4.2. Robust contractions

4.2.1. Definition. A subgraph $C \subseteq G$ is *contractible* iff for every pair of edges $e, f \in \delta(C)$ there exists a path in C connecting the endpoints of the edges e, f such that all edges on this path are lighter than either e or f. Here $\delta(C)$ denotes the set of edges having exactly one endpoint in C.

4.2.3. Lemma. *(Generalized contraction)*
When $C \subseteq G$ is a contractible subgraph, then $\mathrm{msf}(G) = \mathrm{msf}(C) \cup \mathrm{msf}(G/C)$.

4.2.4. Notation. When G is a weighted graph and R a subset of its edges, we will use $G \Uparrow R$ to denote an arbitrary graph obtained from G by increasing the weights of some of the edges in R. Whenever C is a subgraph of G, we will use R^C to refer to the edges of R with exactly one endpoint in C (i.e., $R^C = R \cap \delta(C)$).

4.2.5. Lemma. *(Robust contraction, Chazelle)*
Let G be a weighted graph and C its subgraph contractible in $G \Uparrow R$ for some set R of edges. Then $\mathrm{msf}(G) \subseteq \mathrm{msf}(C) \cup \mathrm{msf}((G/C) \setminus R^C) \cup R^C$.

4.2.7. Algorithm. *(Partition a graph to a collection of contractible clusters)*

Input: A graph G with an edge comparison oracle, a parameter t controlling the size of the clusters, and an accuracy parameter ε.

1. Mark all vertices as "live".
2. $\mathcal{C} \leftarrow \emptyset$, $R^{\mathcal{C}} \leftarrow \emptyset$. *(Start with an empty collection and no corrupted edges.)*
3. While there is a live vertex v_0:
4. $\quad T = \{v_0\}$. *(the tree that we currently grow)*
5. $\quad K = \emptyset$. *(edges known to be corrupted in the current iteration)*
6. $\quad$ *Create* a soft heap with accuracy ε and *Insert* the edges adjacent to v_0 into it.
7. $\quad$ While the heap is not empty and $|T| \leq t$:
8. $\quad\quad$ *DeleteMin* an edge uv from the heap, assume $u \in T$.
9. $\quad\quad$ If uv was corrupted, add it to K.
10. $\quad\quad$ If $v \in T$, drop the edge and repeat the previous two steps.
11. $\quad\quad T \leftarrow T \cup \{v\}$.
12. $\quad\quad$ If v is dead, break out of the current loop.
13. $\quad\quad$ Insert all edges incident with v to the heap.
14. $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{G[T]\}$. *(Record the cluster induced by the tree.)*
15. $\quad$ *Explode* the heap and add all remaining corrupted edges to K.
16. $\quad R^{\mathcal{C}} \leftarrow R^{\mathcal{C}} \cup K^T$. *(Record the "interesting" corrupted edges.)*
17. $\quad G \leftarrow G \setminus K^T$. *(Remove the corrupted edges from G.)*
18. $\quad$ Mark all vertices of T as "dead".

Output: The collection $\mathcal{C}$ of contractible clusters and the set $R^{\mathcal{C}}$ of corrupted edges in the neighborhood of these clusters.

4.2.8. Theorem. *(Partitioning to contractible clusters)*
Given a weighted graph G and parameters ε ($0 < \varepsilon \leq 1/2$) and t, the Partition algorithm (4.2.7) constructs a collection $\mathcal{C} = \{C_1, \ldots, C_k\}$ of clusters and a set $R^{\mathcal{C}}$ of edges such that:

1. All the clusters and the set $R^{\mathcal{C}}$ are mutually edge-disjoint.
2. Each cluster contains at most t vertices.
3. Each vertex of G is contained in at least one cluster.
4. The connected components of the union of all clusters have at least t vertices each, except perhaps for those which are equal to a connected component of $G \setminus R^{\mathcal{C}}$.
5. $|R^{\mathcal{C}}| \leq 2\varepsilon m$.
6. $\operatorname{msf}(G) \subseteq \bigcup_i \operatorname{msf}(C_i) \cup \operatorname{msf}\Big((G/\bigcup_i C_i) \setminus R^{\mathcal{C}}\Big) \cup R^{\mathcal{C}}$.
7. The algorithm runs in time $\mathcal{O}(n + m \log(1/\varepsilon))$.

4.3. Decision trees

4.3.3. Lemma. $D(m,n) \leq 4/3 \cdot n^2$.

4.3.5. Lemma. *(Construction of optimal decision trees)*
An optimal MST decision tree for a graph G on n vertices can be constructed on the Pointer Machine in time $\mathcal{O}(2^{2^{4n^2}})$.

4.3.7. Lemma. The decision tree complexity $D(m,n)$ of the MSF satisfies:

1. $D(m,n) \geq m/2$ for $m, n > 2$.
2. $D(m',n') \geq D(m,n)$ whenever $m' \geq m$ and $n' \geq n$.

4.3.8. Definition. Subgraphs $C_1, \ldots, C_k$ of a graph G are called the *compartments* of G iff they are edge-disjoint, their union is the whole graph G and $\mathrm{msf}(G) = \bigcup_i \mathrm{msf}(C_i)$ for every permutation of edge weights.

4.3.9. Lemma. The clusters $C_1, \ldots, C_k$ generated by the Partition procedure of the previous section (Algorithm 4.2.7) are compartments of the graph $H = \bigcup_i C_i$.

4.3.10. Lemma. Let $C_1, \ldots, C_k$ be compartments of a graph G. Then there exists an optimal MSF decision tree for G that does not compare edges of distinct compartments.

4.3.11. Lemma. Let $C_1, \ldots, C_k$ be compartments of a graph G. Then $D(G) = \sum_i D(C_i)$.

4.3.12. Corollary. If $C_1, \ldots, C_k$ are the clusters generated by the Partition procedure (Algorithm 4.2.7), then $D(\bigcup_i C_i) = \sum_i D(C_i)$.

4.3.13. Corollary. $2D(m,n) \leq D(2m,2n)$ for every m, n.

4.4. An optimal algorithm

4.4.1. Algorithm. *(Optimal MST algorithm, Pettie and Ramachandran)*

Input: A connected graph G with an edge comparison oracle.

1. If G has no edges, return an empty tree.
2. $t \leftarrow \lfloor \log^{(3)} n \rfloor$. *(the size of clusters)*
3. Call *Partition* (4.2.7) on G and t with $\varepsilon = 1/8$. It returns a collection $\mathcal{C} = \{C_1, \ldots, C_k\}$ of clusters and a set $R^{\mathcal{C}}$ of corrupted edges.
4. $F_i \leftarrow \mathrm{mst}(C_i)$ for all i, obtained using optimal decision trees.
5. $G_A \leftarrow (G/\bigcup_i C_i) \setminus R^{\mathcal{C}}$. *(the contracted graph)*
6. $F_A \leftarrow \mathrm{msf}(G_A)$ calculated by the Fredman-Tarjan algorithm.
7. $G_B \leftarrow \bigcup_i F_i \cup F_A \cup R^{\mathcal{C}}$. *(combine subtrees with corrupted edges)*
8. Run two Borůvka steps on G_B, getting a contracted graph G_C and a set F_B of MST edges.
9. $F_C \leftarrow \mathrm{mst}(G_C)$ obtained by a recursive call to this algorithm.
10. Return $F_B \cup F_C$.

Output: The minimum spanning tree of G.

4.4.2. Lemma. The time complexity $T(m, n)$ of the Optimal algorithm satisfies the following recurrence:

$$T(m, n) \leq \sum_i c_1 D(C_i) + T(m/2, n/4) + c_2 m,$$

where c_1 and c_2 are some positive constants and D is the decision tree complexity from the previous section.

4.4.4. Theorem. *(Optimality of the Optimal algorithm)*
The time complexity of the Optimal MST algorithm 4.4.1 is $\Theta(D(m, n))$.

Proof. We will prove by induction that $T(m, n) \leq cD(m, n)$ for some $c > 0$. The base case is trivial, for the induction step we will expand on the previous lemma:

$$\begin{aligned}
T(m,n) &\leq \sum_i c_1 D(C_i) + T(m/2, n/4) + c_2 m && \text{(Lemma 4.4.2)}\\
&\leq c_1 D(\textstyle\bigcup_i C_i) + T(m/2, n/4) + c_2 m && \text{(Corollary 4.3.12)}\\
&\leq c_1 D(m,n) + T(m/2, n/4) + c_2 m && \text{(definition of } D(m,n))\\
&\leq c_1 D(m,n) + cD(m/2, n/4) + c_2 m && \text{(induction hypothesis)}\\
&\leq c_1 D(m,n) + c/2 \cdot D(m, n/2) + c_2 m && \text{(Corollary 4.3.13)}\\
&\leq c_1 D(m,n) + c/2 \cdot D(m,n) + 2c_2 D(m,n) && \text{(Lemma 4.3.7)}\\
&\leq (c_1 + c/2 + 2c_2) \cdot D(m,n)\\
&\leq cD(m,n). && \text{(by setting } c = 2c_1 + 4c_2)
\end{aligned}$$

The other inequality is obvious as $D(m, n)$ is an asymptotic lower bound on the time complexity of every comparison-based algorithm. ♠