

Logistics

Website: cs.cmu.edu/~15-850 (will be updated)

Piazza: piazza.com/cmu/spring2024/cs15850

Gradescope: TBD

HWØ on website, due 1 week from now, no collaboration
(HWØ only)

Jason Li

Emin Berker

GHC 7203

GHC 6207

Office Hours TBD

TBD

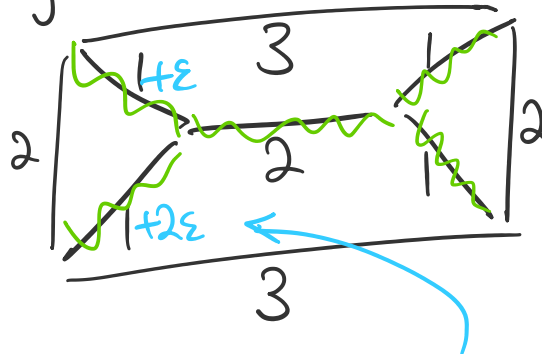
Minimum Spanning Trees

- Recap MST algorithms: Kruskal, Prim, Boruvka
- MST structure: cut rule & cycle rule
- Karger-Klein-Tarjan $O(m)$ time randomized (faster than sorting!)

MST:

Input: undirected, weighted, connected graph $G=(V,E)$

Output: spanning tree of G with minimum total weight.

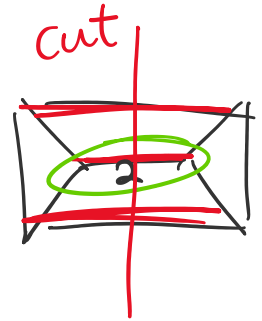


no edge weights are distinct. (Break ties arbitrarily.)

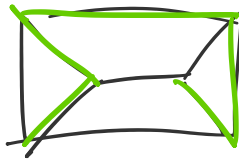
Assume all edge weights are distinct. (Break ties arbitrarily.)

MST structure

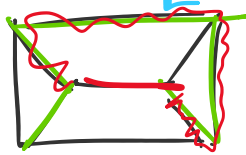
① Cut Rule: For any cut in the graph, the min-weight edge crossing the cut must be in the MST.



Proof: Suppose not:



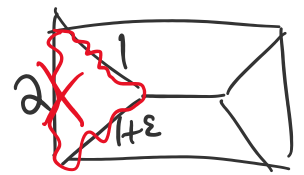
add min-weight edge, get a cycle.



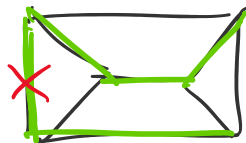
Cycle crosses cut in another edge.

Remove it, get smaller MST, contradiction

② Cycle Rule: For any cycle in the graph, the heaviest edge on that cycle cannot be in the MST.



Proof: suppose not:



remove edge, get two components of spanning tree
→ a cut.

cycle crosses cut in another edge.

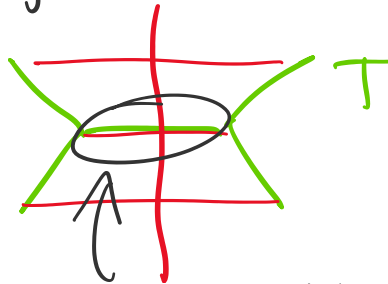
Add it, get smaller MST, ~~≠~~.

Kruskal's Algorithm

- Sort edges from lightest to heaviest: $e_1, e_2, \dots, e_m$.
- Initialize $T = \text{empty graph}$
- For $i = 1, 2, \dots, m$:
 - Add e_i to T if no cycle created

Claim: T is the MST.

Proof: Apply cut rule to each edge in T :



must be min-weight edge, o/w would have added other edge

Remark: All we needed was cut rule of MST.

This abstraction allows generalization to matroids (HW 1)

Implementation: Sort edges in $O(m \log n)$ time
Union-find data structure: $\alpha(n)$ time per edge
 $\Rightarrow m \alpha(n)$ total

Prim's Algorithm

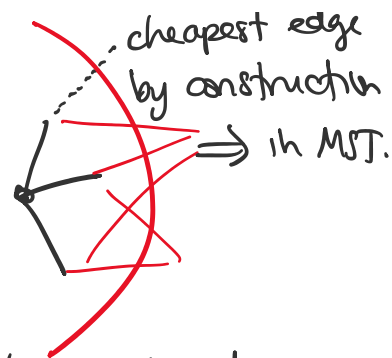
- start with arbitrary root vertex r
- add cheapest edge "out" of current tree

Claim: T is the MST.

~~cheapest edge~~
by construction

Claim: T is the MST.

Proof: Apply cut rule on each new edge:



Implementation: priority queue to maintain cheapest edge
 $O(\log n)$ time per update/query
 $\Rightarrow O(m \log n)$ total.

Remark: Prim's is "local": suppose only want local region of MST. Prim's only explores locally $\Rightarrow$ "sublinear" algorithm

Boruvka's Algorithm

- Start with empty graph: each vertex in its own component.
- In parallel, each component selects cheapest edge out
- Add all selected edges

Claim: T is the MST

Proof: apply cut rule on each selected edge

Claim: #connected components halves per iteration.

Total time: $O(m)$ for $\log_2 n$ iters $\rightarrow O(m \log n)$.

Remark: parallel algo

Karger-Klein-Tarjan Algorithm $KKT(G=(V,E))$

Karger-Klein-Tarjan Algorithm $KKT(G=(V,E))$

① ??? (Later)

② $E_1 \subseteq E$: randomly sample each edge in E w.p $1/2$

③ $F_1 = KKT(G_1=(V,E_1))$

④ Using F_1 and cycle rule, eliminate edges in E ,
let $E_2 \subseteq E$ be ~~left over~~ F_1 -light F_1 -heavy

⑤ $F_2 = KKT(G_2=(V,E_2))$

⑥ Return F_2

Def: Let F be a forest, let e be an edge.

e is F -heavy if ① $F \cup \{e\}$ has a cycle

② e is the heaviest edge on that cycle.

Otherwise, e is F -light.

Claim: Let $F \subseteq G$ be a forest. If e is F -heavy, then
 e is not in MST of G .

Proof: cycle rule.

Claim: $KKT(G)$ returns MST of G .

Proof: only "dangerous" step is eliminating edges. But we
only eliminate F_1 -heavy edges which do not belong
to MST of G .

Claim: $\mathbb{E}[|E_1|] \leq \frac{1}{2}|E|$. (easy)

Claim: $\mathbb{E}[|E_2|] \leq 2(|V|-1)$.

... $\therefore$ F_1 -light edges.

Claim: $|E| |E_2| \leq 2(|V|-1)$.

Proof: Recall that E_2 is the F_1 -light edges.

F_1 is constructed by first sampling edges, then computing MST on sample.

Same as running any MST algo (say Kruskal) and sampling the edges as you read them.

Sort edges in E by weight. Initialize $F_1 = \emptyset$.

For each edge: if e connects two components in F_1 , flip a coin. If heads (e is sampled), add e to F_1 .

Fix current F_1 . Expected 2 coin flips until add next edge. In total, expect $\leq 2(|V|-1)$ coin flips.

Finally, observe that F_1 -light edges are precisely the coin-flip edges. Total $\leq 2(|V|-1)$.

Claim: Assume ④ in $O(|E|)$ time. Then $KKT(G)$ runs in $O(|E|)$ time.

Proof: Suppose ④ in $c|E|$ time.

Let $T(m)$ be expected running time on m edges.

To show: $T(m) \leq 3c|E|$.

$$T(m) = \mathbb{E} [T(|E_1|) + T(|E_2|)]$$

$$\begin{aligned}
T(m) &= \mathbb{E}_{E_1} [T(|E_1|) + T(|E_2|)] \\
&\stackrel{\text{induction}}{\leq} \mathbb{E}_{E_1} [3c|E_1| + 3c|E_2|] \\
&\leq 3c \cdot \frac{1}{2}|E| + 3c \cdot 2 \underbrace{(|V|-1)}_{\leq |E|} \\
&\leq 7.5c|E| \dots
\end{aligned}$$

① Do 3 steps of Boruvka to reduce $|V| \rightarrow |V|/8$ in expectation.

$$\begin{aligned}
&\leq 3c \cdot \frac{1}{2}|E| + 3c \cdot 2 \underbrace{\left(\frac{|V|}{8} - 1\right)}_{\leq |E|/8} \\
&\leq 3c|E|.
\end{aligned}$$

Remark: F_1 was a cheap "approximation" to MST: not the MST, but enough to eliminate a lot of (F_1 -heavy) edges to make progress.

Theme: use approximation as a proxy to speed up graph algorithms

Implementing ④:

Thm (MST Verification): Given a tree $T=(V,E)$ and m pairs of vertices (u_i, v_i) , output heaviest edge on the path $u_i \rightarrow v_i$ in T for each i

output heaviest edge on
(unique) $u_i - v_i$ path in T for each i
in $O(m + |V|)$ time.