COMP90077 Advanced Algorithms and Data Structures

Assignment 1

Zhen Cai
Student ID: 1049487

Problem 1. Amortized Analysis

Prove that the rebuild(u) can be performed in O(size(u)) worst-case time.

Perform BST in-order traversal visits the tree along left subtree → root → right subtree, to return an ascending-ordered array A of T(u) elements, at the same time, know the number of nodes, i.e. size(u). This takes O(size(u)) time.

We take the median A[size(u)/2] as a root, recursively do this and insert child nodes at root from remaining two sides until no element's left. Each selection takes O(1). Therefore, alongside the cost of tree traversal, rebuild(u) will be performed in O(size(u)) worst-case time.

Because the split is at size(u)/2, the left subtree size will either be equal to, or 1 greater than the size of the right subtree, for all elements in destroyed T(u). This ensures a perfect WB-BST.

Prove that the height of a WB-BST on n elements is at most $\lfloor \log_{\beta} n \rfloor + 1$.

When a node u has two child nodes, let's call them ul and ur.

$$h(u) = \begin{cases} \max\{h(ul), h(ur)\} + 1, & \text{if } ul, ur \text{ both exist} \\ h(ul) + 1 \text{ } or \text{ } h(ur) + 1 = 2, & \text{if one of them exist} \\ 1, & \text{if both not exist} \end{cases}$$

The maximum height is obtained by maximizing $\max\{h(ul), h(ur)\}$, so we let one of them to have n/β elements, and it takes $\lfloor \log_{\beta} n \rfloor$ recursions to be reduced to 1. Plus the root at the very beginning, the height is at most $\lfloor \log_{\beta} n \rfloor + 1$.

Show that after the rebuild operation on the highest node u that has violation on the weight balance, the entire tree T is restored to a WB-BST.

As I've shown in question (a): a rebuild(u) operation solves all "violation on the weight balance" problem for u's descendants.

Since u was the highest node with such problem, the problem neither affect u's ancestors, nor any other node our discussion left (this is obvious). Therefore, the entire tree T is restored to a WB-BST.

Design an appropriate potential function and then show that the amortized cost for each insertion and deletion is bounded by $O(\log_\beta n)$.

First, it's trivial to prove the cost of BST insertion and deletion is $O(\log_\beta n)$.

Let root of a WB-BST be u0, the highest node to be balanced be u', and the inserted node be x. Define the potential function to be the sum of difference of subtrees among all nodes:

$$\Phi\big(T(u0)\big) = c \sum_{v \in T(u0):\, \delta(v)>1} \delta(v) \quad \text{where} \quad \delta(v) = |size(v.l) - size(v.r)|, \quad c = f(\beta) \text{ as a constant.}$$

$$\Phi(T_0) = 0, \ \ \Phi(T_i) \geq 0 \ \text{ for all } \ i \geq 1.$$

Now we consider two cases.

(i) No rebuild for insertion i:

Insertion of x only affects the potential of its ancestors whose δ is at least 2. Why I made this threshold? Because I'd like to make the rebuilt tree have 0 potential. Thus, we have

$$\triangle\,\Phi = \Phi(T_i) - \Phi(T_{i-1}) = cO\big(height(x)\big) = O\big(\log_\beta n\big)\,,$$

$$a(\sigma) = \mathrm{cost}(\sigma) + \triangle\,\Phi = O(\log_\beta n) + O(\log_\beta n) = O(\log_\beta n)\,.$$

(ii) Rebuild for insertion i:

$$\triangle\,\Phi = -c \sum_{v \in T_{i-1}(u0):\, \delta(v)>1} \delta(v) \leq -c\left(\frac{1}{\beta} - \left(1 - \frac{1}{\beta}\right)\right) size(u')$$
$$= c\left(1 - \frac{2}{\beta}\right) size(u')\,,$$

$$a(\sigma) = \text{cost}(\sigma) + \triangle\,\Phi \leq O(\log_\beta n) + size(u') - c\left(1 - \frac{2}{\beta}\right) size(u')$$
$$= O(\log_\beta n) \quad \text{if we set } c = \frac{\beta}{2-\beta}.$$

Same proof for deletion.

Problem 2. The Quake Heap.

Prove that the maximum possible height h_{max} is less than $c \cdot \log_{1/\alpha} n + 2 \cdot c$.

If $h_{max} \geq c \cdot \log_{1/\alpha} n + 2 \cdot c$, then

$$h_{max}/c - 2 \geq \log_{1/\alpha} n \;\Rightarrow\; \alpha^{h_{max}/c-2} n \leq 1 \;\Rightarrow\; \alpha^{h_{max}/c} n \leq \alpha^2 \;\Rightarrow\; \alpha^{\lfloor h_{max}/c \rfloor} n \leq \alpha^{2-(h_{max}/c - \lfloor h_{max}/c \rfloor)}.$$

As we have $h_{max}/c - \lfloor h_{max}/c \rfloor < 1$, $c \geq 1$, and $\alpha < 1$,

$$2 - \left(\frac{h_{max}}{c} - \left\lfloor \frac{h_{max}}{c} \right\rfloor\right) > 1 \;\Rightarrow\; \alpha^{2-\left(\frac{h_{max}}{c} - \left\lfloor \frac{h_{max}}{c} \right\rfloor\right)} < \alpha \;\Rightarrow\; \alpha^{\left\lfloor \frac{h_{max}}{c} \right\rfloor} n < \alpha$$
$$\Rightarrow\; 1 > \alpha^{\left\lfloor \frac{h_{max}}{c} \right\rfloor - 1} n = n_{\left(\left\lfloor \frac{h_{max}}{c} \right\rfloor - 1\right)\times c+1} = n_{\left\lfloor \frac{h_{max}}{c} \right\rfloor \times c - (c-1)} \geq n_{h_{max}},$$

which goes wrong because $n_{h_{max}}$ can be 1.

Therefore, sby contradiction, h_{max} is less than $c \cdot \log_{1/\alpha} n + 2 \cdot c$.

Prove that the space consumption bound of the quake heap now is $O(c \cdot n)$.

We can break down levels of tournament forests to create c ordinary forests. Then from the lecture, the space consumption of single quake heap is a geometric series, so takes no more than $O(n)$. c forests sum up to $O(c \cdot n)$.

Define a potential function related to the integer constant c, which can be used to perform amortized analysis for the subsequent questions.

$$\Phi(S_i) = N + 3 \cdot T + \frac{3}{c(2\alpha - 1)} \cdot B.$$

Show that the amortized cost of insert is bounded by O(1).

$$a(\sigma) = cost(\sigma) + \triangle\,\Phi = O(1) + 1 + 3 + \frac{3}{c(2\alpha - 1)} \cdot 0 = O(1).$$

Show that the amortized cost of decrease-key is bounded by $O\left(\frac{1}{c} \cdot \frac{1}{2 \cdot \alpha - 1}\right)$.

$$a(\sigma) = cost(\sigma) + \triangle\,\Phi = O(1) + 0 + 3 + \frac{3}{c(2\alpha - 1)} \cdot 1 = O\left(\frac{1}{c} \cdot \frac{1}{2\alpha - 1}\right).$$

Prove that the amortized cost of delete-min is bounded by $O(c \cdot \log_{1/\alpha} n)$.

Step 1-3 of delete-min has little difference to that of ordinary tournament forest.

$$\begin{aligned} a(\text{delete-min}_{1-3}) &\leq 2 \cdot \left(T^{(0)} + L\right) + \left(T^{(0)} - T^{(1)}\right) + 3 \cdot \left(T^{(1)} - T^{(0)}\right) + 0 \\ &= 2 \cdot \left(T^{(1)} + L\right) \leq 2 \cdot \left(2 \cdot h_{max}^{(0)} + h_{max}^{(0)}\right) < 6c \cdot \left(\log_{1/\alpha} n + 2\right) \\ &= O\left(\log_{1/\alpha} n\right). \end{aligned}$$

For step 4, let h-c+1 be the index of smallest height index of the violation, the actual cost for maintenance is $R = \sum_{h'>h} n_{h'}^{(0)}$. Thus, $\triangle\, N = -R$, $\triangle\, T \leq n_h^{(0)}$.

Observe that n_i can be calculated as the sum of:

- the number of root nodes at height-i,
- twice of the number of nodes at height-(i+1) with two children,
- the number of bad nodes at height-(i+1) with single child.

Then,

$$n_i^{(0)} \geq 2 \cdot \left(n_{i+1}^{(0)} - b_{i+1}^{(0)}\right) + b_{i+1}^{(0)} = 2 \cdot n_{i+1}^{(0)} - b_{i+1}^{(0)} \;\;\Rightarrow\;\; b_{i+1}^{(0)} \geq 2 \cdot n_{i+1}^{(0)} - n_i^{(0)}$$

$$\begin{aligned} \Rightarrow\;\; \triangle\, B = B^{(1)} - B^{(0)} &\leq -\sum_{i=1}^{c} b_{h+i}^{(0)} \leq -\sum_{i=1}^{c} 2 \cdot n_{h+i}^{(0)} - n_{h+i-1}^{(0)} \\ &< -\sum_{i=1}^{c} 2\alpha \cdot n_{h+i-c}^{(0)} - n_{h+i-c}^{(0)} \leq -\sum_{i=1}^{c} (2\alpha - 1) \cdot n_{h+i-c}^{(0)} \\ &= -(2\alpha - 1) \sum_{i=1}^{c} n_{h+i-c}^{(0)} \leq -c(2\alpha - 1) n_h^{(0)}. \end{aligned}$$

Therefore, $a(\text{delete-min}_4) < R - R + 3n_h^{(0)} - 3n_h^{(0)} < 0$.

Put all together, we have delete-min operation bounded by $O(c \cdot \log_{1/\alpha} n)$.