Rohan Singh
UID : 116166331

Homework 1

CMSC 818B
September 18, 2019

1 Problem 1

1.1 Algorithm Description

The Graph is stored in the form of an adjacency list. The list of neighbouring nodes, for each node, is sorted in increasing order. To generate MST, a random node is chosen (1st node in my case). A priority queue is defined which will store all potential candidates of next edges that could be added to the spanning tree. A potential edge is defined as the smallest edge from a node in the spanning tree. The smallest potential edge is chosen (and removed) from the queue and added to the spanning tree, if no cycles are being formed (checked by having a unordered set of visited node indices). Once a node(/edge) is added, the queue is updated by adding the next smallest edge from both the end-nodes of the latest edge. This way we greedily choose edges to our spanning tree to end up with a Minimum Spanning Tree.

1.1.1 Heuristic 0 - Greedy MST traversal with no repetition

This is a simple algorithm where from a randomly chosen node in the MST we traverse greedily till we reach a leaf node. Then we skip all the visited nodes and jump to the next greedy choice from a root.

1.2 RESULTS

Table 1 gives the results for the 3 problem datasets. Along with optimal length, length according to Hueristic 0 (starting with node 1), Length of MST obtained, and the time (in milliseconds) taken to calculate MST and tour is also given.

$Dataset$	$OptimalLength$	$Hueristic0$	$MSTLength$	$Timetaken$
$eil51$	426	641	376	0.002
$eil76$	538	707	472	0.003
$eil101$	629	842	542	0.007

Table 1: Results for given Problem sets

Table 2 gives the results for randomly generated problem sets. 10 each with 100, 200, 300 nodes. These sets and output files are given the drive link shared at the end.

$Dataset$	$Hueristic0$	$MSTLength$	$Timetaken$
$rnd1001$	2006.4	1358	0.004
$rnd1002$	1972	1363	0.004
$rnd1003$	2108	1398	0.005
$rnd1004$	2042	1313	0.005
$rnd1005$	2104	1342	0.008
$rnd1006$	2105	1361	0.007
$rnd1007$	2302	1453	0.005
$rnd1008$	2044	1334	0.004
$rnd1009$	2153	1380	0.005
$rnd10010$	2061	1368	0.006
$rnd2001$	4567	2897	0.016
$rnd2002$	4081	2815	0.018
$rnd2003$	4438	2855	0.015
$rnd2004$	4459	2848	0.012
$rnd2005$	4399	2876	0.019
$rnd2006$	4244	2805	0.017
$rnd2007$	4305	2798	0.014
$rnd2008$	4478	2900	0.014
$rnd2009$	4356	2890	0.013
$rnd20010$	4489	2892	0.020
$rnd3001$	7023	4635	0.024
$rnd3002$	6717	4414	0.027
$rnd3003$	7244	4749	0.038
$rnd3004$	7115	4795	0.035
$rnd3005$	6777	4491	0.047
$rnd3006$	7346	4757	0.032
$rnd3007$	7342	4696	0.050
$rnd3008$	6936	4595	0.024
$rnd3009$	7139	4598	0.025
$rnd30010$	7030	4652	0.050

Table 2: Results for randomly generated problem sets

2 Problem 2

2.1 Initial Thoughts

My approach towards solving this problem was to try and minimize path overlaps between robots. We can apply a k-means clustering algorithm based on edge lengths to classify the graph into k-clusters, and assign a robot to each cluster. We can thus apply a simple 2-approximation on each cluster to find paths for each robot.

3 NOTE

To run the code follow the README file. The videos can be accessed through this drive link. Follow this link