Random Spanning trees & MST verification (Lecture #2).

Last time saw algorithms that all use the cut rule + fancy data structures to get improvements on $m \log n$ $\longrightarrow$ down to $O(m + \log n)$ and $m \log^* n$.

This time: (randomized) $O(m+n)$ time. Two ingredients

(a) sampling and using the cycle rule to discard a lot of edges. (sparsify)

(b) MST verification

Light & Heavy: given a tree T (or a forest F) an edge e is called F-light if $MSF(F \cup \{e\})$ contains e.

An edge is F-heavy otherwise.

Fact: if T is a MST of G then e is T-light $\Leftrightarrow$ $e \in T$. $\forall e \in E(G)$.

Magic Black box: given $G = (V, E)$ and a spanning tree T $= (V, E_T \subseteq E)$, there exists an algorithm Verify(G, T) that returns a list of all T-light edges of E. And runs in time $O(m+n)$.

N.B. if the list of T-light edges is just E_T then T is a MST of G. So this, in linear time verifies if T is a MST of G.

Naively: would need to check if $\forall e \in E \setminus E_T$ it is the heaviest edge on its fundamental cycle.

(2)

Karger's Algorithm (V, E) $\quad |V| = n, \ |E| = m. \quad m \geq n.$

• ~~Let H~~ be$_1$ a random subset ~~of edges of size of E w.p. 1/2~~. the graph induced on

(1) Boruvka on G ~~twice~~ (3 times), clean up to get $G' = (V', E')$.

\# vertices is $\leq n/8$.

(2) $E_1 \leftarrow$ random subset of E' (with sampling probability $\frac{1}{2}$).
+ (some fixed spanning tree edges) drop.

(3) $T_1 \leftarrow \text{Karger}(V', E_1)$

(4) discard all T_1-heavy edges from E'. (call this E_2) remainder — all T_1 light edges of E'.

(5) return $\text{Karger}(V', E_2)$

Fact 1: $\mathbb{E}[\#\text{edges in } E_1] \leq m/2 \ \boxed{+(n'-1)}$ drop.

$\begin{cases} n' = \#V' \leq n/8 \\ m' = \#E' \leq m. \end{cases}$

Fact 2: $\mathbb{E}[\#\text{edges in } E_2] \leq 2n'$.

Say: all rest of work in steps ①, ②, ④ $\leq m$.

Claim: Let $T(m,n)$ be expected runtime on every possible graph on n nodes m edges.
then $T(m,n) \leq 2m+n$.

Pf:
$$T(m,n) \leq \mathbb{E}[T(|E_1|, |V'|)] + \mathbb{E}[T(|E_2|, |V'|)] + m.$$
$$\leq \mathbb{E}[2|E_1|] + n' + \mathbb{E}[2|E_2|] + n' + m$$
$$\leq 2\left[\tfrac{m}{2} + n'\right] + n' + 2[2n'] + n' + m$$
$$= 2m + 8n' \leq 2m+n.$$
∎

Proofs of Facts:

Fact 1: easy. just uses that each $e \in E_1$ w.p. $\frac{1}{2}$. [~~used in addition~~ ~~fixed sp tree~~]

Fact 2: two different proofs depending on how E_1 is chosen.

Pf 1 [Karger]. E_1 is sampled by picking each edge in E' w.p. $\frac{1}{2}$ independently.

Recall: want to bound # of T_1-light edges, $T_1 \leftarrow MST$ on E_1.

~~⇒~~ Build T_1 using Kruskal this way. Sort edges of E' and look at edges in it.

When looking at $e \in E'$,

(i) if e creates a cycle with current forest F, ignore it. → even if $e \in E_1$ it will not be T_1-light.

(ii) if not, flip a coin for e. → if tails, ~~it will~~ not in E_1, it will be T_1-light later

→ if heads, in E_1 and in T_1. ⇒ also T_1 ~~-~~-light, but in T_1.

⇒ every time we are in case (ii) we make 1 T_1-light edge.

But w.p. $\frac{1}{2}$ we add an edge to T_1 ⇒ E[# times before T_1 has $n'-1$ edges]

$= E[\#\text{flips before see } (n'-1) \text{ heads}] \le 2(n'-1)$.

QED.

Pf 2 [Chan]. E_1 is a random subset of ~~2~~ $m'/2 = r$ edges of E'. plus a ~~(sampled with replacement)~~

~~Claim: $\forall e \in E' \setminus E_1$, e is T_1-light w.p. $\le 2n'/m'$. [⇒ linearity of exp.]~~

~~Pf: T_1 is MST of E_1.~~

~~if e is T_1-light then either $e \in MST(T_1 \cup \{e\})$.~~

Claim: Pick a random edge e of E' (indep of E_1). $\Pr_{e,E_1}[e \text{ is } MST(E_1)\text{-light}] \le n'/r$

$$= \Pr_{e,E_1}[e \text{ in } MST(E_1 \cup e)]$$

~~condition on random choice of E_1~~.

$= \Pr[\text{one random edge in } MST(r+1) \text{ random edges}]$

but $MST(E_1 \cup e)$ has $\le n'-1$ edges. $\Pr[e \text{ in this MST}] \le \frac{n-1}{r}$.

Next: How to prove the Magic Black Box? MST verification.

(*) Given T and pairs $(u_1,v_1),(u_2,v_2) \dots (u_m,v_m)$. return the ($w_i$) weight of the heaviest edge on the path $T[u_i,v_i]$. in time $O(m+n)$.

If do this, can solve the MST verification problem.

History:

'79: Tarjan showed how to do it with $O(m\alpha(m,n))$ time

~~First idea~~: '85: Komlos showed how to do it using $O(m+n)$ comparisons, but not clear how to find out what comparisons to make.

'97 Dixon Rauch Tarjan on RAM machine, 98: Buchsbaum Kaplan Rogers on pointer m/c.

97: King also on RAM machine.

09: Hagerup simpler.

Simplifications: given any tree, can assume that v_i is an ancestor of u_i.

(I) Idea: find, for each (u_i,v_i), the LCA $l_i = lca(u_i,v_i)$.

Create pairs (u_i,l_i) and (v_i,l_i) for each original (u_i,v_i).

Now given a sol^n for new instance, just return $w_i \leftarrow \min\{w_i', w_i''\}$

How long to find LCAs of m pairs?

Thm [Harel & Tarjan '84]: Can preprocess a tree T in $O(n)$ time so that can answer LCA queries in $O(1)$ time.

Actually simpler: given T and all the pairs up front can answer in time $O(n+m)$. ← easier problem.

May come back at end to give a simple $O(m\alpha(m,n)+n)$ time algorithm using union-find.

⑤

OK: do Tree T (n), pairs u_i, v_i (m) with v_i ancestor of u_i. Return $w_i \leftarrow$ max wt edge on $T[u_i, v_i]$.

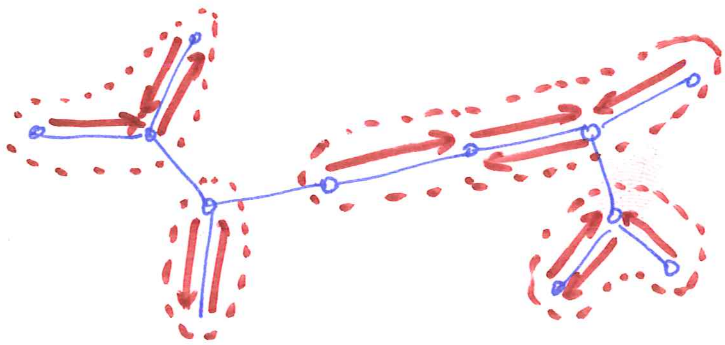
Simplification #2: T is a fully branching tree.

A fully-branching tree T ~~tree~~ is a rooted leveled tree with

(a) all leaves on same level d.

(b) each internal node has at least 2 children.

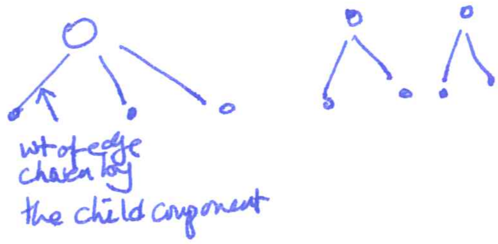
Claim: take T, run Boruvka on it. this gives a nice ~~tree~~ laminar structure which defines a tree T'



Leaves = vertices of T

level one: components after 1 round etc.

• height of $T' \le \log_2 n$.



• Fact:

max wt edge on $T'[u_i, v_i]$

= max wt edge on $T[u_i, v_i]$. [HW #1]

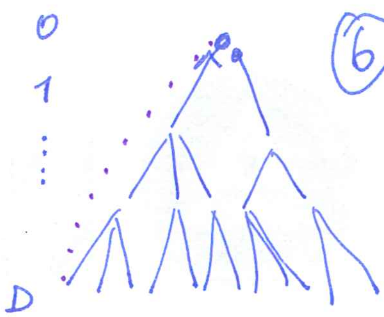
OK: so do simplification #2, and then #1. gives

a fully branching tree T and pairs (u_i, v_i) st v_i is an ~~parent~~ ancestor of u_i, and u_i is a leaf (by simplification #2).

How to solve max queries now!

depth $\overset{\text{root}}{0}, 1, 2, \ldots, D \le \log_2 n$.

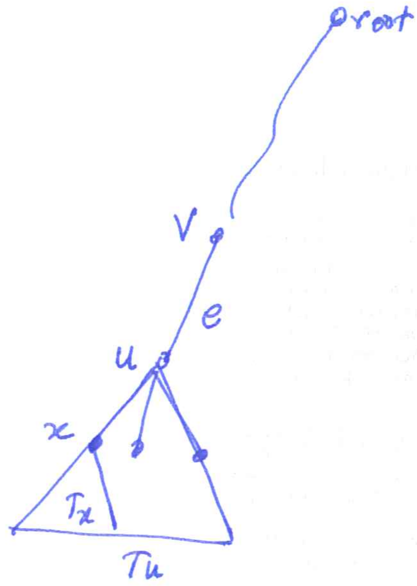
$\boxed{\text{Komlos: just count \# of comparisons for now.}}$



for each edge $e_0 = (u, p(u) = v)$, look at all queries starting in T_u and ending above v.

Say they go to $v_1, v_2, \ldots v_k$ st

$$d(v_1) \le d(v_2) \quad \cdots \quad < d(v_k).$$



And we have a "query string"

$$Q_u = (d_1, d_2, \ldots, d_k).$$

For which we have answer string

$$A_u = (a_1, a_2 \ldots a_k).$$

$a_i =$ max wt edge between u and v_i

$=$ ancestor of u at depth d_i

$\boxed{\text{Crucial Fact: } a_1 \ge a_2 \ge \cdots \ge a_k.}$

Now: want to extend this to Q_x and get A_x.

$Q_x \to$ queries starting in T_x and ending above u.

Note: $Q_x \subseteq Q_u \cup \text{depth}(u)$.

So we ~~can~~ know that A_u consists of

$$\max(a_1, w(x,u)), \max(a_2, w(x,u)) \ldots \max(a_k, w(x,u))..$$

Can use binary search to find the right place after which it is all $w(x,u)$.

$$\# \text{comps} = \lceil \log_2(|Q_u| + 1) \rceil \approx \cancel{\log_2 |Q_u|}$$

$$\Rightarrow \text{total \# comps} \le \sum_u \log_2(|Q_u| + 1) + n.$$

$\uparrow$ for roundup

Claim: $\overset{!}{\le} O\left(n \log \frac{m+n}{n}\right) \le O(m)$.

Pf: let n_i guys at ~~level~~ ~~depth~~ level i (leaves at level 0).

$$\sum_{u \in \cancel{\text{depth } i}\ \text{level}} \log(1+|Q_u|) = n_i \text{ Average}(\log_2(1+|Q_u|))$$

$$\leq n_i \cdot \log_2\left(1+\frac{\sum'_{u \in \text{level } i} |Q_u|}{n_i}\right) \quad \text{Jensen's ineq}$$

$$= n_i \log_2\left(1+\frac{m}{n_i}\right) \leq n_i \log_2\left(\frac{n+m}{n_i}\right)$$

$$\Rightarrow \sum'_{u \in \text{all}} \log(1+|Q_u|) \leq n \log\left(\frac{n+m}{n}\right) + \sum'_i n_i \log(n/n_i)$$

$\boxed{n_i \leq n/2^i \text{ and } x\log(n/x) \text{ is increasing for } x \leq n/4.}$ $\Rightarrow \sum'_i n_i \log(n/n_i) \leq n_0 \log n/n_0 + n_1 \log n/n_1 + \sum'_{i \geq 2} n_i \log(n/n_i)$

$$\leq n_0 + n/2^i \cdot i$$

$$= O(n).$$

$$\Rightarrow \text{total}: \quad O\left(n + n \log_2\left(\frac{m+n}{n}\right)\right).$$

Remark: It is not surprising that $\exists$ a proof that ~~uses~~ can be verified in $O(m)$ comps proving T^* is MST. [Just give sorted list of edges. etc.]

But that this proof can be found using $O(m)$ comps. And in $O(m)$ time.

Rest of analysis: How to implement all of this in $O(m)$ time and not just using $O(m)$ comparisons. ~~Need to do table lookups and store so that~~

High level idea: store the Q_u as a $\log_2 n$ bit word.

see Zwick's notes

store answer also as a set of nodes: a_k is nodes st. the heaviest edge on (u, q_k) is one from a_k to $p(a_k)$

now carefully do the same operations, but ~~store~~ instead of having to do binary search explicitly, store the solutions on $\frac{\log n}{10}$ bitstrings and use them.

Wrap up: MST verification in $O(m+n)$ time.

Saw details except (a) LCA and (b) Lemma from HW1.

$\Rightarrow$ Randomized Algo in $O(m+n)$ time.

Can we make this deterministic? [Can we get an $O(m+n)$ comparison decision tree out of this?]

— x —

Postscript: LCAs in $O(m\,\alpha(m,n))$ time? At least when all queries are given up front. "offline"
List L of queries.

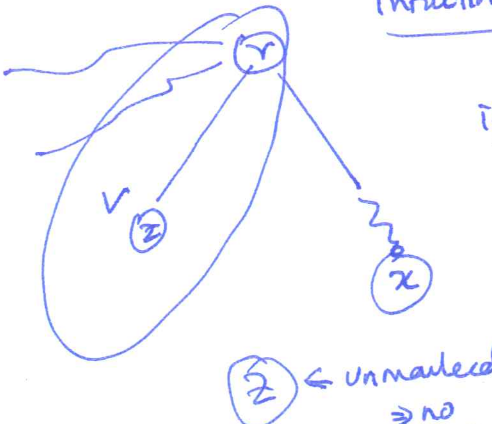
[Tarjan's '79 paper]

LCA(x)

- makeset(x)
- for all children y of x.
  - LCA(y)
  - union(x, y).
  - ~~memo~~ "head" of find(x) $\leftarrow x$. // say that all guys in this set have head x.
    find(y)
- x.marked

  $\forall z$ st $(x,z) \in$ List
  if z marked then LCA(x, z) = head of find(z).

Intuition/Proof:



if z marked then z has been explored and we're in another child of the LCA($=r$).

$\Rightarrow$ find(z).head $= r$.