

Lab Experiment 13

Implementation of DAG

Harshit Aggarwal

RA1911003010782

D2

Date: 12.4.22

AIM: Write a program for Implementation of DAG

Algorithm

for construction of Directed Acyclic Graph:

There are three possible scenarios for building a DAG on three address codes:

Case 1 – $x = y \text{ op } z$

Case 2 – $x = \text{op } y$

Case 3 – $x = y$

Directed Acyclic Graph for the above cases can be built as follows :

Step 1 –

If the y operand is not defined, then create a node (y).

If the z operand is not defined, create a node for case(1) as node(z).

Step 2 –

Create node(OP) for case(1), with node(z) as its right child and node(OP) as its left child (y).

For case (2), see if there is a node operator (OP) with one child node (y).

Node n will be node(y) in case (3).

Step 3 –

Remove x from the list of node identifiers. Step 2: Add x to the list of attached identifiers for node n.

CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define MIN_PER_RANK 1
#define MAX_PER_RANK 5
#define MIN_RANKS 3
#define MAX_RANKS 5
#define PERCENT 30
void main()
{
int i,j,k,nodes=0;
srand(time(NULL));
int ranks=MIN_RANKS+(rand()%(MAX_RANKS-
MIN_RANKS+1));
printf("DIRECTED ACYCLIC GRAPH\n");
for(i=1;i<ranks;i++)
{
```

```

int new_nodes=MIN_PER_RANK+(rand()%(MAX_PER_RANK-
MIN_PER_RANK+1));

for(j=0;j<nodes;j++)
for(k=0;k<new_nodes;k++)
if((rand()%100)<PERCENT)
printf("%d->%d;\n",j,k+nodes);

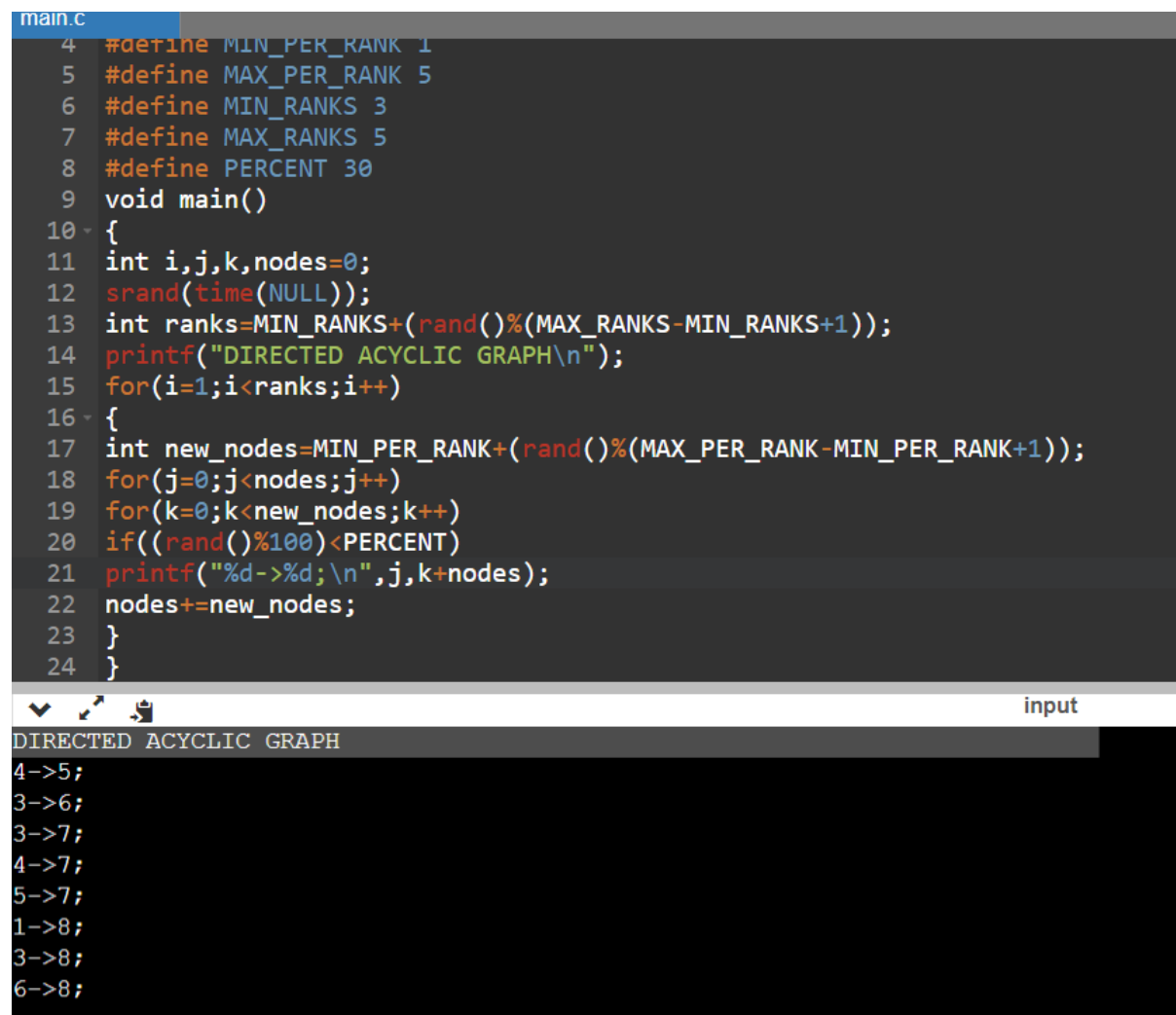
nodes+=new_nodes;

}

}

```

OUTPUT:



The screenshot shows a C program named 'main.c' being executed. The code defines constants for ranks and percentages, and generates a directed acyclic graph by randomly adding nodes and edges. The output displays the title 'DIRECTED ACYCLIC GRAPH' followed by eight edges: 4->5, 3->6, 3->7, 4->7, 5->7, 1->8, 3->8, and 6->8.

```

main.c
4  #define MIN_PER_RANK 1
5  #define MAX_PER_RANK 5
6  #define MIN_RANKS 3
7  #define MAX_RANKS 5
8  #define PERCENT 30
9  void main()
10 {
11  int i,j,k,nodes=0;
12  srand(time(NULL));
13  int ranks=MIN_RANKS+(rand()%(MAX_RANKS-MIN_RANKS+1));
14  printf("DIRECTED ACYCLIC GRAPH\n");
15  for(i=1;i<ranks;i++)
16  {
17  int new_nodes=MIN_PER_RANK+(rand()%(MAX_PER_RANK-MIN_PER_RANK+1));
18  for(j=0;j<nodes;j++)
19  for(k=0;k<new_nodes;k++)
20  if((rand()%100)<PERCENT)
21  printf("%d->%d;\n",j,k+nodes);
22  nodes+=new_nodes;
23  }
24  }

```

input

```

DIRECTED ACYCLIC GRAPH
4->5;
3->6;
3->7;
4->7;
5->7;
1->8;
3->8;
6->8;

```

RESULT: Successful implementation of DAG.