Chris Darnell
CS261-400
10/9/2016
amortizedAnalysis.txt

1. **How many cost units are spent in the entire process of performing 32 consecutive push operations on an empty array which starts out at capacity 8, assuming that the array will *double in capacity each time* a new item is added to an already full dynamic array? As N (ie. the number of pushes) grows large, under this strategy for resizing, what is the big-oh complexity for a push?**

1 1 1 1 1 1 1 1 - **8**

doub 1 1 1 1 1 1 1 1 - **16**   24 pushes, 8 left

doub - 32

8 + 16 + 32 = 56

The total cost is 56 units of time.

The big-oh complexity for a push as N grows large is O(1).

2. **How many cost units are spent in the entire process of performing 32 consecutive push operations on an empty array which starts out at capacity 8, assuming that the array will *grow by a constant 2 spaces* each time a new item is added to an already full dynamic array? As N (ie. the number of pushes) grows large, under this strategy for resizing, what is the big-oh complexity for a push?**

Push 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

Cost 1 1 1 1 1 1 1 1 9 1 1 12 1 1 15 1 1 18 1 1 21 1 1 24 1 1 27 1 1 30 1 1

The total cost is 260 time units.

Since 260/32 is a constant 8.175, as N grows large, the big-oh complexity is O(n).

3. **Suppose that a dynamic array stack doubles its capacity when it is full, and shrinks (on Pop only) its capacity by half when the array is half full or less. Can you devise a sequence of N push() and pop() operations which will result in poor performance (*O(N^2)* total cost)? How might you adjust the array's shrinking policy to avoid this? (Hint: You may assume that the initial capacity of the array is N/2.)**

From the previous questions, it is clear the largest cost gains are incurred when expanding or shrinking the capacity. So a method that requires fewer resizing operations would be faster.

A sequence of push() and pop() operations that result is poor performance (O(N^2)) could be:

Say initial capacity is 8, when N = 16 in N/2. Make 8 pushes, and the capacity will double. Then if you pop one off the stack, the capacity will be less than half, so the capacity would have to be instantly cut in half. This would result in very slow performance.

An adjustment to lower the cost would be to change the resizing policy to shrink the capacity by half when the capacity is only a 1/4 full. This would prevent unnecessary halving of the capacity and lower the time cost.