Jacob Redfern

2/22/2020

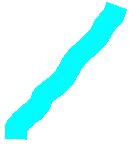
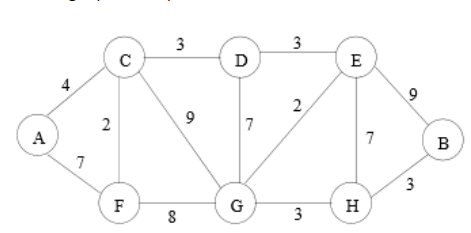
CS 325

Julianne Schutfort

Homework 5 – Essay

Problem 1:

1. I would recommend using Dijkstra’s Algorithm for finding the minimum spanning tree of the town. The algorithm would start at point G and then compare the weight of each node adjacent to G and then choose the node with the least weight. The distance from the starting node to the connected nodes will be updates with the lowest distance. The node with the least weight would then make the same calculation of comparing and choosing the adjacent node with the least weight. This pattern will repeat except for if there is already a pattern to the chosen node, then that branch will be skipped.



|  |  |  |
| --- | --- | --- |
| A | G->E->D->C->A | 12 |
| B | G->H->B | 6 |
| C | G->E->D->C | 8 |
| D | G->E->D | 5 |
| E | G->E | 2 |
| F | G->F | 8 |
| G | G | 0 |
| H | G->H | 3 |

1. def choose\_optimum(region)

paths[]

for town in region

paths <- Dijkstra(town)

return min[paths]

def dijkstra(region, weights, town)

initialize\_start(region, town)

sorted\_paths[]

un\_sorted\_paths[region]

while un\_sorted\_paths is not empty

temp <- min(un\_soted\_paths)

sorted\_paths <- S U temp

for temp\_town in temp

relax(temp, temp\_town, weights)

return(max(sorted\_paths))

def initialize\_start(region, town)

for temp\_town in region

weight[temp\_town] <- inf

posiotins[temp\_town] <- 0

weight[town] <- 0