CS-261: Assignment 2 Written Analysis

Question1: The calculation can be shown in the following table(when N = 40):

a)

Append	1	2	**3**	4	5	**6**	7	8	9	10	11	**12**	13	14	15	16	17	18	19	20
Write Cost	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Copy Cost				3			6						12							
Total Cost	1	2	3	7	8	9	16	17	18	19	20	21	34	35	36	37	38	39	40	41

Append	21	22	23	**24**	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Write Cost	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Copy Cost					24															
Total Cost	42	43	44	45	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85

b) As N (i.e., the number of appends) grows large, under this strategy for resizing, the average big-O complexity for an append can be derived using the following calculation:

Mathematically, total cost = total write cost (S_w) + total copy cost (S_N) .

The total write cost, $S_w = N$.

Recall (from the table you prepared in part (a)) the copy cost is a sequence as follows:

3, 6, 12, … (a geometric sequence)

Now, what is the cost of the last resize (k^{th} term of the sequence)?

The k^{th} term is derived from the following equation: $a_k = a_1 * r^{k-1}$ (where a_1 is the first term and r is the ratio). For the sequence in our example above:

$a_k = 3*(6/3)\text{^}(k-1) = 3*2\text{^}(k-1)$

Now, what is the sum of k terms of the above sequence? You can derive that from the following formula:

$$S_k = a_1 \left(\frac{1 - r^k}{1 - r} \right)$$

For the sequence in the example above:

S_k =3(1-2^k)/(1-2) = 3*(1-2^k)/(-1)

The above equation should work for all integers, k, such that $k \geq 1$. In this instance, k **represents the resize term number** (**the 1st resize term, the 2nd resize term, ie,. 3, 6, 12,….).**

However, we are concerned with finding total cost in terms of N, so you must express the sum of the k resize operations in terms of N.

You already have determined the k^{th}resize cost (a_k) before. And, the k^{th} resize cost (a_k) is also equal to $\frac{N}{2}$because the array capacity will double to achieve a size of N and there would be half that number of existing elements to copy to the new array. Putting the two together, **you will find out the value of k in terms of N.**

[Some of you may have noticed that when discussing the total write cost, which is the size of the array, we say that it's N. And now, when discussing S_N, we say that a_k = N/2, equating N with the capacity. As N increases, the ratio of capacity to the size varies between 1 and 2 - just before a resize, they're the same, and just after a resize the capacity is twice the size. S_N is the same for any values between resizes (between terms of the geometric sequence), so we could assign to a_k any value from N/2 to N. Since the difference is a constant factor, it doesn't affect the big-O, and so we choose N/2 because it makes the algebra come out more neatly.]

Now, use the k and sum equation above (S_k) to derive the total copy cost for the given Nappends:

Since N/2 = 2^k * (2^-1) we get that k = log(n) we sub k into S_k and get:

S_N **= 3N-3**

Recall that total cost = total write cost (S_w)+ total copy cost (S_N) .

As we have conducted N appends, the average cost can be derived from:

$\frac{\{S_w + S_N\}}{N}$ = {(3*2^k) + (3n-3)} / 3*2^k = n/2^k - 1/2^k + 1

This gives us an amortized complexity of O(1).

Question2:

a) The calculation can be shown in the following table(when N = 40):

Append	11	22	33	44	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Write Cost	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Copy Cost				3		5		7		9		11		13		14		17		19
Total Cost	1	2	3	7	8	14	15	16	17	18	19	31	32	19	20	21	22	23	24	25

Append	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
Write Cost	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Copy Cost		21		23		25		27		29		31		33		35		37		39
Total Copy Cost	98	119	119	142	142	167	167	194	194	223	223	254	254	287	287	322	322	359	359	398
Total Cumulative Cost																				

b) As N (i.e., the number of appends) grows large, under this strategy for resizing, the average big-O complexity for an append can be derived using the following calculation:

Mathematically, total cost = total write cost (S_w) + total copy cost (S_N) .

The total write cost, $S_w = N$.

Recall (from the table you prepared in part (a)) the copy cost is a sequence as follows:

3, 5, 7, 9, 11, … (an arithmetic sequence)

Now, what is **the cost of the last resize** (k^{th} term of the sequence)?

The k^{th} term is derived from the following equation, and you will assume the sequence contains k terms:

$a_k = a_1 + d\,(k - 1)$ where a_1 is the first term of the sequence and d is the difference. In our example above:

a_k=3 + 2(k-1)

Now, what is the sum of k terms of the sequence? You can derive that from the following

equation: $S_k = k\left(\frac{a_1 + a_k}{2}\right)$

In our example above:

S_k = k((3 + 3+2(k-1))/2) = k(k+2) = k^2 + 2k

The above equation should work for all integers, k, such that $k \geq 1$. In this instance, k represents the resize term number.

However, we are concerned with finding total cost in terms of N, so you must express the sum of the k resize operations in terms of N.

You already have determined the k^{th} resize cost (a_k) before. Now you determine the k^{th} resize cost (a_k) in terms of N (For example, in part (a) when you had $N = 6$, the last resize cost was 5 (holds for all N, $N > 3$)). You can consider either of the cases (i.e., N is odd or N is even). **You will find out the value of k in terms of N.**

Now, use the k and sum equation above (S_k) to derive the total copy cost for the given Nappends:

We know a_k and using the above example, N = a_k - 1. Thus, N = 2k + 2.

Plugging into S_k we get:

S_N = N^2 + 2N = N(N+2)

S_N = N(N+2)

Recall that total cost = total write cost (S_w) + total copy cost (S_N) .

As we have conducted N appends, the average cost can be derived from:

$\frac{\{S_w + S_N\}}{N}$ = 2k+2+N^2+2N/2(k+1)

Solution:

We know S_W = N = 2k+2 and S_N = N(N+2). Using that we calculate:

{2k+2 + N(N+2)} / 2k+2 = 2k+2+N^2+2N/2k+2 = 2k+2+N^2+2N/2(k+1)

This gives us an amortized complexity of O(N^2).