DHRUV GARG 16 BCE 1190

OPENMP – SCHEDULING

LAB 3

Aim: To understand and implement the three scheduling algorithms (static, dynamic and guided) for a shortest path algorithm.

CODE AND OUTPUTS

1. Dijkstra's shortest path algorithm– CODE WITH OpenMP scheduling

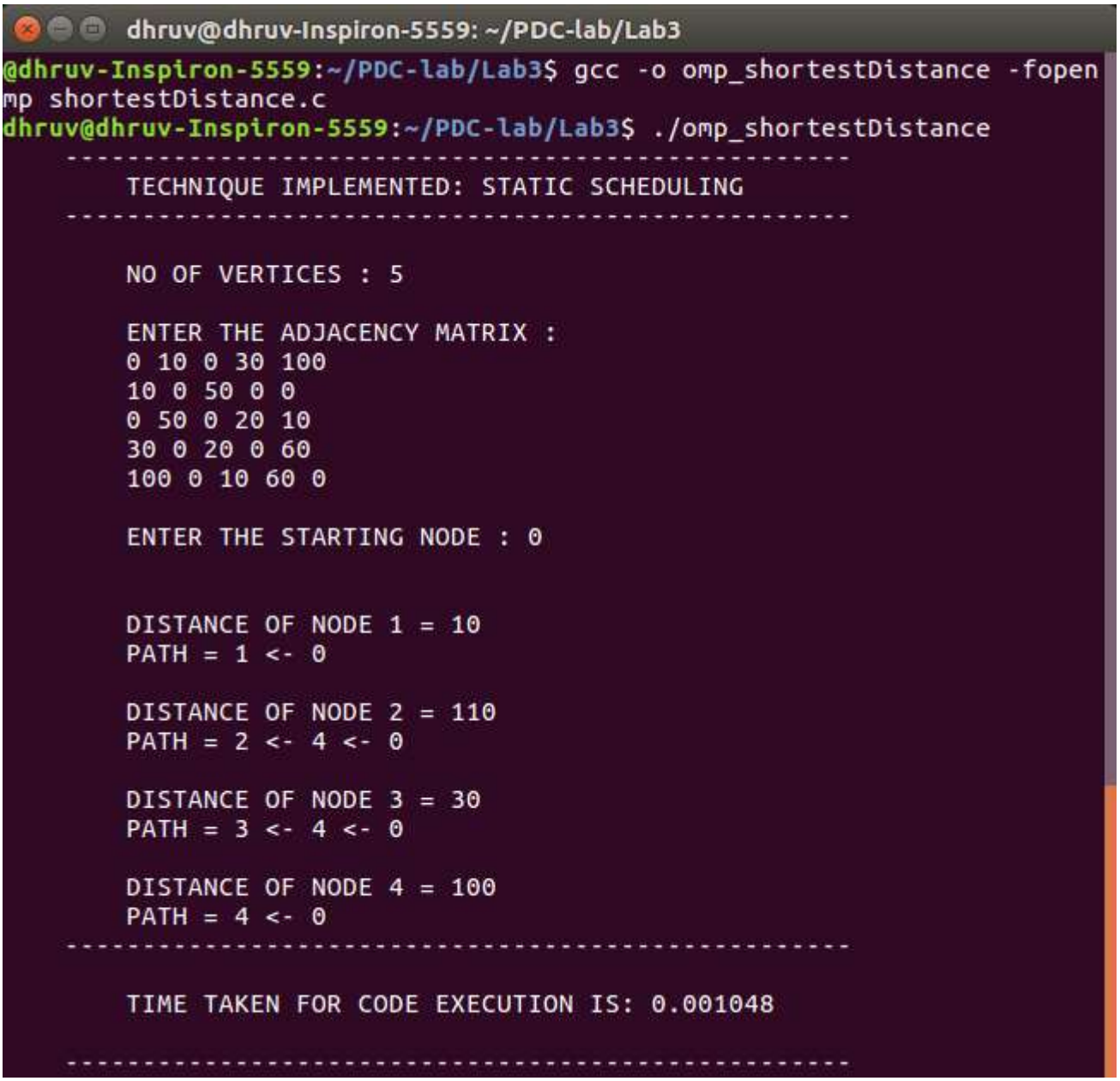
```c
#include<stdio.h>
#include<omp.h>
#include <time.h>

#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
    int G[MAX][MAX],i,j,n,u;
    printf("\n    ------------------------------------------------- ");
    printf("\n\tTECHNIQUE IMPLEMENTED: GUIDED SCHEDULING");
    printf("\n    ------------------------------------------------- ");
    printf("\n\n\tNO OF VERTICES : ");
    scanf("%d",&n);
    printf("\n\tENTER THE ADJACENCY MATRIX : \n\t");

    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
            scanf("%d",&G[i][j]);
    printf("\n\tENTER THE STARTING NODE : ");
    scanf("%d",&u);
    clock_t begin = clock();
    dijkstra(G,n,u);
    clock_t end = clock();
    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;
    printf("\n    ------------------------------------------------- ");
    printf("\n\n\tTIME TAKEN FOR CODE EXECUTION IS: %f\n", time_spent);
    printf("\n    ------------------------------------------------- \n");
    return 0;
}
```

```
35 void dijkstra(int G[MAX][MAX],int n,int startnode)
36 {
37     int cost[MAX][MAX],distance[MAX],pred[MAX];
38     int visited[MAX],count,mindistance,nextnode,i,j,k;
39     for(i=0; i<n; i++)
40         for(j=0; j<n; j++)
41             if(G[i][j]==0)
42                 cost[i][j]=INFINITY;
43             else
44                 cost[i][j]=G[i][j];
45     for(i=0; i<n; i++){
46         distance[i]=cost[startnode][i];
47         pred[i]=startnode;
48         visited[i]=0;
49     }
50     distance[startnode]=0;
51     visited[startnode]=1;
52     count=1;
53
54     #pragma omp parallel for schedule(guided)
55     for(k=1; k<n-1; k++){
56         mindistance=INFINITY;
57         for(i=0; i<n; i++)
58             if(distance[i]<mindistance&&!visited[i])
59             {
60                 mindistance=distance[i];
61                 nextnode=i;
62             }
63
64         visited[nextnode]=1;
65         for(i=0; i<n; i++)
66             if(!visited[i])
67                 if(mindistance+cost[nextnode][i]<distance[i])
68                 {
69                     distance[i]=mindistance+cost[nextnode][i];
70                     pred[i]=nextnode;
71                 }
72         count++;
73     }
```

```
74         count++;
75     }
76
77     for(i=0; i<n; i++)
78         if(i!=startnode)
79         {
80             printf("\n\n\tDISTANCE OF NODE %d = %d",i,distance[i]);
81             printf("\n\tPATH = %d",i);
82             j=i;
83             do
84             {
85                 j=pred[j];
86                 printf(" <- %d",j);
87             }
88             while(j!=startnode);
89         }
90 }
```

OUTPUT

STATIC Scheduling



Dynamic scheduling

```
dhruv@dhruv-Inspiron-5559: ~/PDC-lab/Lab3
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab3$ gcc -o omp_shortestDistance -
fopenmp shortestDistance.c
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab3$ ./omp_shortestDistance

    ----------------------------------------------------
        TECHNIQUE IMPLEMENTED: DYNAMIC SCHEDULING
    ----------------------------------------------------

        NO OF VERTICES : 5

        ENTER THE ADJACENCY MATRIX :
        0 10 0 30 100
        10 0 50 0 0
        0 50 0 20 10
        30 0 20 0 60
        100 0 10 60 0

        ENTER THE STARTING NODE : 0


        DISTANCE OF NODE 1 = 10
        PATH = 1 <- 0

        DISTANCE OF NODE 2 = 60
        PATH = 2 <- 1 <- 0

        DISTANCE OF NODE 3 = 30
        PATH = 3 <- 0

        DISTANCE OF NODE 4 = 100
        PATH = 4 <- 0
    ----------------------------------------------------

        TIME TAKEN FOR CODE EXECUTION IS: 0.002040

    ----------------------------------------------------
```

Guided scheduling

```
dhruv@dhruv-Inspiron-5559: ~/PDC-lab/Lab3
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab3$ gcc -o omp_shortestDistance -
fopenmp shortestDistance.c
dhruv@dhruv-Inspiron-5559:~/PDC-lab/Lab3$ ./omp_shortestDistance

    ----------------------------------------------------
        TECHNIQUE IMPLEMENTED: GUIDED SCHEDULING
    ----------------------------------------------------

        NO OF VERTICES : 5

        ENTER THE ADJACENCY MATRIX :
        0 10 0 30 100
        10 0 50 0 0
        0 50 0 20 10
        30 0 20 0 60
        100 0 10 60 0

        ENTER THE STARTING NODE : 0


        DISTANCE OF NODE 1 = 10
        PATH = 1 <- 0

        DISTANCE OF NODE 2 = 60
        PATH = 2 <- 1 <- 0

        DISTANCE OF NODE 3 = 30
        PATH = 3 <- 1 <- 0

        DISTANCE OF NODE 4 = 100
        PATH = 4 <- 0
    ----------------------------------------------------

        TIME TAKEN FOR CODE EXECUTION IS: 0.010209

    ----------------------------------------------------
```

INFERENCE

The inference from the above three screenshots is that static scheduling performed better than dynamic scheduling, which further performed better than guided scheduling.