Assignment No B-2

Roll No: 4430

Aim

Concurrent Implementation of travelling salesman problem.

Problem Definition

Concurrent Implementation of travelling salesman problem.

Objective

1. To understand idea travelling salesman problem.
2. To implement travelling salesman problem using Cpp.

Mathematical Model

Let S be the system for implementing Binary Search using divide and conquer strategy.

S = (s, e, x, y, F, scr, DD, NDD, Mem_{sh}, Success case, Failure case, CPU core count)

Where, s = Start
State e = End
State x = Input
y = Output
F = Set of function
DD = Deterministic Data
NDD = Non-Deterministic Data

1. Start State = main() 2.

End State = return 0

3. Input:
   Let x be the set of inputs such as $x = \{x_1, x_2\}$ x_1 = no of cities.
   x_2 = cost matrix.

4. Output:
   Let y be the set of outputs such as , y = Concurrent Implementation of travelling salesman problem.

5. F = Set of function

   F = { f_1,f_2,f_3,f_4,f_5,f_6 } f_1 = void getdata() f_2 = void display() f_3 = int fact(int num) f_4 = int min(int list[]) f_5 = void perm(int list[], int k, int m) f_6 = void sol()

6. scr = screen

7. Deterministic Data:

   Get the minimum path.

8. Non-Deterministic Data:Null

9. MEM_{sh} = Shared memory is not used in this case.

10. Success Case = Path with Minimum cost found
11. Failure Case = If every path having same cost

12. CPU core count = 2

Theory

Travelling salesman problem

In travelling salesman problem, the shortest possible route is found that visits every city exactly once and returns to the starting point. We have set of cities and distance between every pair of cities.

TSP can be considered as an undirected weighted graph, in such a way that:

1. The cities are the graph's vertices
2. Paths are the graph's edges
3. A path's distance is the edge's length.

TSP is a minimization problem which starts and finishes at a specified vertex after visiting each other vertex exactly once. Usually the model is a complete graph (i.e. each pair of vertices is connected by an edge). If there is no path existing between two cities, then an arbitrarily long edge is added to complete the graph without affecting the optimal tour.

The Travelling Salesman Problem is NP-hard problem that is, there is no exact algorithm to get it solved in polynomial time. The time complexity of travelling salesman problem using dynamic programming is O(n2*2n).

The exact methods for solving TSP are:

1.Explicit enumeration
2.Implicit enumeration
3.Branch and bound method
4.Cutting plane method
5.Dynamic programming

In our program we will be using dynamic programming in order to redue the time complexity of normal TSP.

Applications of TSP:

1.Planning
2.Logistics
3.Manufacture of microchips
4.Astronomy

Following are steps for execution of this program.

Step 1 : Take number of cities to visit.
Step 2 : Enter the cost of each cities in cost matrix.
Step 3 : Compute the tour starting from the current city.

Step 4 : Set the minimum-tour array. Step 5 : Display the minimum cost.

Program Code and Output

```
#include<stdio.h>
#include<iostream>
#include<omp.h>
using namespace std;
int graph[5][5],cost = 999;

void swap (int *x, int *y) {
int temp;
temp = *x;
*x = *y;
*y = temp;
}

void copy_array(int *a, int n){
int i, sum = 0;
 #pragma omp parallel for
   for(i = 0; i <= n; i++){
            sum += graph[a[i % 5]][a[(i + 1) % 5]];}
if (cost > sum)
{
cost = sum;
}
}

void permute(int *a, int i, int n)
{
int j, k;
if (i == n)
{
 #pragma omp parallel sections
   {
      copy_array(a, n);
   }
  }
 else {
#pragma omp parallel for
for (j = i; j <= n; j++)
{
swap((a + i), (a + j));
 permute(a, i + 1, n);
swap((a + i), (a + j));

     }}}
```

```
int main()

{
cout<<"Enter the elements for 5*5 array";
 for(int i=0;i<5;i++)
   {
      cout<<"\n Enter the elements of "<<i+1<< "th row :\t";
      for(int j=0;j<5;j++)
      {
         cin>>graph[i][j];
         cout<<"\t";
      }
      cout<<"\t";
   }
int i, j;
int a[] = {0, 1, 2, 3,4};
int c = 0;
permute(a, 0, 4);
cout<<"\n\n\t\tminimum cost:"<<cost<<endl;
}

/*
ameeth@ubuntu-16.0.4:~/CL1$ g++ tspconcurrent.cpp -o tspconcurrent -fopenmp
ameeth@ubuntu-16.0.4:~/CL1$ ./tspconcurrent
Enter the elements for 5*5 array
 Enter the elements of 1th row :    -999 10 8 9 7
 Enter the elements of 2th row :    10 -999 10 5 6
Enter the elements of 3th row :    8 10 -999 8 9
Enter the elements of 4th row :    9 5 8 -999 6
  Enter the elements of 5th row :    7 6 9 6 -999
minimum cost:34
ameeth@ubuntu-16.0.4:~/CL1$
*/
```

Conclusion

Hence, in this assignment we calculated the minimum path for Travelling salesman problem.