Jiwon Cha (jc4va)
11/30/17

Post lab 11

Complexity analysis – topological sort

Time complexity

Main function:

- Read in a file, insert the strings in the input file into a vector of strings: Big-theta(n)
- Sort the vector of strings for every input, create a vertex and push it to the adjacency list: Big-theta(n)
- Read in a file again, if the string already exists in the adjacency list, push it to the vector of vertices within that vertex: Big-theta(n)

Total = Big-theta(n^2)

Topological sort: Big-theta(n)

Space complexity

Vertex:

- Integer value: 4 bytes
- String value: 6 bytes
  - Each input contains 6 characters: e.g. CS1110
- vector<vertex*>: 100 x 10 + 24 = 1024 bytes

Main method:

- Strings: 4 x 6 + 13 = 37 bytes
- Integer values: 4 bytes
- Vector of strings: 100 x 6 + 24 = 624 bytes

Topsort function:

- queue<vertex*>: 100 x 10 + 24 = 1024 bytes
- vertex pointers: 8 x 2 = 16 bytes

Total = 2739 bytes

Complexity analysis – traveling salesperson

Time complexity

The next_permutation() method gives a time complexity of Big-theta(n!).

Space complexity

Middle earth:

- Vector of floats
  - xpos = 4 x xpos
  - ypos = 4 x ypos
- Vector of string
  - cities = 4 x num_city_names
- Integers: 4 x 4 = 16 bytes
  - xsize, ysize, numcities, seed
- Pointer to a float: 8 bytes

Total = 4 x xpos + 4 x ypos + 4 x num_city_names + 24

Traveling:

- Middle Earth declaration
- Vector of string: 4 x num_cities bytes
- Float: 4 bytes

Compute distance:

- Float: 4 bytes

Total: Middle Earth total + 4 x num_cities + 8 bytes

Acceleration techniques

1. Branch and bound

Branch and bound algorithm firsts sets a bound for a 'candidate value' for the best possible outcome. In other words, branch and bound algorithm computes the current best outcome, and if the bound is less efficient than the current best value, the bound becomes 'nonpromising'. Therefore, There are three ways of traversing the graph; breadth first search, depth first search and best first search. Breadth first search traverses the graph by level, and best first search starts from the vertices that have the smallest bounds. The worst case time complexity remains the same $O(2^n \times n^2)$ and it is difficult to predict by how much this algorithm will improve the running time but in practice, it runs much faster than the brute-force algorithm as the time complexity of enumeration and pruning is O(1) per each vertex.

2. Minimum spanning tree

The minimum spanning tree assigns a number (cost) to each edge, and when traversing, the program creates a graph that connects every vertex using the edges that have the minimum costs. The minimum spanning tree cannot have a cycle, because the fundamental notion of a tree states that there cannot be a cycle. There are two ways to implement the minimum spanning tree: Prim's algorithm and Kruskal's algorithm. Prim's algorithm chooses a root and then traverses the tree by choosing the connected edges with the minimum costs. These steps are repeated until every vertex is included in the spanning tree. Kruskal's algorithm first chooses an edge that has the smallest cost and then traverses the tree by choosing the edges with the smallest costs. The running time of Prim's algorithm and Kruskal's algorithm is the same (O(m log n))

3. Nearest neighbor

The nearest neighbor uses a similar algorithm with minimum spanning tree and branch and bound. It picks a city as a starting point and goes to an unvisited point that has the shortest distance from the starting point. After each traverse, it sets the new point as the 'current' point and it marks that point as 'visited'. These steps repeat until every point in the graph is visited. The algorithm runs at O(log n) average case.

References

https://link.springer.com/chapter/10.1007/978-3-319-00951-3_11
http://www.geeksforgeeks.org/travelling-salesman-problem-set-2-approximate-using-mst/
http://www.geeksforgeeks.org/branch-bound-set-5-traveling-salesman-problem/
http://findfun.tistory.com/385
https://www.mathworks.com/matlabcentral/fileexchange/25542-nearest-neighbor-algorithm-for-the-travelling-salesman-problem?requestedDomain=www.mathworks.com