MST VERIFICATION

Rajendra "Raz" Magar
magar@cs.dartmouth.edu

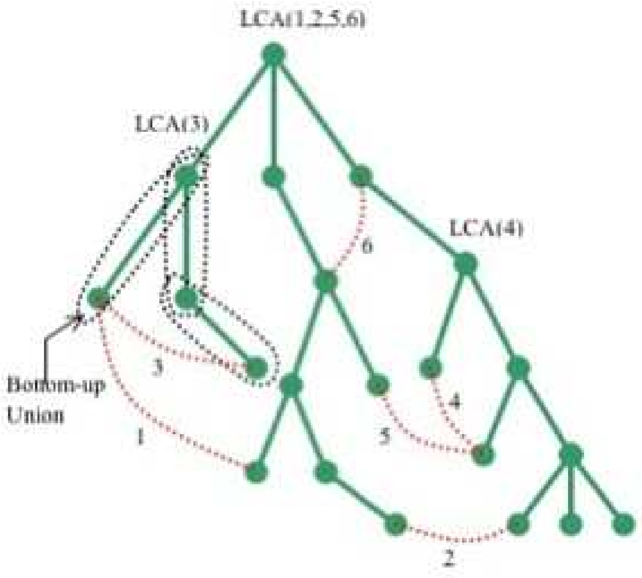
Given a connected and weighted graph $G(V, E)$ and tree $T \subseteq E$, verify whether T is MST of G.

1 Introduction

Basic idea: $\forall\, e \in E - T$, check if e is T-heavy. If any edge is not T-heavy, then $T \neq$ MST.

1998 : MST Verification in linear time on a pointer machine (no arithmetic allowed with memory addresses) was finally solved.

We will see RAM algorithm, instead.



LCA of non-tree edges

Root T arbitrarily.

First compute Least Common Ancestors (LCA's).

COMPUTE LCA'S:

for each $\{u, v\} \in E - T$:
    compute $l = \texttt{LCA}(u, v)$
    and put $\{u, v\}$ into a bucket at l

Now, to determine T-heavy/light, we need "only" compute max. weight along downward path.

2 LCA and Disjoint Set Union

Maintain n elements under two operations.

$\texttt{UNION}(A, B, C) : C \leftarrow A \cup B$ (destroys A and B)
$\texttt{FIND}(x)$: returns A s.t. $x \in A$

To do $n-1$ unions and m finds, it takes $O(m\alpha(m, n) + n)$ time, where α is the Inverse Ackermann function (very very slow function, slower than $\log^*$). Theoretically, α is non-linear (still goes to ∞).

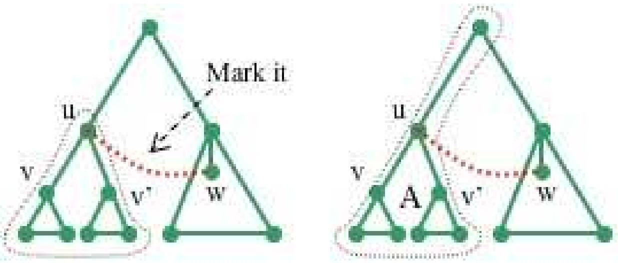
Start at arbitrary root, and do DFS to do unions.

If unions are bottom-up and the tree has l leaves, the uniting of all vertices in the tree takes $O(m\alpha(m, l) + n)$ time.

$\boxed{\text{LCA Algorithm}}$: Given a rooted tree T, a set P of pairs of $\{u, v\} \in V$ and $|P| = m$

Goal: Find $\texttt{LCA}(u, v)$ for each u, v. Have a tag bit for each $\{u, v\} \in P$.

SEARCH(u):
- **for each** child v of u:
  - SEARCH(v)
  - $\texttt{UNION}(v, u, u)$
- **for each** $\{u, w\} \in P$:
  - **if** $\{u, w\}$ is not marked
    - **then** : mark it.
    - **else** : $\texttt{LCA}(u, w) = \texttt{FIND}(w)$



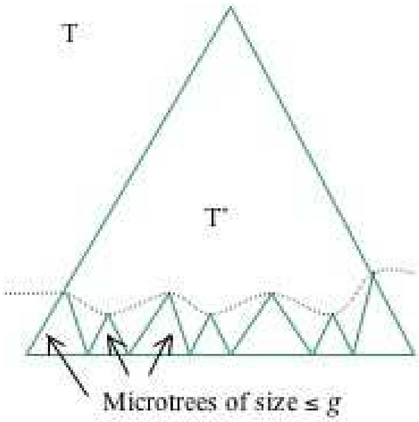
1. unmarked
2. found marked during SEARCH(w) so, LCA(w,u) = find(u) = A

Running time: $O(m\alpha(m, l) + n)$

Reduce l. If $l = m/\log^{\varepsilon} n$ for some $\varepsilon > 0$, then $\alpha(m, l) = O(1)$

So the new running time is $O(m + n)$.

3 Microtree Technique



Each subtree has size $\leq g$. Subtrees cover all the leaves.

Leaves, $l(T') \leq n/g$

If $g = \log^{\varepsilon} n$, then algorithm on T' runs in linear time.

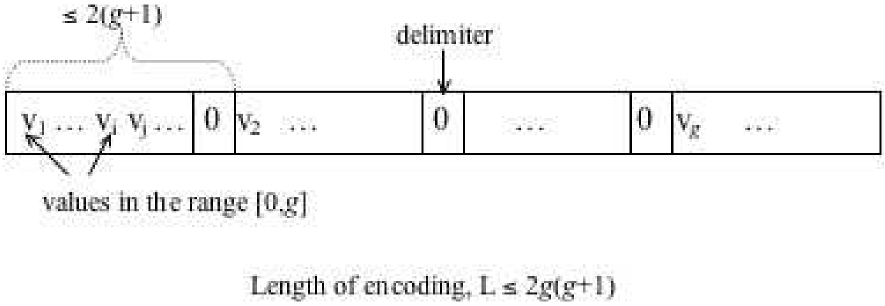
Microtrees are so small $(g = \log^{\frac{1}{3}} n)$ that even the most horrible algorithm for LCA will run in $O(n)$ time on <u>all</u> of them.

Work can be shared among duplicate micro-trees.

3.1 Adjacency List Encodings and Pointer-based Radix Sort

DFS will assign numbers in $[0, g]$ to the vertices.

<u>possible encodings</u>: $O(g^{g^2})$



Get the encodings for all microtrees and use radix sort (lexicographical sort).

There are $g + 1$ buckets, for all vertex numbers + the delimiter zero.

Running time: $O(Lg + \sum_i l_i)$, where l_i is the length of the ith microtree encoding and $L = \texttt{max}_i(l_i)$.

$L = O(g^2)$

$\sum_i l_i = O(n)$, where n is the size of the entire tree.

Running time: $O(g^3 + n)$

Process the sorted list.

For each distinct encoding, we perform computation P (say), which takes time $T(g)$.

So, total time $= O(g^3 + n + g^{g^2} T(g))$

Choosing $g = O(\log^{\frac{1}{3}} n)$ suffices even if $T(g) = O(g^{g^2})$.

So, New running time $= O(n)$.

4 MST Verification

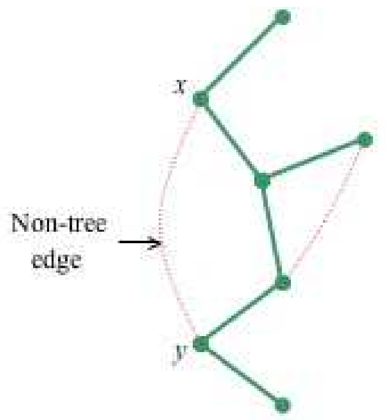
Let $G(V, E)$ be a $\boxed{\text{weighted graph}}$

$c(x, y) =$ weight of edge $\{x, y\}$

$T = (V, E_T)$: a spanning tree.

Goal: Determine whether T is an MST of G.

$\forall \{x, y\} \in E \backslash E_T$ (non-tree edges), $\exists$ a unique path $P(x, y)$ between x and y in T.



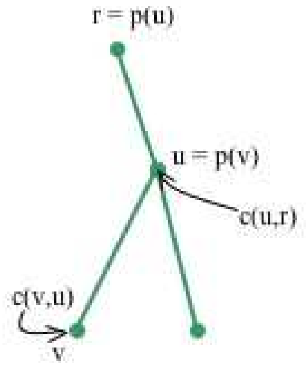
Section of tree T

$\boxed{T \text{ is an MST of } G \textit{ iff } c(x, y) \geq c(u, v) \;\; \forall \{u, v\} \in P(x, y)}$

Rajendra "Raz" Magar
magar@cs.dartmouth.edu

Tarjan's algorithm on a pointer machine (like LCA):

Uses *link-eval* data structure, which maintains a forest F contained in tree T with real-valued vertices.



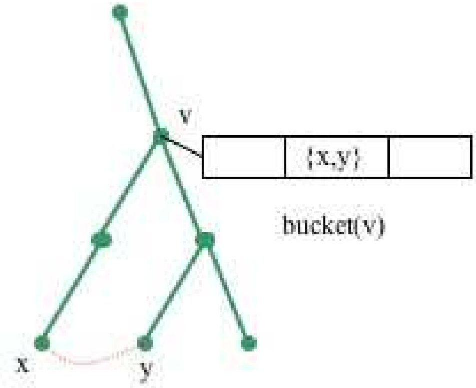
Value of vertex = cost of edge with the parent

$\texttt{LINK}(u)$: Add edge $\{P_T(u), u\}$ to F.
$\texttt{EVAL}(u)$: If $u = r$ (root of the tree containing u in F), return r. Otherwise, return any vertex $x \neq r$ of max value on $P(r, u)$.

Verification: Algorithm

- $\forall \{x, y\} \in E \backslash E_T$, find $v = \texttt{LCA}(x, y)$ and place $\{x, y\}$ into $\texttt{bucket}(v)$
- Perform DFS of T. Call SEARCH(r). See procedure SEARCH below.

SEARCH(v):
  for each child w of v:
      SEARCH(w)
  for each $\{x, y\} \in \texttt{bucket}(v)$:
      $\texttt{EVAL}(x)$ = max weight on $P(x, v)$
      $\texttt{EVAL}(y)$ = max weight on $P(y, v)$
      if $(c(x, y) < \texttt{max}(\texttt{EVAL}(x), \texttt{EVAL}(y)))$
        then : Report $T \neq$ MST. Break
        else : Continue
  for each child w of v:
      $\texttt{LINK}(w)$



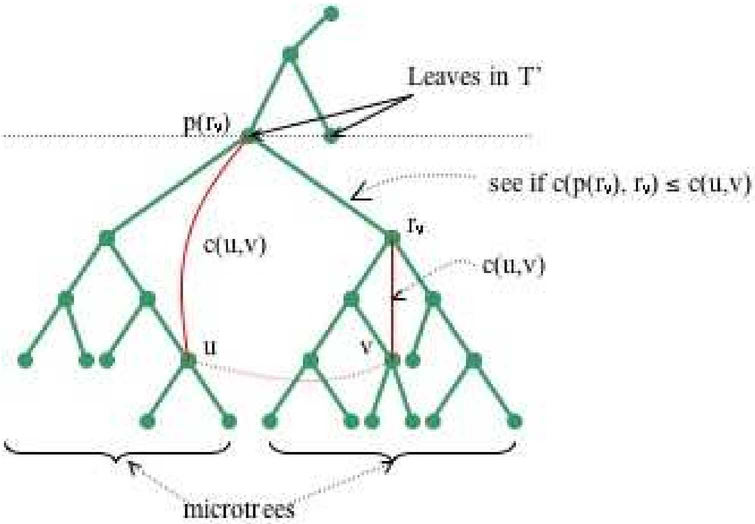
Running time of Tarjan's algorithm: $O(m\alpha(m,l)+n)$

Now, break the tree into T' and microtrees.

We cannot apply the previous micro-tree technique directly because the weights on the edges require $> O(1)$ bits which prevents efficient encoding.

Theorem: We can verify if T is an MST of G in $O(n+m)$ time on a pointer machine.

Simplified DRT (Dixon, Rauch, Tarjan)



Those non-tree edges $\{u,v\}$ whose u,v are not in the same microtrees or if only one of them is in a microtree, get replaced by the edges $\{u,p(r_v)\}$ and $\{v,r_v\}$ as shown in the figure above.

T' has $O(n/g)$ leaves or $O(n/\log^{\frac{1}{3}} n)$ leaves if $g=\log^{\frac{1}{3}} n$.

Rajendra "Raz" Magar
magar@cs.dartmouth.edu

Running time on T' using Tarjan's technique is $O(m\alpha(m, n/\log^{\frac{1}{3}} n) + n) = O(m + n)$.

Micro-tree verification on a pointer machine (Komlós)

Use a decision tree D whose vertices determine comparisons of edge costs. Consider a graph G^* with $|G^*| \leq g$ and $e = O(g^2)$. D has depth $O(e)$ and $2^{O(e)}$ edges and needs $g^3 2^{O(g^2)}$ time for construction on a pointer machine.

Eliminate duplicate replacement edges of the form $\{r_v, v\}$ by scanning r_v's neighbor list, keeping the minimum cost edges, so each microtree's induced graph has at most $g^2/2$ edges. We can compute decision trees for all microtrees in $O(n+m)$ time on a pointer machine if $g = O(\log^{\frac{1}{3}} n)$. Traversing decision tree to verify a microtree takes linear time.