Michael Popplewell

Programming Assignment 5

Traveling Salesperson Problem with a Stack

2 December 2016

Abstract

The problem to be solved with this program is, given a tree, to find the shortest route between any two nodes. This should be done using as time efficient a method as possible. Some issues that will have to overcome to solve this problem is creating a way for various piece of information to be 'remembered' – which cities have been visited, the current path being taken. This is due to the necessity for comparisons but also for output.

The program uses a stack as its main backbone. Visited 'cities' are represented as elements within an array which is empty at the beginning. Also, near the beginning, the first city is set to as visited within the array and that city is placed within the stack. Another variable that tracks the closest city is set to 0 and a Boolean variable 'minFlag' is set to false. This non-recursive program then uses a while statement that continues as long as the stack is not empty. Within the while statement, the current city being analyzed is set to the top of the stack, and minimum distance is set to the max value an integer can reach. This will be used as a form of exception catching without actually catching and throwing an exception. The program, using an 'if' statement, checks if the distance from the current city to an adjacent city is not 0 and is not visited. If so, if the distance is less than min, min is set as the new distance, closest city is set to the new city, and the Boolean variable 'minFlag' is set to true. If this Boolean variable is set to true, another block of code sets the closest city as visited and pushes it onto the stack. Closest city is then outputted and 'minFlag' is reset to false to later be triggered if the need arises. At the end of the while statement that encloses much of the above, the top element of the stack is popped.

Running tsp29.txt through the program took one millisecond. The others didn't register on the time duration counter. So, I took out the time duration converter and kept the time as nanoseconds. 'tsp29.txt' took 617,432 nanoseconds – a little more than a millisecond – to execute. Which means, the one millisecond readout is rather inaccurate. 'tsp.12' took 387,452 nanoseconds to execute. Here is a chart of the times:

File	Time (Nanoseconds)	Avg. per Element (Time ÷ Elem. In Adj. Matrix)
'tsp12.txt'	387452	2690
'tsp13.txt'	371677	2199
'tsp14.txt'	396031	2020
'tsp15.txt'	433670	1927
'tsp16.txt'	424260	1657
'tsp19.txt'	469370	1300
'tsp29.txt'	823058	978

This algorithm is extremely time-efficient. While the depth-first search algorithm in Lab 5 took many milliseconds to execute on the smaller trees such as the one in 'tsp12.txt' and wouldn't even run through 'tsp29.txt' within a reasonable amount of time, the algorithm within Programming Assignment 6 that utilizes stacks blazes through any tree that I've thrown at it thus far. Also, unlike the depth-first algorithm, the faster algorithm does have an exponential increase in time duration. In fact, as the trees grows larger, the time efficiency for each element is increased. While it took an average of 2690 nanoseconds per element for 'tsp12.txt' (144 elements), 'tsp29.txt' (841 elements) was traversed at a rate of 978 nanoseconds per element. This is rather interesting and I'm not sure, exactly what explains these results. However, I do have a hypothesis.

With a larger tree, there are many different paths from one node to another, all the nodes are more interconnected. This means that, when visiting one node, there is easy access to a tremendous amount of others. It’s the same thing that happens within cities. A city has many different streets so, from one location, there are a lot of streets that connect the location to many others. Meanwhile, take a small town. There aren’t as many streets and, in turn, there are less ways of getting from one location to another. If one were to visit every location in the small town, they would be able to do it faster but, because there are less paths, the time to get to each of these locations would take a longer amount of time. The city, has many different ways of getting from one location to another, more direct routes. So, if one were to visit every location in a city, it would take far longer. However, getting to each location would take tremendously less time because of the increase in direct routes.

OUTPUTS

TSP.12
```
1        11       1        4        8        10       9        3        5        7        6        2
```
Cost of path: 935
Time duration for tsp.12: 518079

TSP.13
```
1        10       1        8        7        2        12       6        9        3        11       5        4
```
Cost of path: 1102
Time duration for tsp.13: 478780

TSP.14
```
1        13       1        11       6        7        10       9        3        5        4        8        2
         12
```
Cost of path: 781
Time duration for tsp.14: 479610

TSP.15
```
1        13       14       12       4        8        10       9        3        5        1        11       6
         7        2
```
Cost of path: 956
Time duration for tsp.15: 494278

TSP.16
```
1        9        3        14       13       10       15       12       1        4        8        5        11
         7        6        2
```
Cost of path: 1397
Time duration for tsp.16: 465219

TSP.19
```
1        9        3        14       18       15       12       1        4        8        5        11       7
         6        10       13       17       16       2
```
Cost of path: 1434
Time duration for tsp.19: 844368

TSP.29
```
1        20       1        19       9        3        14       18       24       6        22       26       23
         7        27       5        11       8        4        25       28       2        12       15       10
         21       13       17       16
```
Cost of path: 1658
Time duration for tsp.29: 874534