2d Lt. James Elkins

CSCE 686 - Homework 5

June 02, 2020

Set Complete Problem

Problem 1:

1.a - Small test problem: Graph size - 20 nodes, 20 subsets

RESULTS:

Graph 1 time - .14 seconds

Graph 2 time - .14 seconds

Graph 3 time - .13 seconds

Medium test problem: Graph size - 40 nodes, 40 subsets

Results:

Graph 1 time - .53 seconds

Graph 2 time - .55 seconds

Graph 3 time - .54 seconds

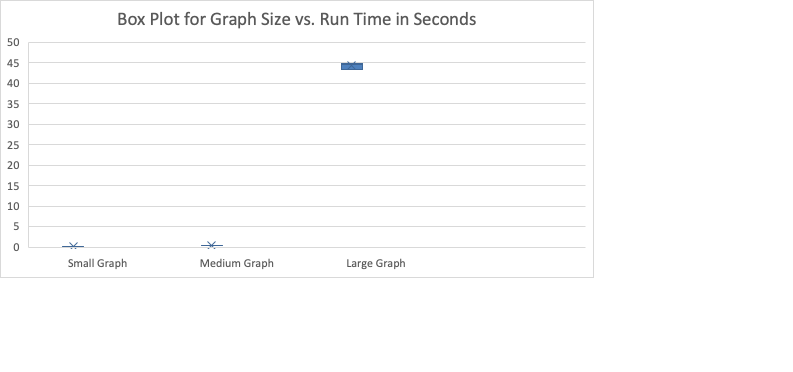
Large test problem: Graph size - 60 nodes, 60 subsets

Results:

Graph 1 time - 43.49 seconds

Graph 2 time - 44.65 seconds

Graph 3 time - 44.96 seconds



From the small amount of data points gathered from running the SCP Solver,

1.b - graph is included in other document named Elkins\_mediumSearchGraph.pdf

1.c - The overall complexity of the problem domain is 2^n. The algorithm domain however includes a polynomial factor due to for loops that occur within the methods that are called in the code making the complexity of the SCP Solver n^2 \* 2^n.

Problem 2:

2.a - Greedy Heuristic:

Q): set covering problem where Z = mins in N{∑j in S fi : z(S) = Z(N)} (

Q): basically the minimization of the set covers

Set t = 1. S0=Ø. Stop if z (Ø) = z(N)

Iteration t. Let θt = minj in N\St-1 {fj / Qj (St-1)}

Let arg min{fj / Qj (St-1} = jt

Let Qt = Qjt (St-1} Set St = St-1 U { jt }and σt = z(St) – z(St-1)

Stop if z(St) = z(N) and set T=t

Otherwise set t = t+1

Referenced from <http://www.bioinfo.org.cn/~dbu/AlgorithmCourses/Lectures/Lec7-SubModular-Set-Cover1982.pdf>

2.b - A greedy approximation for SCP is to pick the set that covers the most amount of uncovered nodes, remove those nodes from the set of nodes that are still uncovered, and then repeat the process until all nodes are covered.

If the optimal algorithm will find the solution with k sets, then at worst, the greedy approximation will find a solution with k log(n) sets.

Reference: <https://www.cs.cmu.edu/~avrim/451f12/lectures/lect1106.pdf>

2.c - I found a python implementation online at the reference below. After running the python code on similar size graphs as to when I ran the java implementation of the AFIT SCP Solver, the python implementation consistently took less time.

<http://www.martinbroadhurst.com/greedy-set-cover-in-python.html>

Problem 3:

3.a -

Primarily the AFIT SCP Solver does not use good software engineering principles that allow for the code to be easily comprehendible, modified, and debugged. The first issue with the code is that there are almost no comments to help understand what the different functions are doing. Secondly, the combination of the python code to generate input files and the java code to be the actual solver was not communicated anywhere and does not make since as to why this is the case. There is no good reason that the java code should not be able to make random graphs as well as be the solver. In addition to that, naming the folder with the python code “AFIT SCP - Python” further complicated this homework making it seem like that code should be a full solver that was written in python. Additionally, the variables in the java code do not give any information for what they are making an uncommented program more difficult to understand.

Problem 4:

4.a - When conducting a bombing raid, pilots need to coordinate their flight paths with each other in order to figure out the most efficient and effective way to destroy all of the desired targets. Each bomb dropped and the area that it effects can be seen as a subset of the universal set that contains the desired targets. Pilots should strive to use the minimum amount of bombs required to get their mission done which makes this a great application for the set covering problem.