Date: May 2, 2016

Definition 1 *A **problematic vertex** is a vertex with degree greater than two. A leaf subpath is a subpath with one endpoint being a problematic vertex and one endpoint being a leaf. An internal subpath is a subpath whose endpoints are both problematic vertices, or the root.*

The algorithm will maintain two selection lists, $\mathcal{M}_1$ and $\mathcal{M}_2$. The first selection list is for paths, and the second selection list is for handling problematic vertices whose leaf paths have been resolved.

Algorithm:

```
1: Initialize the algorithm
2: Repeatedly run rounds such that
3: for Round r: do
4:     Repeat Steps 5 − 8 a total of 3 times:
5:     Run Phase 1 a total of 70r times.
6:     for 6(r^2 + 9) times: do
7:         Run Phase 1 a total of 110r times.
8:         Run Phase 2.
```

Initialization: Given a tree T with n vertices initially, find the edge-path-partition of T and remove all problematic vertices. Take each subpath P_i that remains and extend

the length to t_i, the smallest possible power of two while labeling from the root, by adding in vertices with vertex weight zero, as necessary. For each subpath P_i, create an associated matrix, and insert its largest and smallest values into selection set $\mathcal{M}_1$ with weight $\lceil 4n^4/t_i \rceil$. For each subpath, repeatedly split the subpath, and insert the largest and smallest nonzero values of the associated matrices into $\mathcal{M}_1$ with twice the weight of the previous weight.

Phase 1: Select the weighted and unweighted medians from $\mathcal{M}_1$, test for feasibility, and adjust λ_1 and λ_2 accordingly. During the feasibility testing, clean and glue adjacent subpaths which have been resolved. If any leaf path has been completely resolved, represent the leaf path by a single vertex.

Phase 2: For any problematic vertex with a resolved leaf path, insert into $\mathcal{M}_2$ the weight of that vertex plus the accumulated remaining weight from the resolved leaf path. Select the median from $\mathcal{M}_2$, perform the feasibility test, and adjust λ_1 and λ_2 accordingly. That is, for the max-min problem, if the number of cuts is at least k, then for each vertex

whose weight plus the accumulated remaining weight from the resolved leaf path is at most the median, we merge the vertex with the accumulated remaining weight from the resolved leaf path. On the other hand, if the number of cuts is less than k, then for each vertex whose weight plus the accumulated remaining weight from the resolved leaf path is at least the median, we cut above the parent vertex, noting that any future feasibility tests would do the same.

Definition 2 *A **blocked** path is a path between a problematic vertex and either another problematic vertex or the root of the tree which is completely cleaned and glued. Conversely, an **active** path contains some value that has not been resolved.*

Definition 3 *Define an **iteration** of the feasibility test as a single feasibility test on $\mathcal{M}_1$ or $\mathcal{M}_2$.*

Theorem 4 *Tree $\mathcal{T}$ with n vertices can be modified to a binary tree $\mathcal{T}'$ with at most $2n$ vertices which has the same result.*

Proof For a problematic vertex v and corresponding children $c_1, c_2, \ldots, c_m$ in $\mathcal{T}$, create

m vertices $a_1, a_2, \ldots, a_m$ in $\mathcal{T}'$, along with vertices v and $c_1, c_2, \ldots, c_m$ such that v has children c_1 and a_1 in $\mathcal{T}'$. Furthermore, set the children of each a_i in $\mathcal{T}'$ to be c_{i+1} and a_{i+1}. The resulting structure is a binary tree with at most double the number of vertices. By setting the weight of each a_i in $\mathcal{T}'$ to be zero, the resulting structure clearly has the same λ. ■

Lemma 5 *Suppose all paths in $\mathcal{M}_1$ have weights as defined above. Then, following i iterations of feasibility testing in $\mathcal{M}_1$ without an iteration of Phase 2, at most $1/2^{5k}$ of the subpaths, of length 2^{j-k}, in $\mathcal{M}_1$ can be unresolved following, where $2^{5j} = 6 * (6/5)^i$.*

Proof Corresponding to the subpaths of length 2^j, there are at most $n/2^j$ submatrices in $\mathcal{M}_1$. If such a matrix is quartered repeatedly until 1×1 submatrices result, each such submatrix will have weight $n^4/2^{4j-5}$. Thus, when as little as $(n/2^j) * (n^4/2^{4j-5})$ weight remains, all subpaths of length 2^j can still be unresolved. This can be as late as iteration i, where i satisfies $(4/3) * 4n^5 * (5/6)^i = n^5/2^{5j-5}$, or $2^{5j} = 6 * (6/5)^i$. While all subpaths of length 2^j can still be unresolved on this iteration, at most $(1/2 * 1/2^4)^k = 1/2^{5k}$ of the

subpaths of length 2^{j-k} can be unresolved for $k = 1, \ldots, j-1$ and at most $1/2^{5j-2}$ of the subpaths of length 1 can still be unresolved. ■

Theorem 6 *The total time for handling $\mathcal{M}_1$ and performing selection over all iterations is $O(n)$.*

Proof First, we show that the time to form and handle R, the multiset consisting of the smallest element and the largest element from each matrix in $\mathcal{M}_1$, is linear in n. We give an accounting argument. Charge 2 credits for each value inserted into R. As R changes, we maintain the invariant that the number of credits is twice the size of R, as R changes. When R has k elements, a selection takes $O(k)$ time, paid for by k credits, leaving k credits still available. Then, $k/2$ elements are removed from R, so that the invariant is maintained. Since n elements are inserted into R during the whole of the algorithm, the time for forming R and performing selections is $O(n)$.

It remains to count the number of submatrices inserted into $\mathcal{M}_1$. Initially, at most $2n-1$ submatrices are inserted into $\mathcal{M}_1$. For $j = 1, 2, \ldots, \log n - 1$, consider all submatrices of

size $2^j \times 2^j$ that are at some point inserted into $\mathcal{M}_1$. A matrix that can be split must have its smallest value at most λ_1 and its largest value at least λ_2. However, $M_{i,j} > M_{i-k,j+k}$ for $k > 0$, since the path represented by $M_{i-k,j+k}$ is a subpath of the path represented by $M_{i,j}$. Hence, for any submatrix of size $2^j \times 2^j$ which is split, at most one submatrix can be split in each diagonal extending upwards from left to right. There are fewer than $2n$ diagonals, so there will be fewer than $2(n/2^j)$ submatrices that are split. Thus the number resulting from quartering is less than $8(n/2^j)$. Summing over all j gives $O(n)$ submatrices in $\mathcal{M}_1$ resulting from quartering. ■

We state three lemmas which we shall prove subsequently.

Lemma 7 *Suppose at the beginning of round r, that the feasibility test searches at most $n/2^{r-1}$ vertices. If the feasibility test searches at least as many vertices in active paths as vertices in blocked paths, then following $70r$ iterations, which takes $O(nr/2^r)$ time, either:*

1. *The number of vertices in active paths that the feasibility test searches is halved.*

2. *The number of vertices in active paths that the feasibility test searches is at most* $n/2^{r+1}$.

Lemma 8 *Suppose at the beginning of round* r*, that the feasibility test searches at most* $n/2^{r-1}$ *vertices. If the feasibility test searches more vertices in blocked paths than vertices in active paths, then following* $6(r^2+9)(110r+1)$ *iterations, which takes* $O(nr^3/2^r)$ *time, either:*

1. *The number of vertices in blocked paths that the feasibility test searches is halved*

2. *The number of vertices in blocked paths that the feasibility test searches is at most* $n/2^{r+1}$.

Lemma 9 *At the beginning of round* r*, the feasibility test searches at most* $n/2^{r-1}$ *vertices.*

Proof We note that prior to round 1, the feasibility test searches exactly n vertices and proceed via induction. Suppose at the beginning of round r, the feasibility test searches

at most $n/2^{r-1}$ vertices. If the feasibility test in fact searches at most $n/2^r$ vertices, then the induction already holds. Then we have two cases:

1. The feasibility test spends as much on active paths or more, compared to blocked paths

2. The feasibility test spends more time on blocked paths than active paths

If the feasibility test spends as much on active paths or more, compared to blocked paths, then by Lemma 7, $O(nr/2^r)$ time is used to reduce the portion spent by the feasibility test on active paths by at least half.

Otherwise, if the feasibility test spends more time on blocked paths than active paths, then by Lemma 8 and $O(nr^3/2^r)$ time is used to reduce the portion spent by the feasibility test on blocked paths by at least half. Thus, we can reduce the overall number of vertices that the feasibility test searches by $1/4$ in $O(nr^3/2^r)$ time, for each r. Since $(3/4)^3 < 1/2$, then three repetitions suffice to halve the overall number of vertices checked by the feasibility test. Indeed, each round contains three repetitions, and so at the beginning of

round $r+1$, the runtime is at most $n/2^r$. ■

Now, we formally state the main result of paper.

Theorem 10 *The total runtime required by the algorithm is $O(n)$.*

Proof By Lemma 9, the feasibility test searches at most $n/2^{r-1}$ vertices at the beginning of round r. Furthermore, note that $\sum_{r=0}^{\infty} nr^3/2^{r-1}$ is $O(n)$. By Theorem 6, the total time for handling $\mathcal{M}_1$ and performing selection over all iterations is $O(n)$. Therefore, the total time required by the algorithm is $O(n)$. ■

Observation 11 *The time spent by the feasibility test on an active path is at least as much time as spent by the feasibility test as if the active path were a blocked path.*

Observation 12 *Resolving a blocked path does not increase the feasibility test time spent on active paths.*

We can now prove Lemma 7.

Proof of Lemma 7: By assumption, the feasibility test searches at most $n/2^{r-1}$ vertices and searches at least as many vertices in active paths as vertices in blocked paths. If the number of vertices in active paths that the feasibility test searches is at most $n/2^{r+1}$, then the result follows. Thus, we assume the feasibility test searches more than $n/2^{r+1}$ vertices in active paths. By Lemma 5, in iteration i, at most $1/2^{5k}$ of the subpaths of length 2^{j-k} can be unresolved, where $2^{5j} = 6(6/5)^i$. Then for $j = 2(r+1)$ and $k = r+1$, at most $1/2^{5(r+1)}$ of the subpaths of length $2^(r+1)$ can be unresolved, so there are at most $n/2^{5(r+1)}$ vertices remaining in active paths. Thus, it takes $i = (10(r+1) - \log 6)/\log(6/5) < 70r$ iterations to reduce the amount of time spent on the active paths by at least half. Since each iteration of feasibility testing searches at most $n/2^{r+1}$ vertices, the total number of vertices checked is at most $(n/2^{r-1})\,70r$. ■

Before we can prove Lemma 8, we introduce three preliminary lemmas.

Lemma 13 *Let P be a blocked path of length at most 2^l. Then a feasibility test searches at most $2l^2$ vertices in P.*

Proof The feasibility test searches at most $1 + 2 + \ldots + l$ vertices in each half of P. Thus, it searches at most $2l^2$ vertices of P in total. ■

Lemma 14 *Suppose the feasibility test searches at most $n/2^r$ vertices but more than $n/2^{r+1}$ vertices. If the feasibility test spends more time on blocked paths than active paths, then the median length of a leaf path is at most 2^{2r}.*

Proof Suppose, by way of contradiction, the median length of a leaf path is more than 2^{2r}. By assumption, the feasibility test spends more time on blocked paths than active paths, but the number of leaf paths is at least one more than the number of internal paths, active or blocked, so the median time spent on a blocked path is more than 2^{2r}. For each blocked path on which the feasibility test spends 2^{2r} time, the path length is at least $2^{2^{r-1}}$. Thus, for each active path of length 2^{2r} on which the feasibility test spends 2^{2r} time, the feasibility test spends at most 2^{2r} time per $2^{2^{r-1}}$ vertices on some subpath.

But then the ratio of the feasibility test to the total number of vertices is at most

$$\frac{2^{2r} + 2^{2r}}{2^{2r} + 2^{2^{r-1}}} << \frac{1}{2^r},$$

which contradicts the assumption that the feasibility test searches more htan $n/2^{r+1}$ vertices. Thus, the median length of a leaf path is at most 2^{2r}. ■

Lemma 15 *Suppose the feasibility test searches at most $n/2^r$ vertices but more than $n/2^{r+1}$ vertices. If the feasibility test spends more time on blocked paths than active paths, then the median length of a blocked path is at most 2^{r^2+9}, and the number of vertices the feasibility test searches in the median length blocked path is at most $2(r^2+9)^2$.*

Proof Suppose, by way of contradiction, the median length of a blocked path is more than 2^{r^2+9}. Then by Lemma 13, the number of vertices in blocked paths that the feasibility test searches is at most $\left(2(r^2+9)^2/2^{r^2+9}\right) n$. By assumption, the feasibility spends more time on blocked paths than active paths, so the number of vertices in blocked path-

s that the feasibility test searches is at least $n/2^{r+1}$. But for all positive integers i, it holds that $1/2^{i+1} > 2(i^2+9)/2^{i^2+9}$. Thus, $n/2^{r+1}\left(2(r^2+9)^2/2^{r^2+9}\right)n$, which contradicts the assumption that the feasibility test searches more htan $n/2^{r+1}$ vertices. Hence, the median length of a blocked path is at most 2^{r^2+9}. ■

We now finish the proof of Lemma 8.

Proof of Lemma 8: By assumption, the feasibility test searches at most $n/2^{r-1}$ vertices and searches more vertices in blocked paths than vertices in active paths. If the number of vertices in blocked paths that the feasibility test searches is at most $n/2^{r+1}$, then the result follows. Thus, we assume the feasibility test searches more than $n/2^{r+1}$ vertices in blocked paths.

By Lemma 14, the median length of a leaf path is at most $2^{2(r+1)}$. By Lemma 5, in iteration i, at most $1/2^{5k}$ of the subpaths of length 2^{j-k} can be unresolved, where $2^{5j} = 6(6/5)^i$. Then for $j = 3(r+1)$ and $k = r+1$, at most $1/2^{5(r+1)}$ of the subpaths of length $2^{2(r+1)}$ can be unresolved, so at least half of the leaf paths are resolved. It takes

$i = (15(r+1) - \log 6)/\log(6/5) < 110r$ iterations to resolve half of the leaf paths, so that the appropriate values can be inserted into $\mathcal{M}_2$. Another iteration of feasibility testing is run using the selected median from $\mathcal{M}_2$. Thus, running $110r$ iterations of Phase 1, followed by an iteration of Phase 2 reduces the number of leaf paths by a factor of $1/4$, or equivalently, reducing the total number of paths by a factor of $1/8$.

Since $(7/8)^6 < 1/2$, then by repeating $6(r^2 + 9)$ times, the total number of paths is reduced by a factor of at least $1/2^{r^2+9}$. If the average length of the remaining blocked paths is more than 2^{r^2+9}, then by Lemma 15, the feasibility test searches at most $n/2^{r+1}$ vertices, which is a reduction of $1/2 > 1/4$ in the number of vertices checked by the feasibility test. Otherwise, if the average length of the remaining blocked paths is less than 2^{r^2+9}, then by reducing the total number of paths by factor of at least $1/2^{r^2+9}$, the time spent by the feasibility test on vertices in blocked paths is at least halved. We require $6(r^2+9)$ cycles, each with $110r+1$ iterations of feasibility testing. Thus, at most $6(r^2+9)(110r+1) = O(r^3)$ iterations are needed to reduce the feasibility test by at least

half, each checking at most $n/2^r$ vertices, for a total of $O(nr^3/2^r)$ time. ■