Recitation 11

Graph Contraction and MSTs

11.1 Announcements

- *SegmentLab* has been released, and is due **Monday afternoon**. It's worth 135 points.
- *Midterm 2* is on **Friday, November 11**.

11.2 Contraction

In the textbook, we presented an algorithm for counting the number of connected components in a graph:

Algorithm 11.1. *(Algorithm 17.22 in the textbook.)*

countComponents (V, E) =
  if $|E| = 0$ **then** $|V|$ **else**
  let
    (V', P) = *starPartition* (V, E)
    $E' = \{(P[u], P[v]) : (u, v) \in E \,|\, P[u] \neq P[v]\}$
  in
    countComponents (V', E')
  end

with `starPartition` implemented as follows:

Algorithm 11.2. *(Algorithm 17.15 in the textbook.)*

starPartition (V, E) =
let
  $TH = \{(u, v) \in E \,|\, \neg heads(u) \wedge heads(v)\}$
  $P = \bigcup_{(u,v) \in TH} \{u \mapsto v\}$
  $V' = V \setminus domain(P)$
  $P' = \{u \mapsto u : u \in V'\}$
in
  $(V', P' \cup P)$
end

Now, suppose we implemented star partitioning for enumerated graphs as follows:

```
val enumStarPartition : (int * int) Seq.t * int → int Seq.t
```

Specifically, given a graph represented as a sequence of edges E where every vertex is labeled $0 \leq v < n$, (`enumStarPartition` (E, n)) returns a mapping P where $P[v]$ is the super-vertex containing v. (If v was a star center or was unable to contract, then $P[v] = v$.)

Task 11.3. *Implement a function* `enumCountComponents` *which counts the number of components of an enumerated graph. It should take in a graph represented as* (E, n) *and use* `enumStarPartition` *internally.*

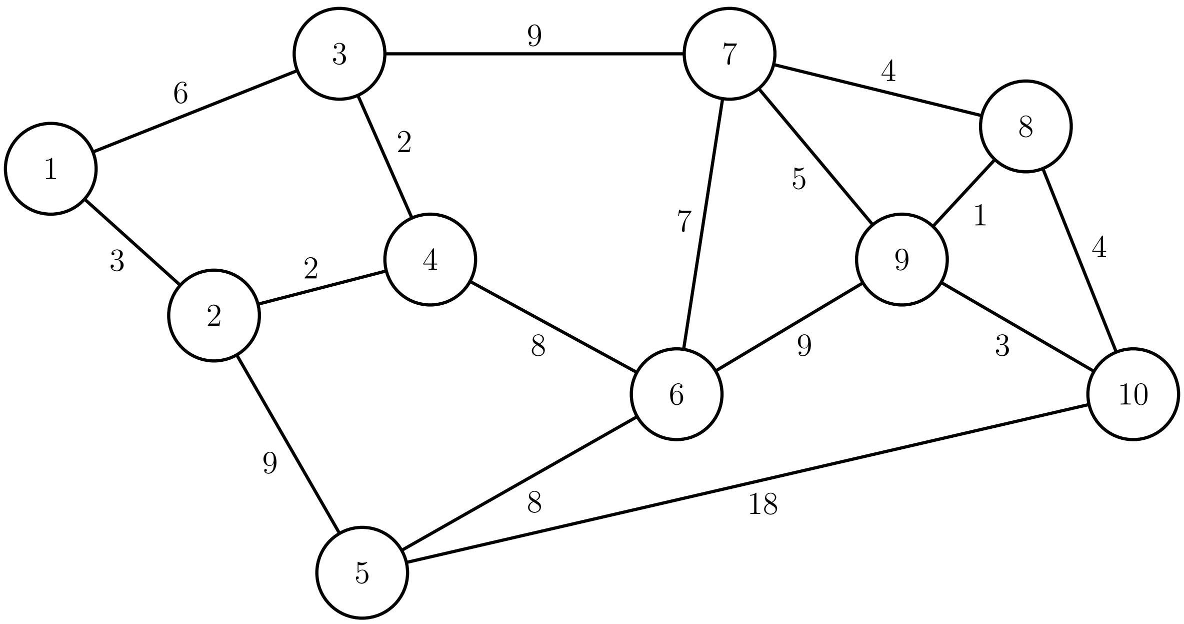
11.2.1 Cost Bounds

Task 11.4. *Recall that a **forest** is a collection of trees. What are the work and span of* `enumCountComponents` *when applied to a forest? Assume that* $($`enumStarPartition` $(E, n))$ *requires* $O(n + |E|)$ *work and* $O(\log n)$ *span.*

11.3 Borůvka's Algorithm

The textbook describes two versions of Borůvka's algorithm: one which performs tree contraction at each round, and another which performs a single round of star contraction at each round. We will be using the latter, since it has better overall span ($O(\log^2 n)$ rather than $O(\log^3 n)$).

Task 11.5. *Run Borůvka's algorithm on the following graph. Draw the graph at each round, and identify which edges are MST edges. Use the coin flips specified.*



Round	Vertices									
	1	2	3	4	5	6	7	8	9	10
0	H	T	H	T	T	H	T	H	T	T
1	T	H	T			T		T		H
2		T				H				T

11.4 Additional Exercises

Exercise 11.6. *In graph theory, an* ***independent set*** *is a set of vertices for which no two vertices are neighbors of one another. The* ***maximal independent set*** *(MIS) problem is defined as follows:*

> *For a graph* (V, E)*, find an independent set* $I \subseteq V$ *such that for all* $v \in (V \setminus I)$*,* $I \cup \{v\}$ *is not an independent set.*[a]

Design an efficient parallel algorithm based on graph contraction which solves the MIS problem.

[a] The condition that we cannot extend such an independent set I with another vertex is what makes it "maximal." There is a closely related problem called **maximum independent set** where you find the largest possible I. However, this problem turns out to be NP-hard!

.