Colin Rubow

CS 312

Project 5: TSP Project

Contents:

1) Well-Commented Code

2) Time and Space Complexity

3) Data Structures for States

4) Priority Queue

5) Initial BSSF Approach

6) Table of Trials

7) Discussion on Results

8) Search Mechanism

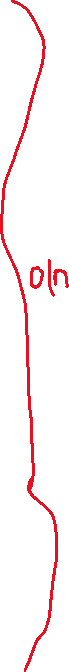
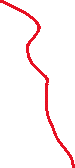
**1) Well-Commented Code:**

The following is the code for the BranchAndBound implementation.

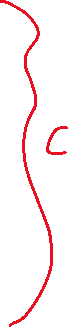












**2) Time and Space Complexity**

The time complexity of this algorithm can be quite horrendous, overall, about . Following the code from top to bottom we see that we initialize some conditions and then perform the greedy algorithm, which has a complexity of about . We initialize a few more conditions, including a 2x2 array for storing the costs of going from one state to another which is clearly . We initialize a few more conditions and start the searching portion of our algorithm. We see a while loop and can determine that the complexity of the overall process of this loop is in . This is because for every city, we create n more states to explore. The next iteration we create at most n – 1 more states and so forth and thus we have about which is equipollent to . In this looping process we retrieve the next state to expand, being time, and get some data and make some constant time decisions. We then for every available city to go to we create a new state and place it in the queue which can be seen to be an process. We finish the process with some constant time reporting. The particulars of various parts of this algorithm will be discussed later.

The space complexity of this algorithm can be estimated from the maximum number of states that are stored in the priority queue. At 15 cities on Hard (Deterministic), the max queue size is at 37.

**3) Data Structures for States**

The way a state was stored was using a simple object called a State. The state held three objects. The scoring matrix, the lower bound, and the route taken thus far. It had two methods: one for retrieving the current city location, and the other for getting the length of the route taken thus far. This second method gives us an idea of how deep this state is in the branching process.

**4) Priority Queue**

In project 3 we implemented a priority queue heap. This project uses the same priority queue heap but with some tweaks. The priority was determined by the depth of the state, or how many cities the state had already visited. This guarantees that the algorithm dives deep quickly to find solutions and perform pruning quickly. The time complexity of insertion and retrieval is .

**5) Initial BSSF Approach**

The initial BSSF approach uses a greedy algorithm that sequentially starts at each city and returns the best solution found in that approach.

**6) Table of Trials**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| #Cities | Seed | Running Time (sec.) | Cost of Best Tour (\*=optimal) | Max # of stored States | # of BSSF updates | Total # of states created | Total # states pruned |
| 15 | 20 | 10.863958 | 10534\* | 37 | 3 | 69584 | 58435 |
| 16 | 902 | 60.001 | 8192 | 68 | 34 | 347024 | 287685 |
| 17 | 82 | 60.000 | 9987 | 73 | 2 | 317317 | 265335 |
| 18 | 453 | 33.698958 | 10197\* | 108 | 72 | 166378 | 141828 |
| 19 | 947 | 60.0016 | 10143 | 65 | 7 | 264609 | 228264 |
| 20 | 59 | 60.00 | 13716 | 88 | 10 | 247311 | 216626 |
| 21 | 476 | 60.00 | 0 | 88 | 0 | 32066 | 28017 |
| 22 | 977 | 60.00 | 0 | 110 | 0 | 205344 | 180006 |
| 23 | 888 | 60.002 | 0 | 113 | 0 | 192674 | 170925 |
| 24 | 546 | 60.001 | 0 | 106 | 0 | 182299 | 156108 |

**7) Discussion on Trials**

It can be seen that at higher numbers of cities the algorithm is not as likely to find a solution in the given time. The trial of 18 cities was surprising that it was able to find the optimal solution while it couldn’t the previous two trials. It can be clearly seen that number of states created and overall complexity of the algorithm increases greatly with the addition of just one city.

**8) Search Mechanism**

The priority queue prioritized the depth of a state. This means the algorithm is basically performing a depth-first search for solutions. This is probably why my algorithm is not very good at finding solutions at 16+ cities. If my priority also allowed more branching than diving, maybe more solutions could be found easier and thus more pruning as well.