Homework 1

Wednesday, September 4, 2024 9:52 PM

For each of the following code snippets (written in C), give an exact closed form formula for the number of comparisons executed, in terms of the variable n. Use a summation table whenever you see necessary. Make sure to show your work for partial credit.

Snippet (a)

```
for(int i = 0; i < n; i += 2) {
  if(a[i] > max) {
    max = a[i];
  }
}
```

i	# of comparisons
0	1
2	1
4	1
⋮	⋮
n-4	1
n-2	1
n	1

Time complexity $= \frac{n}{2}$

$\Theta(n)$

Snippet (b)

```
for(int i = 0; i < n; i++) {
  for(int j = 0; j < i; j++) {
    if(a[i] > max) {
      max = a[i];
    }
  }
}
```

i	j	# comparisons
0	0	0
1	0	1
2	0, 1	2
⋮		
n-3	0, 1	n-3
n-2	0, 1 ... n-3, n-2	n-2
n-1	0, 1 ... n-3, n-2, n-1	n-1

+

summation of all #'s up to and including n-1

$$\sum_{k=0}^{(n-1)} k = \sum_{k=1}^{(n-1)} k = \frac{(n-1)((n-1)+1)}{2} = \frac{(n-1)n}{2}$$

$$= \frac{n^2-n}{2} = \frac{n^2}{2} - \frac{n}{2}.$$

$\Theta(n^2)$

Snippet (c)

```
for(int i = 1; i < n; i *= 2) {
  if(a[i] > max) {
    max = a[i];
  }
}
```

i	# times compared
1	1
2	1
4	1
⋮	
$\frac{n}{8}$	1
$\frac{n}{4}$	1
$\frac{n}{2}$	1

1, 2, 4, 8, 16, 32 ......

to find how many times we can multiply i *2, we do $\log_2(n)$.

$\Theta(\log_2 n)$

Consider the follow implementation of the bubble sort algorithm:

```
void bubble_sort(int A[], int n) {
  for(int i = 0; i < n-1; i++) {
    for(int j = 0; j < n - i - 1; j++) {
      if(A[j] > A[j+1]) {
        // assume the swap function is implement elsewhere
        swap(A, j, j+1);
      }
    }
  }
}
```

Part (1): 10 points.

Consider the following instance of the array `A[5] = {4, -2, 12, 5, 0}`. Describe, step by step, how bubble sort would approach sorting this array.

i	j	# of swaps	(max)
0 ...	0, 1, 2, 3	4	$(n-1)$
1 ...	0, 1, 2	3	$(n-2)$
2 ...	0, 1	2	$(n-3)$
3 $(n-2)$	0	1	$(n-4)$
4 $(n-1)$		0	$(n-n)$

$$\sum_{k=0}^{n-1} k = \sum_{k=1}^{n-1} = \frac{(n-1)(n-1+1)}{2}$$

$\Theta(n^2)$

How bubble sort approaches sorting the array is quite simple. When I = 0, we go through the list starting at index j (0) and comparing if index 0 is greater than index j + 1 (1), which since 4 is greater than -2, we will swap these values. A = {-2, 4, 12, 5, 0}, next when j = 1, we check if 4 > 12, which it is not so no swap. Next j = 2 we check if 12 > 5 which it is so we do another swap. A = {-2, 4, 5, 12, 0}. Next j = 3, check if 12 > 0, which it is so we do a swap.
After the first iteration of the outer loop while I = 0, our array is A = {-2, 4, 5, 0, 12}.

We repeat this process for when I = 1, this means we will only go till j = n - 1 - 1, which in this case j < 3.
First we check if -2 < 4, so no swap, second we check 4 < 5, so no swap, next 5 > 0 so we swap it.
After I = 1, the array is A = {-2, 4, 0, 5, 12}.

Now for I = 2,
After running the loop the array is A = {-2, 0, 4, 5, 12}. Which the array is now sorted, however the algorithm will continue running performing its last checks.

For I = 3
We will check if -2 > 0 which is false so no swap. We are now done sorting