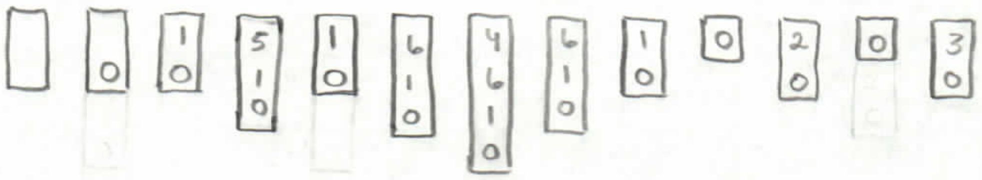
Data Structures Lab 09:

Edwin (Tony) Sparks
10-20-21

1) - Screen shot at the end.

2) Topological Sort: (DFS)



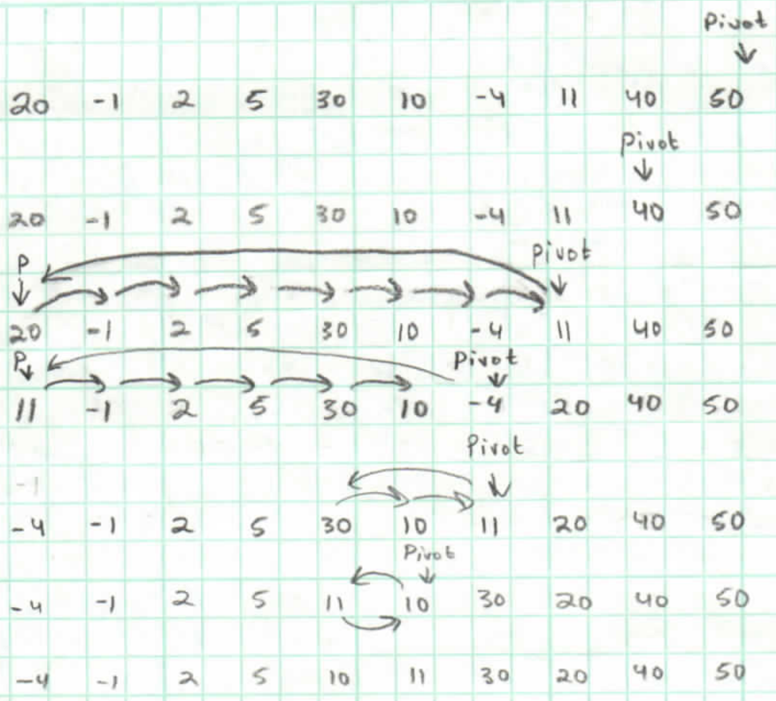
$$\underbrace{\frac{1}{5}\ \frac{2}{4}\ \frac{3}{6}\ \frac{4}{1}\ \frac{5}{2}\ \frac{6}{3}}_{\text{PoP order ☺}}\ \frac{7}{0}$$ = 0156423 DFS Traversal

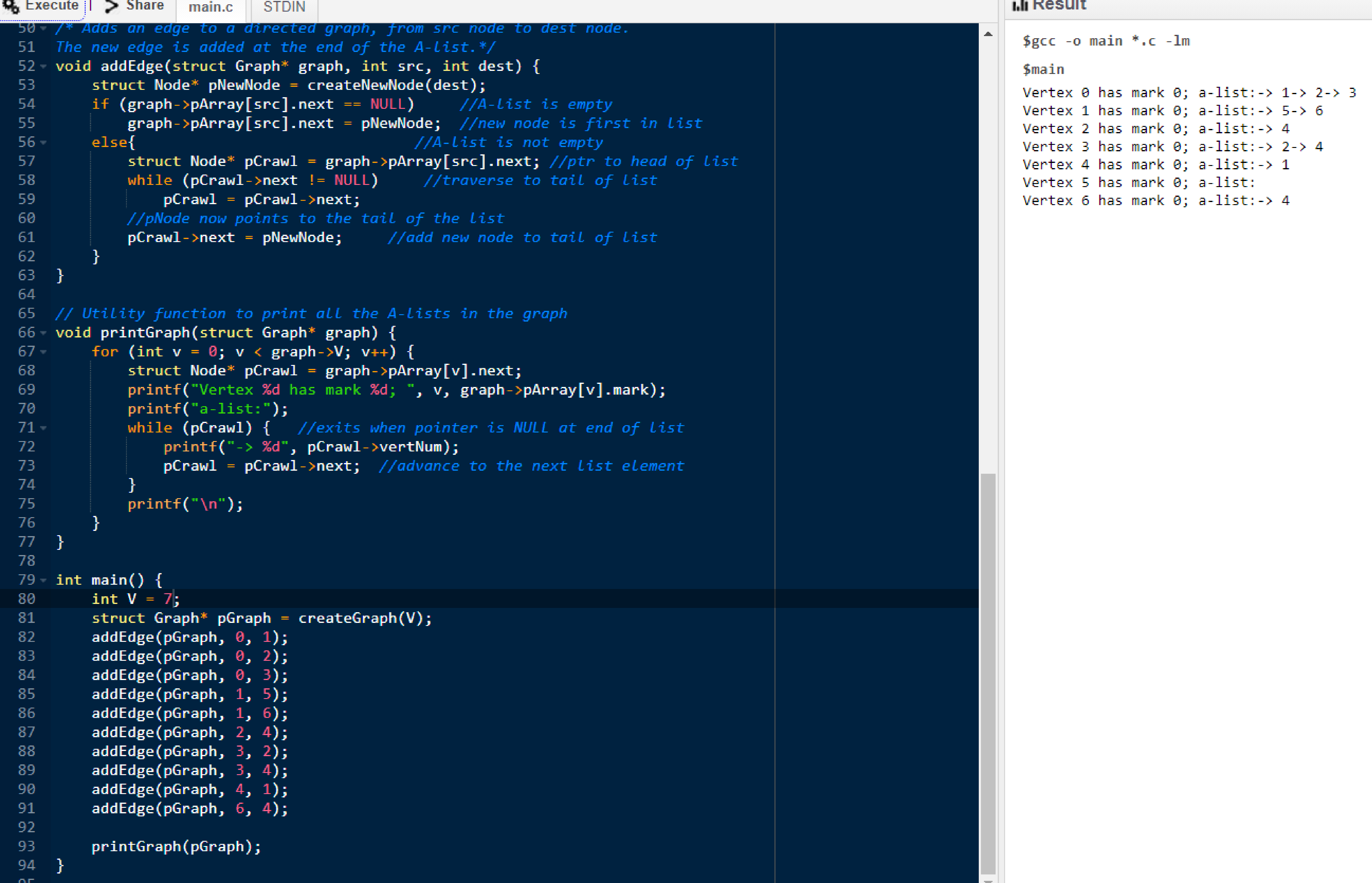
3) - Screen Shot at the end.

4)

Lomuto Partition
$\frac{1}{n}\sum_{p=1}^{n}(p-1) = (\frac{n}{2}-\frac{1}{2})$
$\Theta(n^2)$

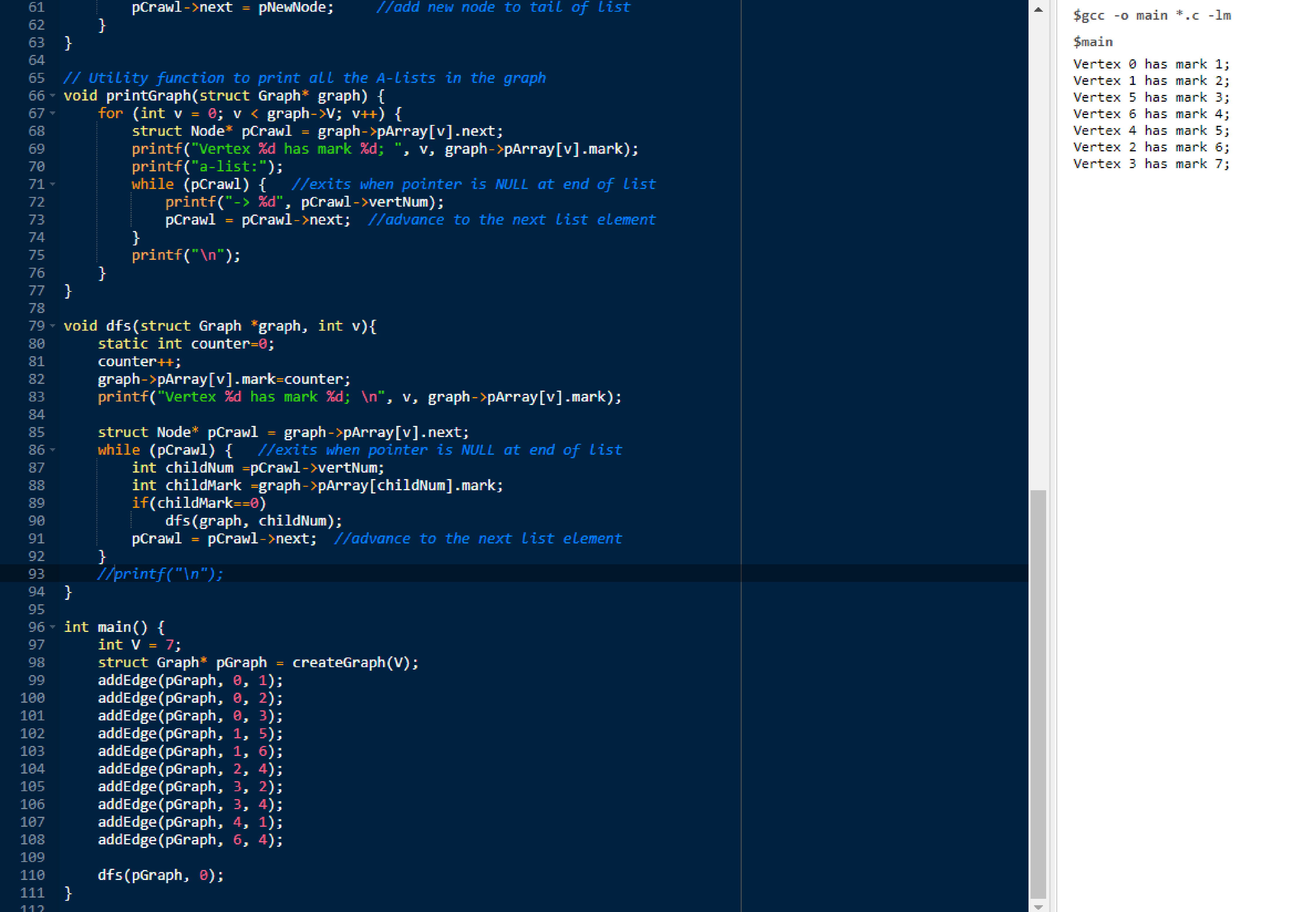
```
    0    1    2    3    4    5    6    7    9    9
   20   -1    2    5   30   10   -4   11   40   50
  (s)  (i)                ↶
       (s)  (i)
            (s)  (i)
                 (s)  (i)  (i)
                      (s)  ←
   20   -1    2    5   10   30   -4   11   40   50
                      (s)  (i)

                                (s)  (i)
   20   -1    2    5   10   -4   11   30   40   50
                                     (s)  (i)       ← Reset
   20 → -1    2    5   10   -4   11   30   40   50
  (s)  (i)
   -1   20 →  2    5   10   -4   11   30   40   50
       (s)  (i)
   -1    2    5   20 ← 10   -4   11   30   40   50
                 (s)  (i)
   -1    2    5   10   20   -4   11   30   40   50
```


Pivot
20 -1 2 5 30 10 -4 11 40 50
Pivot
20 -1 2 5 30 10 -4 11 40 50
P
Pivot
20 -1 2 5 30 10 -4 11 40 50
P
Pivot
11 -1 2 5 30 10 -4 20 40 50
Pivot
-1
-4 -1 2 5 30 10 11 20 40 50
Pivot
-4 -1 2 5 11 10 30 20 40 50
-4 -1 2 5 10 11 30 20 40 50



```c
/* Adds an edge to a directed graph, from src node to dest node.
The new edge is added at the end of the A-list.*/
void addEdge(struct Graph* graph, int src, int dest) {
    struct Node* pNewNode = createNewNode(dest);
    if (graph->pArray[src].next == NULL)     //A-list is empty
        graph->pArray[src].next = pNewNode;  //new node is first in list
    else{                               //A-list is not empty
        struct Node* pCrawl = graph->pArray[src].next; //ptr to head of list
        while (pCrawl->next != NULL)     //traverse to tail of list
            pCrawl = pCrawl->next;
        //pNode now points to the tail of the list
        pCrawl->next = pNewNode;     //add new node to tail of list
    }
}

// Utility function to print all the A-lists in the graph
void printGraph(struct Graph* graph) {
    for (int v = 0; v < graph->V; v++) {
        struct Node* pCrawl = graph->pArray[v].next;
        printf("Vertex %d has mark %d; ", v, graph->pArray[v].mark);
        printf("a-list:");
        while (pCrawl) {   //exits when pointer is NULL at end of list
            printf("-> %d", pCrawl->vertNum);
            pCrawl = pCrawl->next;  //advance to the next list element
        }
        printf("\n");
    }
}

int main() {
    int V = 7;
    struct Graph* pGraph = createGraph(V);
    addEdge(pGraph, 0, 1);
    addEdge(pGraph, 0, 2);
    addEdge(pGraph, 0, 3);
    addEdge(pGraph, 1, 5);
    addEdge(pGraph, 1, 6);
    addEdge(pGraph, 2, 4);
    addEdge(pGraph, 3, 2);
    addEdge(pGraph, 3, 4);
    addEdge(pGraph, 4, 1);
    addEdge(pGraph, 6, 4);

    printGraph(pGraph);
}
```

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

$main
Vertex 0 has mark 0; a-list:-> 1-> 2-> 3
Vertex 1 has mark 0; a-list:-> 5-> 6
Vertex 2 has mark 0; a-list:-> 4
Vertex 3 has mark 0; a-list:-> 2-> 4
Vertex 4 has mark 0; a-list:-> 1
Vertex 5 has mark 0; a-list:
Vertex 6 has mark 0; a-list:-> 4
```



```c
        pCrawl->next = pNewNode;     //add new node to tail of list
    }
}

// Utility function to print all the A-lists in the graph
void printGraph(struct Graph* graph) {
    for (int v = 0; v < graph->V; v++) {
        struct Node* pCrawl = graph->pArray[v].next;
        printf("Vertex %d has mark %d; ", v, graph->pArray[v].mark);
        printf("a-list:");
        while (pCrawl) {   //exits when pointer is NULL at end of list
            printf("-> %d", pCrawl->vertNum);
            pCrawl = pCrawl->next;  //advance to the next list element
        }
        printf("\n");
    }
}

void dfs(struct Graph *graph, int v){
    static int counter=0;
    counter++;
    graph->pArray[v].mark=counter;
    printf("Vertex %d has mark %d; \n", v, graph->pArray[v].mark);

    struct Node* pCrawl = graph->pArray[v].next;
    while (pCrawl) {   //exits when pointer is NULL at end of list
        int childNum =pCrawl->vertNum;
        int childMark =graph->pArray[childNum].mark;
        if(childMark==0)
            dfs(graph, childNum);
        pCrawl = pCrawl->next;  //advance to the next list element
    }
    //printf("\n");
}

int main() {
    int V = 7;
    struct Graph* pGraph = createGraph(V);
    addEdge(pGraph, 0, 1);
    addEdge(pGraph, 0, 2);
    addEdge(pGraph, 0, 3);
    addEdge(pGraph, 1, 5);
    addEdge(pGraph, 1, 6);
    addEdge(pGraph, 2, 4);
    addEdge(pGraph, 3, 2);
    addEdge(pGraph, 3, 4);
    addEdge(pGraph, 4, 1);
    addEdge(pGraph, 6, 4);

    dfs(pGraph, 0);
}
```

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

$main
Vertex 0 has mark 1;
Vertex 1 has mark 2;
Vertex 5 has mark 3;
Vertex 6 has mark 4;
Vertex 4 has mark 5;
Vertex 2 has mark 6;
Vertex 3 has mark 7;
```