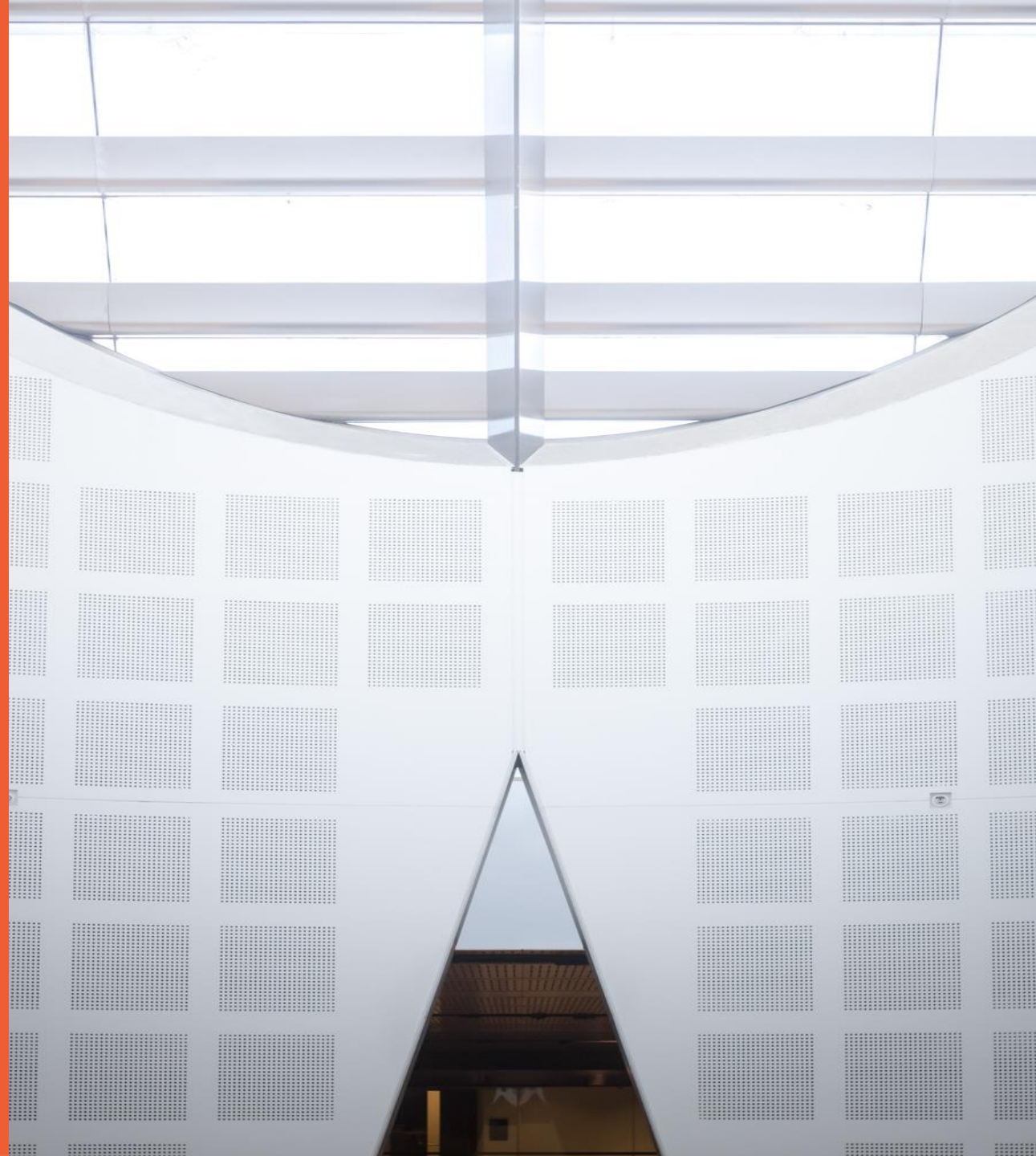
COMPx270: Randomised and Advanced Algorithms
Lecture 5 (Extra): MST in Expected Linear Time

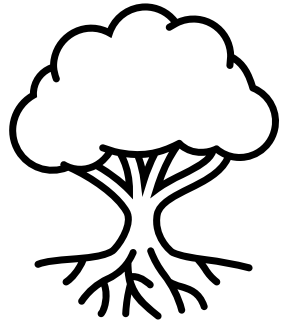
Clément Canonne

School of Computer Science

THE UNIVERSITY OF SYDNEY



And now, for something else completely different!



Minimum Spanning Tree in (Expected) Linear Time: Karger-Klein-Tarjan.

We are given a weighted, connected graph $G=(V,E)$ on n vertices and m edges, want to compute an MST.

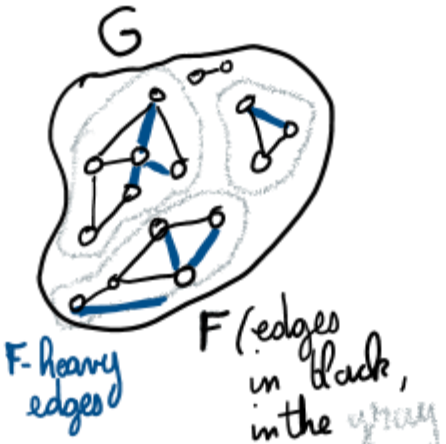
Simplification: assume MST is unique

Generalisation: drop the "connected" part, compute (the) Minimum Spanning Forest (MSF)

(Why? Will make recursion on subgraphs easier)

Key idea #1: this takes time because G has many edges. If we could sparsify it (remove a bunch of edges) while preserving the MSF, we'd be good!

Key idea #2 Can prove (similar to cycle property in Kruskal's analysis) that if we have any forest F of a graph G, then the edges heavy (in G, relative to F) cannot be part of an MSF of G.

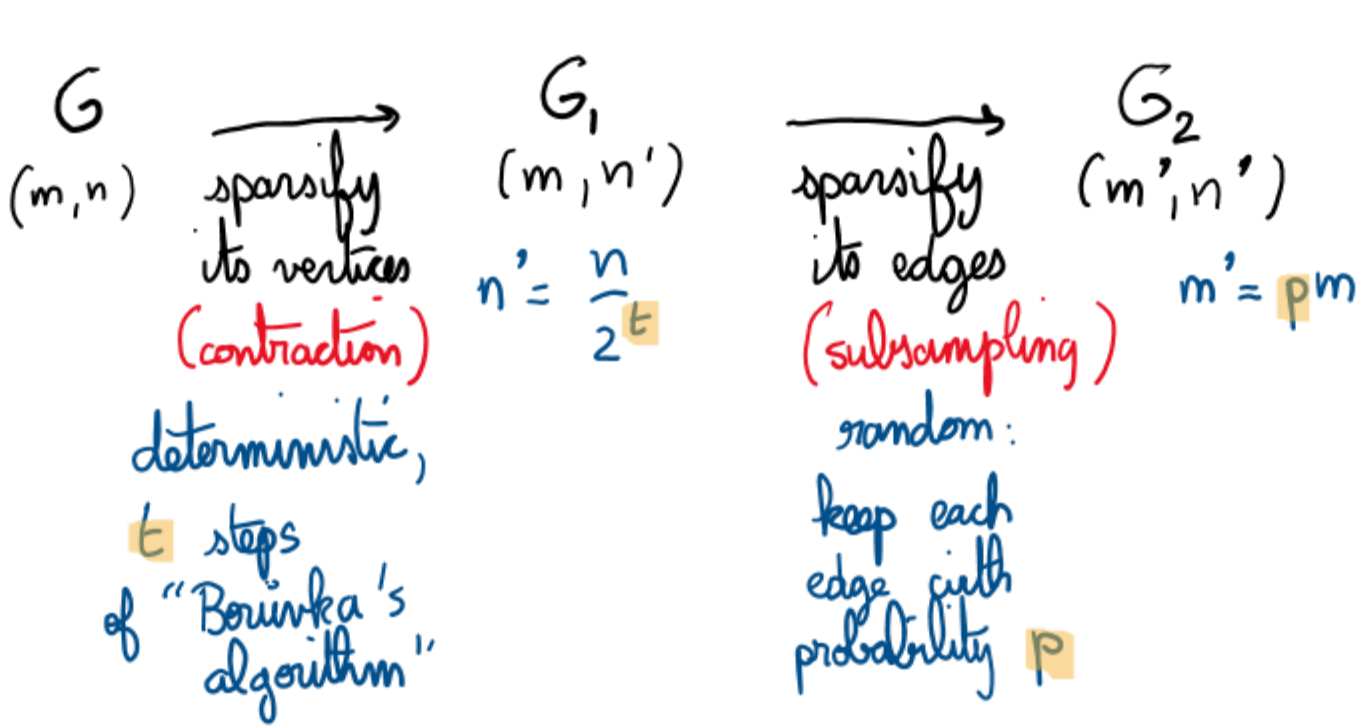


Key ~~idea~~ fact: given a forest F of G, can efficiently find all such F-heavy edges. (Not obvious, but it's known)

So if we could find (efficiently) any forest $\tilde{F}$ of G (not necessarily an MSF) with a lot of $\tilde{F}$-heavy edges in G, then we could remove all these heavy edges efficiently, and only have to find the MSF in whatever remains of G!

But how do we find such a good forest $\tilde{F}$? Seems circular.

Key idea #3: sampling as a guide.



(don't worry about details: just remember one step reduces # of vertices by a factor at least 2)

+ recursive call to find MSF of this smaller G_2 → get forest $\tilde{F}$

★ can show that $\tilde{F}$, being MSF of G_2, has (in expectation) many heavy edges in G_1

→ at least $m - \frac{n'}{p}$

So now we have this "guide" $\tilde{F}$ which will help sparsify G_1, which then will allow us to efficiently find an MSF of G_1 (since we'll only have to look at whatever edges of G_1 remain after the sparsification)

① Given $\tilde{F}$, find all $\tilde{F}$-heavy edges in G_1
(we said ⓐ this can be done in linear time, and ⓑ there are many of them)

② Remove all these heavy edges from G_1
(that's OK, since by that cycle property-type thing, we know none of these heavy edges can be part of the MSF of G_1)

③ Now we only have $\approx \frac{n'}{p} = \frac{n}{p^{2^t}}$ edges left in G_1, and n' vertices.
Much smaller graph! We can afford to compute its MSF now! (recursively)

Important detail:
Finding MSF of G_1
⇕
Finding MSF of original G (by adding edges contracted during Borůvka steps

Final touch: time complexity, and how to set the two parameters $t \geq 1$, $p \in (0,1)$?

Solve recurrence by substitution (we want

$$\mathbb{E}[T] \leq C'(n+m)$$

for some $C' > 0$)

and see what t, p need to be for proof to go through. E.g., $t=3$, $p=\frac{1}{2}$ work.

(also $t=2$, $p=$ sthg... for instance)

Success

Summary:

1. Sparsify G into much smaller G_2 to compute guide MSF (of G_2) $\tilde{F}$
   (both wrt vertices (deterministic contraction) and edges (random subsampling))
   ↑ efficient since G_2 is small
2. Use $\tilde{F}$ as guide to remove many edges from G (call resulting graph G_3)
   ↑ they cannot be in the MSF of G anyway
3. Compute (recursively) the MSF of the resulting graph G_3
   ↑ same MSF as G, but can be computed efficiently since G_3 is small