Date: August 23, 2016

(For a review of the algorithm, see the end of the notes)

For the purpose of analysis, consider a data structure $\mathcal{M}$ which consists of matrices representing subpaths of the tree, along with their corresponding weight as a function of their lengths, as defined in Section **??** (Section 2). Intuitively, $\mathcal{M}$ represents a superset of the matrices whose values are tested in a selection and feasibility test. Namely, $\mathcal{M}$ consists of submatrices of both heavy paths and light paths, but since $\mathcal{M}$ consists of representations instead of the actual values, we need not compute values for light paths, which we may not be able to do in linear time. Indeed, the subset of $\mathcal{M}$ contained in heavy paths will be in $\mathcal{H}$, while a separate subset of $\mathcal{M}$ contained in light paths will be in $\mathcal{L}$. For heavy paths, we generate the entire submatrix, and insert the representatives whose values are within (λ_1, λ_2) into $\mathcal{H}$. For light paths, we do not generate the submatrix, but we nevertheless count them towards our progress by establishing artificial weights for matrices both generated and ungenerated, and showing a reduction in artificial weight.

Lemma 0.1 *At any point in the algorithm, the weight of $\mathcal{M}$ is less than* $(4/3) * 4n^5$.

Proof We first consider the weights assigned to submatrices in $\mathcal{M}$. Corresponding to subpaths of lengths $1, 2, 4, \ldots, n$, procedure *mats_for_path* creates at most n submatrices of size 1×1, at most $n/2$ submatrices of size 1×1, at most $n/4$ submatrices of size 2×2, and so on, up through one submatrix of size $n/2 \times n/2$. The total weight of the submatrices corresponding to each path length is at most $4n^5, n^5, n^5/4, \ldots, 4n^3$, resp. It follows that the total weight for submatrices of all path lengths is less than $(4/3) * 4n^5$. ■

Let $wgt(M)$ be the weight assigned to submatrix M. Define the *effective weight*, denoted *eff_wgt*(M), of a submatrix M in $\mathcal{M}$ to be $wgt(M)$ if both its smallest value and largest value are contained in the interval (λ_1, λ_2) and $(3/4)wgt(M)$ if only one of its smallest value and largest value is contained in the interval (λ_1, λ_2). Note that since *eff_wgt*(M) is purely for analysis, we can bound the number of matrices with values outisde (λ_1, λ_2) without explicitly calculating the values. Let *eff_wgt*$(\mathcal{M})$ be the total effective weight of all submatrices in $\mathcal{M}$.

Lemma 0.2 *A weighted and unweighted selection from* $\mathcal{L}$ *and then* $\mathcal{H}$ *resolves at least* $1/24$ *of eff_wgt*(M)

Proof When a feasibility test renders a value (the smallest or largest) from M no longer in the interval (λ_1, λ_2), we argue that *eff_wgt*(M) is reduced by at least $wgt(M)/4$ because of that value.

If both values were contained in the interval (λ_1, λ_2), and one is no longer is, then clearly it is true. If only one value was contained in the interval (λ_1, λ_2), and it no longer is, then M is replaced by four submatrices of effective weights $wgt(M)/8 + wgt(M)/8 + (3/4)wgt(M)/8 + (3/4)wgt(M)/8 < wgt(M)/2$, so that there is a reduction in weight by greater than $wgt(M)/4$. If both values were contained in the interval (λ_1, λ_2), and both are no longer in, then we consider first one and then the other.

Thus every element that was in (λ_1, λ_2) but no longer is causes a decrease in *eff_wgt*$(\mathcal{M})$ by an amount at least equal to its weight in $\mathcal{M}$. Values with more than half of the total weight in $\mathcal{M}$ find themselves no longer in (λ_1, λ_2).

Recall that we include values representing a submatrix M in selection list $\mathcal{H}$ if and only if M represents a heavy path, and the corresponding values are contained within (λ_1, λ_2). Of a submatrix M which does have representative values contained in $\mathcal{H}$, then M has either two values in $\mathcal{M}$ at a total of $2w(M)/4 = (1/2)w(M)$ or one value at a weight of $w(M)/4 = (1/3)(3w(M)/4)$. For each selection and feasibility test from $\mathcal{H}$ and $\mathcal{L}$, either at least half of $\mathcal{M}$ is contained in $\mathcal{L}$ or contained in $\mathcal{H}$.

Suppose at least half of $\mathcal{M}$ is contained in $\mathcal{L}$. Then following a selection and feasibility test from $\mathcal{L}$, at least one quarter of the values and weight from $\mathcal{M}$ will be resolved, either determined to be smaller than an updated λ_1, or larger than an updated λ_2 and inserted into $\mathcal{H}$ as a part of a heavy path. Thus, following a round of selection and feasibility testing from $\mathcal{H}$, at least $1/8$ of $\mathcal{M}$ will be resolved.

On the other hand, if at least half of $\mathcal{M}$ is contained in $\mathcal{H}$, then a round of selection and feasibility testing from $\mathcal{H}$ resolves at least $1/4$ of $\mathcal{M}$.

Thus *eff_wgt*$(\mathcal{M})$ decreases by a factor of at least $(1/8)(1/3) = 1/24$ per iteration. ■

Lemma 0.3 *Following i iterations of weighted and unweighted selection and feasibility testing separately for $\mathcal{L}$ and $\mathcal{H}$, where $2^{5j} = (3/4)*(24/23)^i$, at most $1/2^{5k}$ of the subpaths, of length 2^{j-k}, in $\mathcal{H}$ can be unresolved.*

Proof Corresponding to the subpaths of length 2^j, there are $n/2^j$ submatrices created by *mats_for_path*. If such a matrix is quartered repeatedly until 1×1 submatrices result, each such submatrix will have weight $n^4/2^{4j-2}$. Thus when as little as $(n/2^j)*(n^4/2^{4j-2})$ weight remains, all subpaths of length 2^j can still be unresolved. This can be as late as iteration i, where i satisfies $(4/3)*4n^5*(23/24)^i = n^5/2^{5j-2}$, or $2^{5j} = (3/4)*(24/23)^i$. 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 $k = 1, \ldots, j$. Also, by Lemma 3.1, each subpath can be searched by *FTEST1* in amortized time proportional to the logarithm of its length. Thus the time to search path P on iteration i is at worst proportional to $(j/2^j)n(1+1/2^5+1/2^{10}+\cdots)$, which is $O((j/2^j)n)$. From the relationship of i and j, $2^j = (3/4)^{1/5}(24/23)^{i/5}$, and $j = (1/5)\log(3/4) + (i/5)\log(24/23)$. The lemma then follows. ■

Lemma 0.4 *The total time to generate the submatrices for all the heavy paths is $O(n)$.*

Proof Since each vertex can be in at most two overlapping subpaths, the total amortized time for generating all heavy paths is at most $2n$. ■

A Review of the Algorithm The algorithm will maintain three selection lists, $\mathcal{H}$, $\mathcal{L}$, and $\mathcal{V}$. The first selection list is for paths, the second selection list is for sequences of light paths formed by the resolution of a problematic vertex, and the third selection list is for handling problematic vertices whose leaf paths have been resolved.

The algorithm will employ three procedures. In procedure *handle_light_paths*, the algorithm selects the weighted and unweighted medians from $\mathcal{L}$ separately, tests for feasibility, and adjusts λ_1 and λ_2 accordingly. For any paths which become heavy, the algorithm generates the corresponding submatrix and inserts the representative whose values are within (λ_1, λ_2) into $\mathcal{H}$.

The second procedure is *handle_active_paths*, in which the algorithm selects the weighted and unweighted medians from $\mathcal{H}$ separately, tests for feasibility, and adjusts λ_1 and λ_2 accordingly. During the feasibility testing, the algorithm cleans and glues adjacent subpaths that have been resolved. When a leaf path has been completely resolved, the algorithm represents the leaf path by a single vertex with the remaining accumulated weight, $accum\_wgt(v)$, as described in Section **??** (Section 2).

The third procedure is *handle_leaves*. For any problematic vertex with a resolved leaf path hanging off it, the procedure inserts into $\mathcal{V}$ the weight of that vertex plus the accumulated remaining weight from the resolved leaf path. The procedure then selects the median from $\mathcal{V}$, performs the feasibility test, and adjusts 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, it merges 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, it cuts above the parent vertex, noting that any future feasibility tests would do the same. Note that any extra vertices we had artifically included to round the total number of vertices in each path to a power of two have weight zero, which will have already been resolved by this point. If a problematic vertex is resolved, leaving a sequence of light paths, insert the sum of the light paths into $\mathcal{L}$.

TREE1:

Initialize the data structures for the algorithm and set round $r \leftarrow 1$.

```
**while** $\lambda_1 < \lambda_2$ **do**
    **for** three times **do**
        Call procedure *handle_light_paths* a total of $163r + 170$ times.
        Call procedure *handle_active_paths* a total of $163r + 170$ times.
        **for** $6(r^2 + 9)$ times **do**
            Call procedure *handle_light_paths* a total of $245r + 252$ times.
            Call procedure *handle_active_paths* a total of $245r + 252$ times.
            Call procedure *handle_leaves*.
        **endfor**
    **endfor**
    $r \leftarrow r + 1$
**endwhile**
```