CS 325 Project 4: The Travelling Salesman Problem (TSP)

Kyle Guthrie
Michael C. Stramel
Alex Miranda

December 2, 2016

Summary

Problem Statement

what was the basic deal with the project

Algorithms Researched/Implemented

what were the four algorithms we implemented - which one did we ultimately use (2-opt) why did we select them? what research did we do?

Results

maybe put the raw data from our summary stats here? or maybe just the results that we are turning in from 2-opt

Algorithm: Braindead Tour

These sections are the minimum of what we need to cover based on the Project 4 guidelines. Feel free to add additional sections!

Algorithm Description

A description of the algorithm, and a summary of the research performed. I'd like to recommend we use BibTeX here to keep track of our sources, and make referencing easy. In the report directory, you'll find a `report.bib` file. In that file, you can add your sources. Then to cite a source, you use the `\cite{citationName}` command. Check out the section titled "3 Bibliography management with Bibtex" in the referenced website to see how it works. [1] References appear at the end of the report PDF.

Algorithm Discussion

A brief discussion of why you chose this algorithm.

Algorithm Pseudo-code

```
if (fast_way)
  your pseudo_code here
  return
```

Algorithm: Depth First Search of Minimum Spanning Tree

Algorithm Description

We learned about the minimum spanning tree heuristic from a discussion in [2]. The concept is to find the minimum spanning tree (MST). Then the tour order is set by the discovery order while performing a depth first search (DFS) on the minimum spanning tree.

Algorithm Discussion

We implemented this algorithm because it seemed relatively simple. This this implementation we used Python 3.5. At first we implement Prim's algorithm to find the MST but we found it to be too slow. Therefore, we switched to using Kruskal's algorithm to find the MST which provided a better running time of O(n lg n). Finally, according to [2], this heuristic is typically 15% to 20 over optimal in practice. Unfortunately, while it was decently fast for smaller input sizes, it was inefficent for very large input sizes ($n \geq 5000$). Also, for the three example cases it didn't provide the expected results relative to optimal. Depending on the case the route found was 31% to 88% worse than optimal.

Algorithm Pseudo-code

```
DFSofMST()
    cities = []
    read date from file into cities[] // each city has id, x, and y attributes

    adjMatrix=[len(cities)][len(cities)]
    for i = 0 to len(cities) - 1
        for j = i + 1 to len(cities)
            ij = dist(cities[i][j])
            adjMatrix[i][j] = ij
            adjMatrix[i][j] = ji

    mst = kruskalsAlg(adjMatrix)
    adjList = mstToAdjList(adjMatrix, mst)
    disc = dfs(adjList)

    totalDist = 0
    prevCity = cities[disc[0]]
    for i = 1 to len(disc)
        eachCity = cities[disc[i]]
        addDist = dist(eachCity, prevCity)
        totalDist += addDist
        prevCity = eachCity
    addDist = dist(prevCity, cities[disc[0]])
```

```
    totalDist += addDist

    write totalDist to file
    write contents of disc[] to file // 1 item per line

dist(cityOne, cityTwo)
    dx = cityOne['x'] - cityTwo['x']
    dy = cityOne['y'] - cityTwo['y']
    dxSq = dx ^ 2
    dySq = dy ^ 2
    return ((dxSq + dySq) ^ 0.5)

dfs(adjList)
    vstd = []
    stack = []
    stack.append(0)

    while(len(stack) > 0)
        u = stack[len(stack) - 1]
        stack.remove(stack[len(stack) - 1])
        if not (u in vstd)
            vstd.append(u)
            auxStack = []
            for eachAdj in reverse_sorted(adjList[u].items()):
                v = eachAdj[0]
                if not (v in vstd):
                    auxStack.append(v)
            while len(auxStack) > 0
                v = auxStack[0]
                stack.append(v)
                auxStack.remove(v)

    return vstd

mstToAdjList(adjMatrix, mst)
    adjList = {}
    for i = 0 to len(adjMatrix)
        adjV = {}
        for j = 0 to len(mst)
            if i == mst[j]
                adjV[j] = adjMatrix[i][j]
        adjList[i] = adjV
    return adjList

popMin(prqu)
```

```
    minU = prqu[0]
    prqu.remove(prqu[0])
    return minU

decreaseKey(prqu, cost)
    for i = 0 to len(prqu)
        for j = 0 to len(prqu)
            if cost[prqu[i]] < cost[prqu[j]]
                prqu[i] = prqu[i] + prqu[j]
                prqu[j] = prqu[i] - prqu[j]
                prqu[i] = prqu[i] - prqu[j]

def kruskalsAlg(adjMatrix)
    edges = []
    for i = 0 to len(adjMatrix) - 1
        for j = i + 1 to len(adjMatrix)
            e = edge(i, j, adjMatrix[i][j])
            edges.append(e)
    edges.sort(key = lambda x: x.d)

    prev = [None for x in range(len(adjMatrix))]
    vstd = []
    for e in edges:
        if (not(e.v in vstd))
            vstd.append(e.v)
            prev[e.v] = e.u
    return prev
```

Algorithm: 2-Optimal Tour

These sections are the minimum of what we need to cover based on the Project 4 guidelines. Feel free to add additional sections!

Algorithm Description

A description of the algorithm, and a summary of the research performed. I'd like to recommend we use BibTeX here to keep track of our sources, and make referencing easy. In the report directory, you'll find a `report.bib` file. In that file, you can add your sources. Then to cite a source, you use the `\cite{citationName}` command. Check out the section titled "3 Bibliography management with Bibtex" in the referenced website to see how it works. [1] References appear at the end of the report PDF.

Algorithm Discussion

A brief discussion of why you chose this algorithm.

Algorithm Pseudo-code

```
if (fast_way)
  your pseudo_code here
  return
```

Algorithm: Nearest Neighbor

These sections are the minimum of what we need to cover based on the Project 4 guidelines. Feel free to add additional sections!

Algorithm Description

A description of the algorithm, and a summary of the research performed. I'd like to recommend we use BibTeX here to keep track of our sources, and make referencing easy. In the report directory, you'll find a `report.bib` file. In that file, you can add your sources. Then to cite a source, you use the `\cite{citationName}` command. Check out the section titled "3 Bibliography management with Bibtex" in the referenced website to see how it works. [1] References appear at the end of the report PDF.

Algorithm Discussion

A brief discussion of why you chose this algorithm.

Algorithm Pseudo-code

```
if (fast_way)
  your pseudo_code here
  return
```

References

[1] ShareLaTeX. (Nov. 2016). Bibliography management with bibtex, [Online]. Available: https://www.sharelatex.com/learn/Bibliography_management_with_bibtex#/Bibliography_management_with_Bibtex.

[2] S. S. Skiena, *The Algorithm Design Manual*. Springer, 2008.