Problem Set #2

5/5 points earned (100%)

Back to Week 2

Quiz passed!

1 / 1 points

1. This question will give you further practice with the Master Method. Suppose the running time of an algorithm is governed by the recurrence $T(n) = 7 * T(n/3) + n^2$. What's the overall asymptotic running time (i.e., the value of T(n))?

Note: If you take this quiz multiple times, you may see different variations of this question.

- $\theta(n^{2.81})$
- $\theta(n^2)$

Correct Response
a=7, b=3, d=2. Since b^d > a, this is case 2 of the Master Method.

- $\theta(n \log n)$
- $\theta(n^2 \log n)$

---

1 / 1 points

2. Consider the following pseudocode for calculating a^b (where a and b are positive integers)

```
FastPower(a,b) :
  if b = 1
    return a
  else
    c := a*a
    ans := FastPower(c,[b/2])
  if b is odd
    return a*ans
  else return ans
end
```

Here [x] denotes the floor function, that is, the largest integer less than or equal to x.

Now assuming that you use a calculator that supports multiplication and division (i.e., you can do multiplications and divisions in constant time), what would be the overall asymptotic running time of the above algorithm (as a function of b)?

- $\Theta(b)$
- $\Theta(b \log(b))$
- $\Theta(\log(b))$

Correct Response
Constant work per digit in the binary expansion of b.

- $\Theta(\sqrt{b})$

---

1 / 1 points

3. Let $0 < \alpha < .5$ be some constant (independent of the input array length n). Recall the Partition subroutine employed by the QuickSort algorithm, as explained in lecture. What is the probability that, with a randomly chosen pivot element, the Partition subroutine produces a split in which the size of the smaller of the two subarrays is $\geq \alpha$ times the size of the original array?

- α
- $1 - \alpha$
- $1 - 2 * \alpha$

Correct Response
That's correct!

That's correct!

$2 - 2 * \alpha$

4. 1 / 1 points

Now assume that you achieve the approximately balanced splits above in every recursive call --- that is, assume that whenever a recursive call is given an array of length k, then each of its two recursive calls is passed a subarray with length between αk and $(1-\alpha)k$ (where α is a fixed constant strictly between 0 and $.5$). How many recursive calls can occur before you hit the base case? Equivalently, which levels of the recursion tree can contain leaves? Express your answer as a range of possible numbers d, from the minimum to the maximum number of recursive calls that might be needed.

- [x] $-\frac{\log(n)}{\log(\alpha)} \leq d \leq -\frac{\log(n)}{\log(1-\alpha)}$

Correct Response
That's correct!

- [ ] $-\frac{\log(n)}{\log(1-2*\alpha)} \leq d \leq -\frac{\log(n)}{\log(1-\alpha)}$
- [ ] $-\frac{\log(n)}{\log(1-\alpha)} \leq d \leq -\frac{\log(n)}{\log(\alpha)}$
- [ ] $0 \leq d \leq -\frac{\log(n)}{\log(\alpha)}$

5. 1 / 1 points

Define the recursion depth of QuickSort to be the maximum number of successive recursive calls before it hits the base case --- equivalently, the number of the last level of the corresponding recursion tree. Note that the recursion depth is a random variable, which depends on which pivots get chosen. What is the minimum-possible and maximum-possible recursion depth of QuickSort, respectively?

- [x] Minimum: $\Theta(\log(n))$; Maximum: $\Theta(n)$

Correct Response
The best case is when the algorithm always picks the median as a pivot, in which case the recursion is essentially identical to that in MergeSort. In the worst case the min or the max is always chosen as the pivot, resulting in linear depth.

- [ ] Minimum: $\Theta(\log(n))$; Maximum: $\Theta(n \log(n))$
- [ ] Minimum: $\Theta(1)$; Maximum: $\Theta(n)$
- [ ] Minimum: $\Theta(\sqrt{n})$; Maximum: $\Theta(n)$