SCHEDULING OF TASKS - SHORTEST PATH ALGORITHM

Name – Raagul N

Reg.No. – 16BCE1241

Course – Parallel and Distributed Computing

Course Code – CSE 4001

Faculty – Prof. Harini S.

PROBLEM STATEMENT -

For the shortest path finding problem, identify which of the scheduling is better.

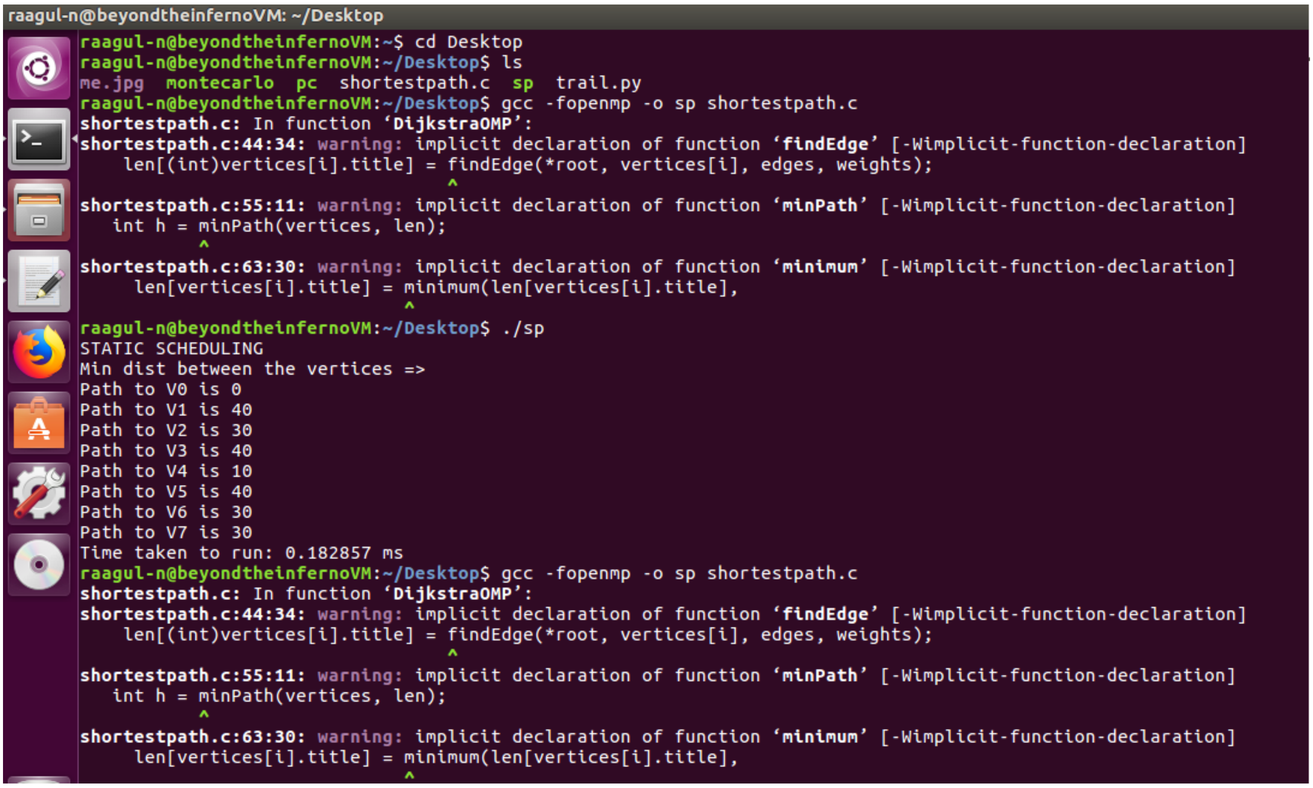
1. Static
2. Dynamic
3. Guided

CODE -

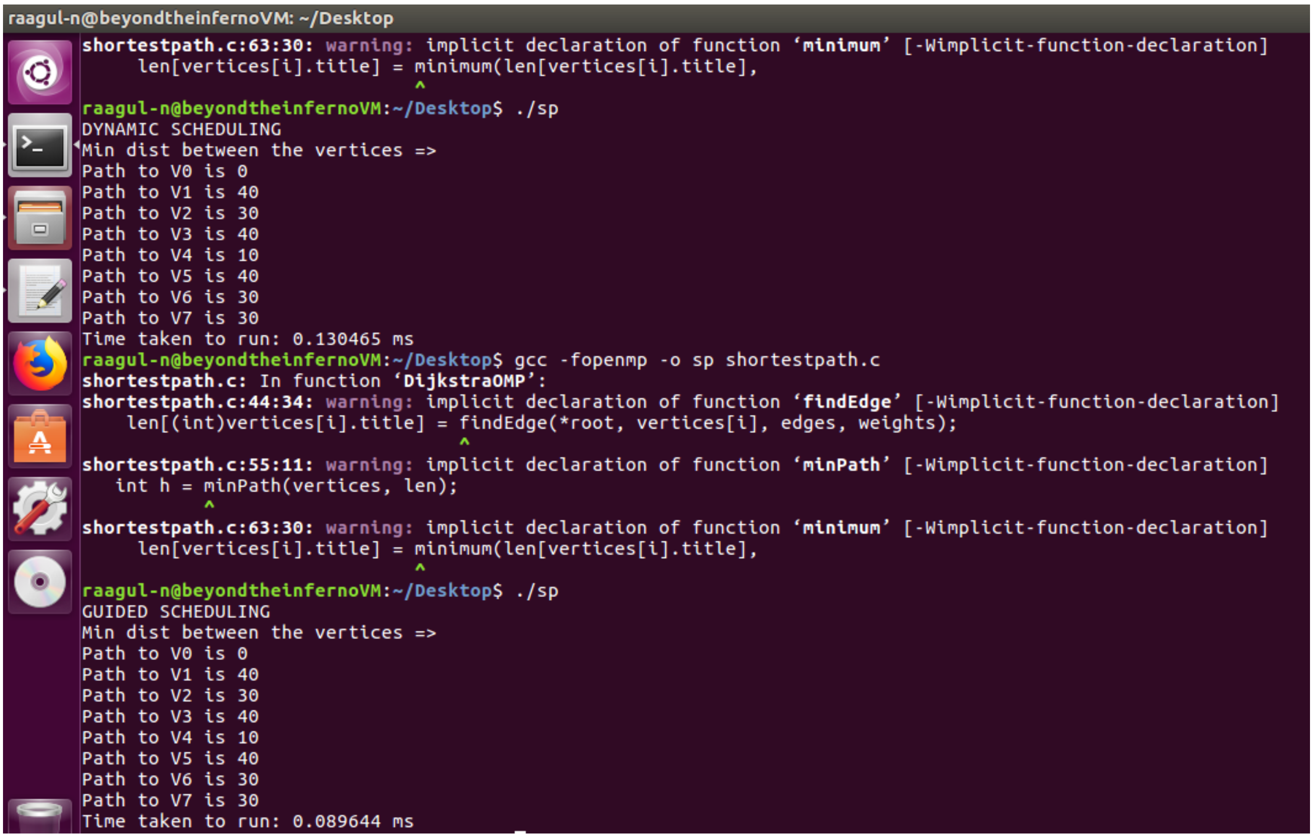
```
#include <stdio.h>
#include <time.h>
#include <math.h>
#include <omp.h>

#define INT_MAX 100000
#define TRUE 1
#define FALSE 0
#define V 8
#define E 11

typedef int bool;

typedef struct
{
  int u;
  int v;
} Edge;

typedef struct
{
  int title;
  bool visited;
} Vertex;

void printArray(int *array)
{
  int i;
  for(i = 0; i < V; i++)
    printf("Path to Vertex %d is %d\n", i, array[i]);
}

void DijkstraOMP(Vertex *vertices, Edge *edges, int *weights, Vertex *root)
{
  double start, end;
  root->visited = TRUE;
  int len[V];
  len[(int)root->title] = 0;
  int i, j;
  for(i = 0; i < V;i++)
  {
    if(vertices[i].title != root->title)
    {
      len[(int)vertices[i].title] = findEdge(*root, vertices[i], edges, weights);
    }
    else
    {
      vertices[i].visited = TRUE;
    }
  }
  start = omp_get_wtime();
  for(j = 0; j < V; j++)
  {
    Vertex u;
    int h = minPath(vertices, len);
    u = vertices[h];
    #pragma omp parallel for schedule(guided) private(i)
    for(i = 0; i < V; i++)
    {
      if(vertices[i].visited == FALSE)
      {
        int c = findEdge( u, vertices[i], edges, weights);
        len[vertices[i].title] = minimum(len[vertices[i].title],
        len[u.title] + c);
      }
    }
  }
  end = omp_get_wtime();
  printArray(len);
  printf("Running time: %f ms\n", (end - start)*1000);
}
```

```c

int findEdge(Vertex u, Vertex v, Edge *edges, int *weights)
{
  int i;
  for(i = 0; i < E; i++)
  {
    if(edges[i].u == u.title && edges[i].v == v.title)
    {
      return weights[i];
    }
  }
  return INT_MAX;
}

int minimum(int A, int B)
{
  if( A > B)
  {
    return B;
  }
  else
  {
    return A;
  }
}

int minWeight(int *len, Vertex *vertices)
{
  int i;
  int minimum = INT_MAX;
  for(i = 0; i < V; i++)
  {
    if(vertices[i].visited == TRUE)
    {
      continue;
    }
    else if(vertices[i].visited == FALSE && len[i] < minimum)
    {
      minimum = len[i];
    }
  }
  return minimum;
}

int minPath(Vertex *vertices, int *len)
{
  int i;
  int min = minWeight(len, vertices);
  for(i = 0; i < V; i++)
  {
    if(vertices[i].visited == FALSE && len[vertices[i].title] == min)
    {
      vertices[i].visited = TRUE;
      return i;
    }
  }
}

int main(void)
{
  Vertex nodes[V];
  Edge edges[E] ={{0, 4}, {0, 6}, {0,2}, {4,6}, {4,7}, {0, 7}, {7, 3}, {3, 1}, {2,5}, {2, 1},{5,3}};
  int weights[E] = {10, 90, 30, 20, 20, 50, 10, 20, 10, 10, 10};
  int i = 0;
  for(i = 0; i < V; i++)
  {
    Vertex a = { .title =i , .visited=FALSE};
    nodes[i] = a;
  }
  Vertex root = {0, FALSE};
  printf("Min dist between the vertices => \n");
  DijkstraOMP(nodes, edges, weights, &root);
}

```

OUTPUT -



```
raagul-n@beyondtheinfernoVM:~$ cd Desktop
raagul-n@beyondtheinfernoVM:~/Desktop$ ls
me.jpg  montecarlo  pc  shortestpath.c  sp  trail.py
raagul-n@beyondtheinfernoVM:~/Desktop$ gcc -fopenmp -o sp shortestpath.c
shortestpath.c: In function 'DijkstraOMP':
shortestpath.c:44:34: warning: implicit declaration of function 'findEdge' [-Wimplicit-function-declaration]
    len[(int)vertices[i].title] = findEdge(*root, vertices[i], edges, weights);
                                  ^
shortestpath.c:55:11: warning: implicit declaration of function 'minPath' [-Wimplicit-function-declaration]
   int h = minPath(vertices, len);
           ^
shortestpath.c:63:30: warning: implicit declaration of function 'minimum' [-Wimplicit-function-declaration]
     len[vertices[i].title] = minimum(len[vertices[i].title],
                              ^
raagul-n@beyondtheinfernoVM:~/Desktop$ ./sp
STATIC SCHEDULING
Min dist between the vertices =>
Path to V0 is 0
Path to V1 is 40
Path to V2 is 30
Path to V3 is 40
Path to V4 is 10
Path to V5 is 40
Path to V6 is 30
Path to V7 is 30
Time taken to run: 0.182857 ms
raagul-n@beyondtheinfernoVM:~/Desktop$ gcc -fopenmp -o sp shortestpath.c
shortestpath.c: In function 'DijkstraOMP':
shortestpath.c:44:34: warning: implicit declaration of function 'findEdge' [-Wimplicit-function-declaration]
    len[(int)vertices[i].title] = findEdge(*root, vertices[i], edges, weights);
                                  ^
shortestpath.c:55:11: warning: implicit declaration of function 'minPath' [-Wimplicit-function-declaration]
   int h = minPath(vertices, len);
           ^
shortestpath.c:63:30: warning: implicit declaration of function 'minimum' [-Wimplicit-function-declaration]
     len[vertices[i].title] = minimum(len[vertices[i].title],
                              ^
```



```
shortestpath.c:63:30: warning: implicit declaration of function ‘minimum’ [-Wimplicit-function-declaration]
     len[vertices[i].title] = minimum(len[vertices[i].title],
                              ^
raagul-n@beyondtheinfernoVM:~/Desktop$ ./sp
DYNAMIC SCHEDULING
Min dist between the vertices =>
Path to V0 is 0
Path to V1 is 40
Path to V2 is 30
Path to V3 is 40
Path to V4 is 10
Path to V5 is 40
Path to V6 is 30
Path to V7 is 30
Time taken to run: 0.130465 ms
raagul-n@beyondtheinfernoVM:~/Desktop$ gcc -fopenmp -o sp shortestpath.c
shortestpath.c: In function ‘DijkstraOMP’:
shortestpath.c:44:34: warning: implicit declaration of function ‘findEdge’ [-Wimplicit-function-declaration]
    len[(int)vertices[i].title] = findEdge(*root, vertices[i], edges, weights);
                                  ^
shortestpath.c:55:11: warning: implicit declaration of function ‘minPath’ [-Wimplicit-function-declaration]
   int h = minPath(vertices, len);
           ^
shortestpath.c:63:30: warning: implicit declaration of function ‘minimum’ [-Wimplicit-function-declaration]
     len[vertices[i].title] = minimum(len[vertices[i].title],
                              ^
raagul-n@beyondtheinfernoVM:~/Desktop$ ./sp
GUIDED SCHEDULING
Min dist between the vertices =>
Path to V0 is 0
Path to V1 is 40
Path to V2 is 30
Path to V3 is 40
Path to V4 is 10
Path to V5 is 40
Path to V6 is 30
Path to V7 is 30
Time taken to run: 0.089644 ms
```