Problem Set 1

Evan Hataishi
ICS 621: Analysis of Algorithms
Prof. N. Sitchinava

September 6, 2019

1 Dynamic Arrays (40 pts)

Using the potential method, prove that insertion into the dynamic array has amortized cost of O(1).

Let D_i be a dynamic array, n be the number of items in D_i, and s be the size of D_i after i insertions. At any given time, D_i will have $n - \frac{s}{2}$ more items since the most recent resize. Each of these item's insertions must add enough "potential" to "pay" for a future resize. The potential must be enough to copy each item and a counterpart (in the first half of D_i). Thus, we can define our potential function to be

$$\begin{aligned}\Phi(D_i) &= 2(n - \frac{s}{2}),\\ &= 2n - s.\end{aligned}$$

D_0 is initially an array of size 0 with no elements. Items are only inserted and never removed, so the number of items n is guaranteed to be at least $\frac{s}{2}$. Therefore, this potential function is always greater than or equal to zero, and the amortized cost is an upper bound on the actual cost:

$$\Phi(D_i) \geq \Phi(D_0). \quad \checkmark$$

Suppose at the i-th insertion, the array is already full. A new array of size $2s$ is allocated, all n elements are copied over, and then the new element is finally inserted. n has increased by 1, and the array size has doubled. We obtain potential functions

$$\Phi(D_i) = 2(n+1) - 2s, \Phi(D_{i-1}) = 2n - s.$$

The actual cost c_i is the sum of the cost to copy s elements and insert the new element i.e. $c_i = s+1$. By equation (17.2), the amortized cost of this insertion is

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}),\\ &= (s+1) + (2(n+1) - 2s) - (2n - s),\\ &= 3,\\ &= O(1).\end{aligned}$$

In the simpler and cheaper case, where the i-th insertion is into a non-full array, we only need to insert without resizing. The potential functions become

$$\Phi(D_i) = 2(n+1) - s, \Phi(D_{i-1}) = 2n - s.$$

The actual cost c_i is simply the cost of inserting a new item i.e. $c_i = 1$. By equation (17.2), the amortized cost of this insertion is

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(D_i) - \Phi(D_{i-1}),\\ &= 1 + (2(n+1) - s) - (2n - s),\\ &= 3,\\ &= O(1).\end{aligned}$$

As we can see, both the cheap and expensive cases for insertion yield an amortized cost of O(1).

2 Binomial Trees (60 pts)

Prove that for the binomial tree B_k,

1. there are 2^k nodes
   Base Case: If $k = 0$, $2^0 = 1$ node. B_0 is a binomial tree with only 1 node, so $k = 1$ holds.
   Inductive Hypothesis: For any $k \geq 0$, a binomial tree B_k has 2^k nodes.
   Induction Step: Prove that binomial tree B_{k+1} has 2^{k+1} nodes.
   B_{k+1} is a binomial tree made up of 2 binomial trees B_k. Therefore, the total number of nodes in B_{k+1} is the sum of the nodes in both B_k subtrees. By our inductive hypothesis:

$$\begin{aligned}\#\text{ nodes} &= 2^k + 2^k,\\ &= 2(2^k),\\ &= 2^{k+1}. \quad \checkmark\end{aligned}$$

2. the height of the tree is k
   Base Case: If $k = 0$, height is also zero. B_0 is a binomial tree with only 1 node, so its height is 0.
   Inductive Hypothesis: For any $k \geq 0$, a binomial tree B_k has height k.
   Induction Step: Prove that binomial tree B_{k+1} has height $k+1$.
   B_{k+1} is a binomial tree made up of 2 binomial trees B_k with the root of one raised as the root of the other. Therefore, the height of B_{k+1} is the height of $B_k + 1$. By our inductive hypothesis:

$$\begin{aligned}\text{height} &= k + (1),\\ &= k + 1. \quad \checkmark\end{aligned}$$

3. there are exactly $\binom{k}{i}$ nodes at depth i for $i = 0, 1, ..., k$
   Base Case: If $k = 0$, depth can only be $i = 0$ (single root node). $\binom{0}{0}$ gives 1 node. B_0 is a binomial tree with only 1 node, so its depth is 0, and the root is its only node.
   Inductive Hypothesis: For any $k \geq i$, a binomial tree B_k has $\binom{k}{i}$ nodes at depth i.
   Induction Step: Prove that binomial tree B_{k+1} has $\binom{k+1}{i}$ nodes.
   B_{k+1} is a binomial tree made up of 2 binomial trees B_k with the root of one raised as the root of the other. The tree B_k that became the leftmost child of the now root tree B_k has the same group of nodes at depth-1 of B_{k+1} now. Therefore, the number of nodes at depth

i is the sum of the number of nodes in B_k at depth i + the number of nodes in B_k at depth $i-1$. By our inductive hypothesis:

$$\binom{k+1}{i} = \frac{(k+1)!}{i!(k+1-i)!} = \binom{k}{i} + \binom{k}{i-1}.$$

Expand:

$$\begin{aligned}
\binom{k}{i} + \binom{k}{i-1} &= \frac{k!}{i!(k-i)!} + \frac{k!}{(i-1)!(k-i+1)!}, \\
&= \frac{k!}{i(i-1)!(k-i)!} + \frac{k!}{(i-1)!(k-i+1)(k-i)!}, \\
&= \frac{k!(k-i+1)}{i(i-1)!(k-i)!(k-i+1)} + \frac{k!(i)}{i(i-1)!(k-i)!(k-i+1)}, \\
&= \frac{k!(k-i+1)+k!(i)}{i(i-1)!(k-i)!(k-i+1)}, \\
&= \frac{k!(k-i+1+i)}{i(i-1)!(k-i)!(k-i+1)}, \\
&= \frac{k!(k+1)}{i(i-1)!(k-i)!(k-i+1)}, \\
&= \frac{(k+1)!}{i!(k+1-i)!}. \quad \checkmark
\end{aligned}$$

4. the root has degree k, which is greater than that of any other node; moreover, as the Figure below shows, if we number the children of the root from left to right by $k-1, k-2, ..., 2, 1, 0$, then child i is the root of a subtree B_i
Proof. By the binomial tree's recursive definition, B_k is comprised of two binomial trees B_{k-1}, and each tree B_{k-1} is comprised of two more trees B_{k-2}, and so on. Therefore, by this recursive definition, B_k has higher degree than B_{k-1}, B_{k-1} has higher degree than B_{k-2}, etc..
Proof. By the binomial tree's recursive definition, to create a tree B_k, subtrees B_{k-1} are created, then B_{k-2}, and so on until B_0. The leftmost subtree will then be B_0. Ordering the children of the root by $k-1, k-2, ..., 2, 1, 0$ causes B_0 to become the 0-th child. B_1 becomes the 1-st child, and so on.