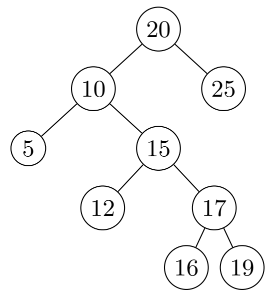
| CS 240E: Structures and Data Management | Winter 2021 |
|---|---|

# Tutorial 5: Splay trees and Interpolation search

**Warmup.** Given the following splay tree $S$, calculate its potential using the potential function

$$\Phi(i) := \sum_{v \in S} \log n_v^{(i)},$$

where $n_v^i$ is the number of nodes in the subtree rooted at $v$ after $i$ operations, including $v$ itself. Insert the key 18. Calculate the new potential. Verify that the difference between the potential difference is less than $4 \log n - 2R + 2$, where $R$ is the number of rotations.




**1.** Let $A$ be an unordered array with $n$ distinct items $k_0, ..., k_{n-1}$. Give an asymptotically tight $\Theta$-bound on the expected access cost if you put $A$ in the optimal static order for the folliwng probability distributions:

(a) $p_i = \frac{1}{n}$ for $1 \leq i \leq n-1$

(b) $p_i = \frac{1}{2^{i+1}}$, for $1 \leq i \leq n-2$, $p_{n-1} = 1 - \sum_{i=0}^{n-2} p_i = \frac{1}{2^{n-1}}$

**2.** This assignment will guide you towards a proof that a different modification of interpolation search also has expected run-time $O(\log \log n)$. Consider the modification shown in Algorithm 1 below, which compares not only at $A[m]$, but also at two indices $m_\ell$ and $m_r$ that are roughly $\sqrt{N}$ indices to the left and right of $m$ (where $N = r - l$), and repeats in the appropriate sub-array.

**Algorithm 1:** *interpolation-search-3way*$(A, n, k)$

Input: Sorted array $A$ of $n$ integers, key $k$ **if** *($k < A[0]$)* **then return** *"not found, would be left of index 0"*;
**if** *($k > A[n-1]$)* **then return** *"not found, would be right of index $n-1$"*;
**if** *($k = A[n-1]$)* **then return** *"found at index $n-1$"*;
$\ell \leftarrow 0$, $r \leftarrow n-1$ ;
**while** $(N \leftarrow (r - \ell) \geq 2)$ **do** `// inv:` $A[\ell] \leq k < A[r]$
    $m \leftarrow \ell + \lfloor \frac{k - A[\ell])}{A[r] - A[\ell]} \cdot (r - \ell) \rfloor$ ;
    $m_\ell \leftarrow \max\{\ell, m - \lfloor \sqrt{N} \}$; $m_r \leftarrow \min\{r, m + \lfloor \sqrt{N} \rfloor\}$;
    **if** *($k < A[m_\ell]$)* **then** $r \leftarrow m_\ell$;
    **else if** *($k < A[m]$* **then** $\ell \leftarrow m_\ell, r \leftarrow m$;
    **else if** *($k < A[m_r]$* **then** $\ell \leftarrow m, r \leftarrow m_r$;
    **else** $\ell \leftarrow m_r$;
**end**
**if** *($k = A[\ell]$)* **then return** *"found at index $\ell$"*;
**else return** *"not found, would be between index $\ell$ and $\ell + 1$"*;

**a)** Assume that the items in $A$ were randomly and uniformly chosen. Consider one execution of the while-loop, and call search-key $k$ *good* $A[m_\ell] \leq k < A[m_r]$ and *bad* otherwise. Show that $P(k \text{ is good}) \geq \frac{3}{4}$.

You may assume that all items in $A$ are distinct. You may also ignore rounding issues, i.e., assume $N$ is a perfect-square and $(r-\ell)\frac{k-A[\ell]}{A[r]-A[\ell]}$ is an integer.

(Hint: Define $idx(k)$ and $offset(k)$ as we did in class and use the properties of $offset(k)$ that we derived there.)

**b)** Let $T(n)$ be the expected run-time on $n$ items if items in $A$ were randomly and uniformly chosen, Argue that $T(n)$ satisfies the recursion $T(n) \leq T(\sqrt{n}) + O(1)$.

You may make the same assumptions as in the previous part, and also use without proof that $T(n)$ is monotone, i.e., $T(n-1) \leq T(n)$.

Hint: What is the run-time if $k$ is good? What if $k$ is bad?

Note that the last part implies that $T(n) \in O(\log \log n)$ as shown in class.

For all parts, you may use previous parts even if you did not prove them.