CS222 Algorithm Design and Analysis Homework 1

Zhou Litao 518030910407 F1803016

September 19, 2020, 2020 Fall Semester

Exercise 1 Prove that $\log(\log n) = o(n^k)$, where k is a positive constant. (ps: $\log n$ refers to $\log_2 n$.)

Proof.

$$\lim_{n\to\infty} \frac{\log(\log n)}{n^k} = \lim_{n\to\infty} \frac{\frac{1}{n \ln n \cdot \ln 2}}{kn^{k-1}} = \lim_{n\to\infty} \frac{1}{kn^k \ln n \ln 2} = 0 < \infty \tag{1}$$

Hence, by definition, $\log(\log n) = O(n^k)$. □

Exercise 2 Prove that for any integer $n^2 - 1 > 3$, there is a prime p satisfying $n! > p > n$

Proof. Consider $n! - 1$.

Since $n!$ is the product of $1, 2, \ldots, n$, $n! - 1$ can't be divided by $1, 2, \ldots, n$ with a remainder of 0.

If $n! - 1$ is a prime, then the result follows.

If $n! - 1$ is not a prime, then we can always find a prime divider greater than n,

i.e. there is a prime p satisfying $n! > p > n$ □

Exercise 3 Assume that there is a recurrence formula as follows:

$$D(x) = \begin{cases} 1, & if\ \lfloor x \rfloor \le 1 \\ 3D(x/4) + x - 2, & if\ \lfloor x \rfloor > 1 \end{cases}$$

Please deduce the non-recursive expression of $D(x)$ and point out its asymptotic complexity.

Solution. For $x \in (-\infty, 2)$, $D(x) = 1$

For $x \in [2, 8)$, Let $g(k = 0, x) = D(x) = 3D(x/4) + x - 2 = 3 + x - 2 = x + 1, \ldots$

For $x \in \left[2 \cdot 4^k, 2 \cdot 4^{k+1}\right)$, we have $g(k, x) = 3D(x/4) + x - 2 = 3g(k-1, x) + x - 2$.

Note that $g(k+1) - g(k) = 3(g(k) = g(k-1))$, we have

$$\begin{aligned} g(k) &= (g(k) - g(k-1)) + \ldots + (g(1) - g(0)) + g(0) \\ &= 3^k x + \ldots + 3^2 x + 3x + (x+1) \\ &= (x+1) + \frac{3}{2}(3^k - 1)x = \frac{3^{k+1} - 1}{2} x + 1 \end{aligned} \tag{2}$$

$$\begin{aligned} D(x) &= \frac{3^{k+1} - 1}{2} x + 1, \text{ for } k \in \left(\log_4 \frac{x}{2} - 1, \log_4 \frac{x}{2}\right] \\ &= \frac{3^{\max\left(\lfloor \log_4 \frac{x}{2} \rfloor + 1, 0\right)} - 1}{2} x + 1 \end{aligned} \tag{3}$$

To analyze the asymtotic complexity, note that

$$\begin{aligned} D(x) &= \frac{3^{\max\left(\lfloor \log_4 \frac{x}{2} \rfloor + 1, 0\right)} - 1}{2} x + 1 \le \frac{3^{\log_4 \frac{x}{2} + 2}}{2} x + 1 \\ &= \frac{9 \cdot \left(\frac{x}{2}\right)^{\log_4 3} - 1}{2} x + 1 \in O\left(x^{\log_4 3 + 1}\right) \end{aligned} \tag{4}$$

□

Exercise 4 Use the minimal counterexample principle to prove that for any integer $n > 10$, there exist integers $i_n \geq 0$ and $j_n \geq 0$, such that $n = i_n \times 3 + j_n \times 4$.

Proof. First, it is easy to check 6, 7, 8, 9, 10 can be written as some combination of 3 and 4.

Suppose otherwise, then there exists a minimal counterexample $n' > 10$ and n' can't be written as the combination of 3 and 4. Since $\gcd(3, 4) = 1$, $n' - 3$ can't neither be written as the combination of 3 and 4.

If $n' - 3 \leq 10$, it contradicts to the previous checked fact. If $n' - 3 > 10$, it contradicts to the assumption that n' is the minimal counterexample. $\square$

Exercise 5 Analyze the **average** time complexity of QuickSort in Alg. 1.

Algorithm 1: QuickSort

```
   Input: An array A[1, ··· , n]
   Output: A[1, ··· , n] sorted nondecreasingly
 1 if n ≤ 1 then
 2 |  return;
 3 end
 4 pivot ← A[n]; i ← 1;
 5 for j ← 1 to n − 1 do
 6 |  if A[j] < pivot then
 7 |  |  swap A[i] and A[j];
 8 |  |  i ← i + 1;
 9 |  end
10 end
11 swap A[i] and A[n];
12 if i > 1 then QuickSort(A[1, ··· , i − 1]);
13 if i < n then QuickSort(A[i + 1, ··· , n]);
```

Solution. For input of the size N, the loops from line 5 to 10 takes $O(N)$ time, and the recursion will take $T(i-1)$ and $T(N-i-1)$ time for elements smaller than the pivot and greater than the pivot, where $T(n)$ is the time taken for n inputs.

For average case, we can take the average of every possible dividing case. The partition of N elements can be $\{0, N-1\}$, $\{1, N-2\}$, ... , $\{N-1, 0\}$. Hence we have

$$T(N) = 2\left(\frac{T(0) + T(1) + \ldots + T(N-1)}{N}\right) + cN \tag{5}$$

$$NT(N) = 2\left(T(0) + T(1) + \ldots + T(N-1)\right) + cN^2 \tag{6}$$

$$(N-1)T(N-1) = 2\left(T(0) + T(1) + \ldots + T(N-2)\right) + c(N-1)^2 \tag{7}$$

By subtracting Equation 7 from Equation 6 we have

$$NT(N) - (N-1)T(N-1) = 2T(N-1) + 2cN - c^2 \tag{8}$$

Since constant c^2 can be ignored

$$\begin{aligned} NT(N) &= (N+1)T(N-1) + 2cN \\ \frac{T(N)}{N+1} &= \frac{T(N-1)}{N} + \frac{2c}{N+1} \\ \frac{T(N-1)}{N} &= \frac{T(N-2)}{N-1} + \frac{2c}{N} \\ &\cdots\cdots \\ \frac{T(2)}{3} &= \frac{T(1)}{2} + \frac{2c}{3} \end{aligned} \tag{9}$$

By taking the sum of Equation 9 together, we have

$$
\begin{aligned}
T(N) &= (N+1)\left(\frac{T(1)}{2} + 2c\sum_{i=3}^{N+1}\frac{1}{i}\right) \\
&\le (N+1)2c\ln N \in O(N\log N)
\end{aligned}
\tag{10}
$$

□

Exercise 6 Rank the following functions by order of growth with explanations: that is, find an arrangement $g_1, g_2, \ldots, g_k$ of the functions $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots, g_{k-1} = \Omega(g_k)$. Partition your list into equivalence classes such that functions $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. Use symbols "=" and "$\prec$" to order these functions appropriately. (ps: $\log n$ refers to $\log_2 n$.)

$$
\begin{array}{ccccc}
2^{\log n} & (\log n)^{\ln n} & n^2 & n! & (n-1)! \\
2^n & n^3 & \log^2 n & e^n & 2^{2^n} \\
\log\log n & (n+1)\cdot 2^n & n & \log(n^2-n) & 2^{\ln n}
\end{array}
$$

Solution.

$$
\begin{aligned}
&\log\log n \prec \log(n^2-n) \prec \log^2 n \prec 2^{\log n} = n = 2^{\ln n} \prec n^2 \prec n^3 \\
&\prec ()\log n)^{\ln n} \prec 2^n = e^n \prec (n+1)\cdot 2^n \prec (n-1)! \prec n! \prec 2^{2^n}
\end{aligned}
\tag{11}
$$

□