Recitation 10 – *Graph Contractions*

Parallel and Sequential Data Structures and Algorithms, 15-210 (Spring 2015)

March 31st, 2015

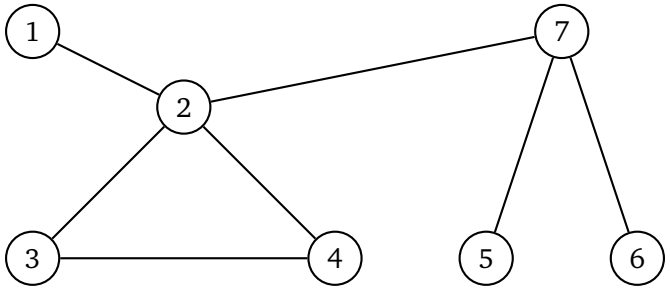
1 Announcements

- *AbridgedLab* is due tomorrow!
- *Exam II* will be on April 8, 2015. Prepare yourselves.
- *SegmentLab* will be released on Wednesday, but isn't due until after the exam. Take a look at it before the exam!

2 Star Contraction

2.1 Review

We will take a look at star contraction using an example graph:



Q: Assuming we receive $\langle(1,H),(2,T),(3,T),(4,H),(5,T),(6,T),(7,T)\rangle$ as our sequence of coin flips, how will we partition the given graph in the first round of star contraction? Assume that we break ties by choosing the center with the smaller label.

A:

Q: After each round, we merge the partitions by collapsing each partition to the center node. What does the resulting graph look like after performing this process?

A:

Q: Given the following set of coin flips for each round, complete the table of partitions for each round of star contraction.

Round	1	2	3	4	5	6	7	Partitions
1	H	T	T	H	T	T	T	$\{1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 4, 4 \mapsto 4, 5 \mapsto 5, 6 \mapsto 6, 7 \mapsto 7\}$
2	H	T	H	H	T	T	H	
3	H	H	T	T	H	T	H	
4	T	T	H	T	T	H	H	

A:

2.2 Random Graphs

We know from lecture that star contraction on a graph G will reduce the number of nodes in G by an expected constant factor each round, but will it necessarily do the same for the number of edges?

Let G be a graph on n vertices where each vertex is connected to three random neighbors.

Q: What is the expected number of edges left in G after perfoming one round of star contraction?

A:

Q: What is the expected degree of a node x after performing a round of star contraction?

A:

2.3 Contracting Trees

In the previous questions, we've considered contractions on arbitrary undirected graphs, which take expected $O(|E| \log |V|)$ work to contract. In this question, we will examine contractions on trees to see if we can get better cost bounds.

Recall the `connectedComponents` function from lecture, which has the following pseudocode:

```
1  fun connectedComponents(G = (V, E)) =
2  if |E| = 0 then (V, {v ↦ v : v ∈ V})
3  else let
4     val (V', P) = starPartition(V, E)
5     val E' = {(P[u], P[v]) : (u, v) ∈ E | P[u] ≠ P[v]}
6     val (V'', P') = connectedComponents(V', E')
7  in
8     (V'', {v ↦ P'[s] : (v ↦ s) ∈ P})
9  end
```

In lecture, we found that the work of `connectedComponents` was in $O(|E| \log |V|)$, as the number of edges didn't necessarily decrease by a constant factor at each round like the vertices.

Q: What is the relationship between the number of edges and vertices in a tree?

A:

This means the number of edges and the number of vertices decrease geometrically each round, as contracting any vertex must remove at least one edge. Therefore, the work for `connectedComponents` on a tree is $\sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i cn = O(n)$.

3 Minimum Spanning Trees

As discussed in lecture, a minimum spanning tree of a weighted, undirected graph G (directed graphs have the similar concept of an "arborescence") is a subgraph of G that is a tree that touches every vertex in G while minimizing the total weight of its edges.

One of the differences between a minimum spanning tree and many of the trees you've seen before is that MSTs are "unrooted", meaning there is no root or concept of parent/child nodes in a minimum spanning tree.

In the future, we'll discuss algorithms for finding minimum spanning trees, but during this recitation we will only consider some basic properties.

Q: How many edges does a minimum spanning tree have?

A:

Q: A graph with distinct edge weights must have a single unique minimum spanning tree. Why is this?

Hint: adding an edge to a minimum spanning tree must create a cycle.

A: