CSC373 Tutorial 2, Summer 2015

TA: Eric Zhu

Problems are based on tutorial exercises from previous offerings of the course by Francois Pitt (Fall 2014) and Milad Eftekhar (Summar 2013).

Brief review of Minimum Spanning Tree (MST):

1. It is a "spanning tree" - acyclic connected subset of edges

2. It has the minimum sum of edge weights.

Prim's Algorithm:

- Start with some vertex $r \in V$ and at each step, add smallest-weight edge that connects a new vertex to the existing partial tree.

Kruskal's Algorithm:

- Repeatedly put in smallest-weight edge remaining, as long as it doesn't create a cycle

- Proof?

Problem 1

Prove or disprove: If e is a minimum-weight edge in connected graph G (where not all edge weights are necessarily distinct), then every minimum spanning tree of G contains e.

What if the edge weight of e is unique (but e is still has the smallest weight)?

Problem 2

Think: what is the most general proerty of an edge e, in the most general kind of input graph G, for which you can gaurantee that e either belongs (or does not belongs) to every MST of G?

Problem 3

If graph G is connected anf contains more than $n-1$ edges (where $n = V$ as usual), and if there is a unique edge e with maximum cost, then is e guaranteed **not** to be in any MST of G?

If not, what other conditions can you put on G to guarantee that e will be in no MST of G?

Problem 4

Prove or disprove: for every graph G whose edge weights are all distinct, every MST of G contains the two edges e_1, e_2 with the two smallest weights.

Problem 5

Consider the "reverse-delete" algorithm to find MST:

Algorithm 1 Reverse-delete

Sort edges E so $c(e_1) \geq ... \geq c(e_m)$

$T = E$

for $j = 1...m$ **do**

  if $T - \{e_j\}$ is connected **then**

    $T = T - \{e_j\}$

  end if

end for

return T

Prove that this algorithm always finds a MST.