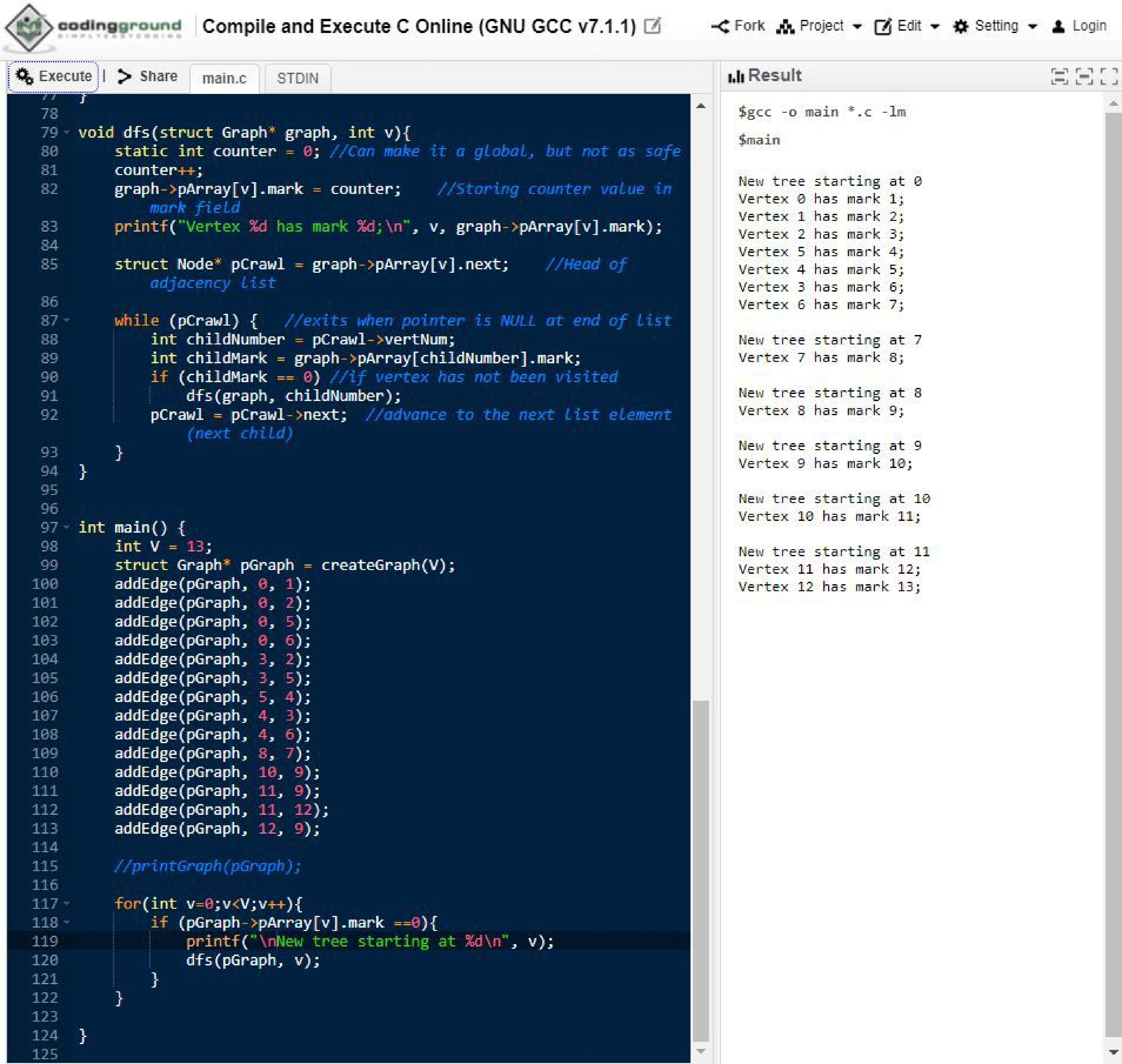
Equation 3: $4 \cdot x_1 + x_2 + 2x_3 = 3$

$\Rightarrow {}^{51}/_{17} = 3$ ✓



```c
        }
        printf("\n");
    }
}

void dfs(struct Graph* graph, int v){
    static int counter = 0; //Can make it a global, but not as safe
    counter++;
    graph->pArray[v].mark = counter;    //Storing counter value in mark field
    printf("Vertex %d has mark %d;\n", v, graph->pArray[v].mark);

    struct Node* pCrawl = graph->pArray[v].next; //Head of adjacency list

    while (pCrawl) {   //exits when pointer is NULL at end of list
        int childNumber = pCrawl->vertNum;
        int childMark = graph->pArray[childNumber].mark;
        if (childMark == 0) //if vertex has not been visited
            dfs(graph, childNumber);
        pCrawl = pCrawl->next;  //advance to the next list element (next child)
    }
}


int main() {
    int V = 13;
    struct Graph* pGraph = createGraph(V);
    addEdge(pGraph, 0, 1);
    addEdge(pGraph, 0, 2);
    addEdge(pGraph, 0, 5);
    addEdge(pGraph, 0, 6);
    addEdge(pGraph, 3, 2);
    addEdge(pGraph, 3, 5);
    addEdge(pGraph, 5, 4);
    addEdge(pGraph, 4, 3);
    addEdge(pGraph, 4, 6);
    addEdge(pGraph, 8, 7);
    addEdge(pGraph, 10, 9);
    addEdge(pGraph, 11, 9);
    addEdge(pGraph, 11, 12);
    addEdge(pGraph, 12, 9);

    printGraph(pGraph);
}

```

```
$gcc -o main *.c -lm

$main
Vertex 0 has mark 0; a-list:-> 1-> 2-> 5-> 6
Vertex 1 has mark 0; a-list:
Vertex 2 has mark 0; a-list:
Vertex 3 has mark 0; a-list:-> 2-> 5
Vertex 4 has mark 0; a-list:-> 3-> 6
Vertex 5 has mark 0; a-list:-> 4
Vertex 6 has mark 0; a-list:
Vertex 7 has mark 0; a-list:
Vertex 8 has mark 0; a-list:-> 7
Vertex 9 has mark 0; a-list:
Vertex 10 has mark 0; a-list:-> 9
Vertex 11 has mark 0; a-list:-> 9-> 12
Vertex 12 has mark 0; a-list:-> 9
```



```c
}

void dfs(struct Graph* graph, int v){
    static int counter = 0; //Can make it a global, but not as safe
    counter++;
    graph->pArray[v].mark = counter;    //Storing counter value in mark field
    printf("Vertex %d has mark %d;\n", v, graph->pArray[v].mark);

    struct Node* pCrawl = graph->pArray[v].next;    //Head of adjacency list

    while (pCrawl) {   //exits when pointer is NULL at end of list
        int childNumber = pCrawl->vertNum;
        int childMark = graph->pArray[childNumber].mark;
        if (childMark == 0) //if vertex has not been visited
            dfs(graph, childNumber);
        pCrawl = pCrawl->next;  //advance to the next list element (next child)
    }
}


int main() {
    int V = 13;
    struct Graph* pGraph = createGraph(V);
    addEdge(pGraph, 0, 1);
    addEdge(pGraph, 0, 2);
    addEdge(pGraph, 0, 5);
    addEdge(pGraph, 0, 6);
    addEdge(pGraph, 3, 2);
    addEdge(pGraph, 3, 5);
    addEdge(pGraph, 5, 4);
    addEdge(pGraph, 4, 3);
    addEdge(pGraph, 4, 6);
    addEdge(pGraph, 8, 7);
    addEdge(pGraph, 10, 9);
    addEdge(pGraph, 11, 9);
    addEdge(pGraph, 11, 12);
    addEdge(pGraph, 12, 9);

    //printGraph(pGraph);

    for(int v=0;v<V;v++){
        if (pGraph->pArray[v].mark ==0){
            printf("\nNew tree starting at %d\n", v);
            dfs(pGraph, v);
        }
    }

}
```

```
$gcc -o main *.c -lm

$main

New tree starting at 0
Vertex 0 has mark 1;
Vertex 1 has mark 2;
Vertex 2 has mark 3;
Vertex 5 has mark 4;
Vertex 4 has mark 5;
Vertex 3 has mark 6;
Vertex 6 has mark 7;

New tree starting at 7
Vertex 7 has mark 8;

New tree starting at 8
Vertex 8 has mark 9;

New tree starting at 9
Vertex 9 has mark 10;

New tree starting at 10
Vertex 10 has mark 11;

New tree starting at 11
Vertex 11 has mark 12;
Vertex 12 has mark 13;
```