Homework #3
Introduction to Algorithms/Algorithms 1
600.363/463
Spring 2013

Yifan Ge

February 19, 2013

1 Problem 1 (20 points)

Assume that there are N robots $R_1, \ldots, R_N$ and N tasks, $T_1, \ldots, T_N$. Typically, robot R_i performs task T_i. Also, the power of robots grows with their index. Thus, robot R_i can perform any task T_j without a failure for $j \leq i$ but it will fail if $i < j$. As a result of a program bug, all tasks have been permuted randomly and then assigned to robots. That is, R_i performs task $T_{\pi(i)}$, where π is a random permutation of the numbers $\{1, 2, \ldots, N\}$.

1. What is the expected number of robots that perform their original tasks?
Solution: Let random variable $X_i = 1$ if the robot R_i performs its original task, and $X_i = 0$ if R_i does not perform its original task. We have

$$P\{X_i = 1\} = \frac{1}{N} \tag{1}$$

$$E[X_i] = \frac{1}{N} \tag{2}$$

We assume the number of robots that perform their original task to be a random variable Y, we have $Y = \sum_{i=1}^{N} X_i$. Then the expectation of Y is:

$$E[Y] = E[\sum_{i=1}^{N} X_i] \tag{3}$$

$$= \sum_{i=1}^{N} E[X_i] \tag{4}$$

$$= 1 \tag{5}$$

2. What is the expected number of failures?
Solution: Let random variable $X_i = 1$ denote that the robot R_i encounters a failure. We have:

$$P\{X_i = 1\} = \frac{N-i}{N} \tag{6}$$
$$E[X_i] = \frac{N-i}{N} \tag{7}$$

We assume the number of failures to be a random variable Y, and $Y = \sum_{i=1}^{N} X_i$. The expectation of Y is:

$$\begin{aligned} E[Y] &= E[\sum_{i=1}^{N} X_i] && (8)\\ &= \sum_{i=1}^{N} E[X_i] && (9)\\ &= \sum_{i=1}^{N} \frac{N-i}{N} && (10)\\ &= \frac{N-1}{2} && (11) \end{aligned}$$

2 Problem 2 (20 points)

2.1 (10 points)

Resolve the following recurrences. Use Master theorem, if applicable. In all examples assume that $T(1) = 1$. To simplify your analysis, you can assume that $n = a^k$ for some a, k.

1. $T(n) = 5T(n/2) + \sqrt{n}$
Solution: As $\sqrt{n} = n^{0.5} = O(n^{\log_2 5 - \epsilon})$, with $\epsilon = 0.1$, $T(n) = \Theta(n^{\log_2 5})$.

2. $T(n) = T(n/2) + 10$
Solution: As $10 = \Theta(n^{\log_2 1}) = \Theta(1)$, $T(n) = \Theta(n^{\log_2 1} \lg n) = \Theta(\lg n)$.

3. $T(n) = 200T(\sqrt{n}) + n$
Solution: As $n = a^k$, we have $k = \log_a n$. Rewrite the recurrence as follows:

$$T(a^k) = 200T(a^{\frac{1}{2}k}) + a^k \tag{12}$$

Let $S(k) = T(n) = T(a^k)$, we get

$$S(k) = 200T(a^{\frac{1}{2}k}) + a^k \tag{13}$$
$$= 200S(\frac{1}{2}k) + a^k \tag{14}$$

Now we can use Master theorem to resolve this recurrence for $S(k)$. Here we have $f(n) = a^k = \Omega(k^{\log_2 200+\epsilon})$ and if we choose $c = 200$ we can satisfy $200a^{k/2} \leq ca^k$ with sufficiently large k. As a result we get $S(k) = \Theta(a^k)$. Substitute $S(k)$ with $T(n)$ and a^k with n, we have $T(n) = \Theta(n)$.

4. $T(n) = 12T(n/12) + n^2$
   Solution: As $f(n) = n^2 = \Omega(n^{\log_{12} 12+\epsilon}) = \Omega(n^\epsilon)$ with $\epsilon = 0.1$, and if we choose $c = \frac{1}{10}$, $12f(n/12) \leq cf(n)$ satisfies for all sufficiently large n. So $T(n) = \Theta(f(n)) = \Theta(n^2)$.

5. $T(n) = T(n/200) + n^{200}$
   Solution: As $f(n) = n^{200} = \Omega(n^{\log_{200} 1+\epsilon}) = \Omega(n^\epsilon)$ with $\epsilon = 0.1$, and if we choose $c = \frac{1}{20^{200}-1}$, $f(\frac{n}{200}) \leq cf(n)$ will satisfy for sufficiently large n. So we have $T(n) = \Theta(f(n)) = \Theta(n^{200})$.

6. $T(n) = n + T(n-1)$
   Solution: We cannot use Master theorem for this problem. We can use the substitution method to solve the recurrence:

$$T(n) = n + T(n-1) \tag{15}$$
$$= n + (n-1) + T(n-2) \tag{16}$$
$$= n + (n-1) + (n-2) + T(n-3) \tag{17}$$
$$= n + (n-1) + (n-2) + \cdots + (n-k+1) + T(n-k) \tag{18}$$
$$= \sum_{k=2}^{n} k + T(1) \tag{19}$$
$$= \frac{(n+2)(n-1)}{2} + 1 \tag{20}$$
$$= \Theta(n^2) \tag{21}$$

7. $T(n) = 50T(n/45) + n^3$
   Solution: Applying Master theorem, $f(n) = n^3 = \Omega(n^{\log_{45} 50+\epsilon})$ with $\epsilon = 0.1$, and choosing $c = \frac{1}{50 \cdot 45^3 - 1}$ satisfies $50f(\frac{n}{45}) \leq cf(n)$ for sufficiently large n. As a result, $T(n) = \Theta(f(n)) = \Theta(n^3)$.

8. $T(n) = \sqrt{n}T(n/2)$
   Solution: Master theorem is not applicable for this problem. Using substitution method:

$$
\begin{aligned}
T(n) &= \sqrt{n}T(n/2) && (22)\\
&= \sqrt{n}\sqrt{n/2}T(n/4) && (23)\\
&= \sqrt{n}\sqrt{n/2}\ldots\sqrt{n/2^{k-1}}T(n/2^k) && (24)\\
&= \sqrt{n(n/2)\ldots(n/2^{k-1})}T(n/2^k) && (25)\\
&= \sqrt{n(n/2)\ldots(n/2^{\log_2 n-1})}T(1) && (26)\\
&= \sqrt{\frac{n^{\log_2 n}}{2^{(\log_2 n-1)(\log_2 n)/2}}} && (27)\\
&= \sqrt{\frac{n^{\log_2 n}}{n^{(\log_2 n-1)/2}}} && (28)\\
&= n^{\frac{1}{4}\log_2 n+\frac{1}{4}} && (29)
\end{aligned}
$$

9. $T(n) = 5T(n/4) + n$
   Solution: Using Master theorem, $f(n) = n = O(n^{\log_4 5-\epsilon})$ with $\epsilon = 0.1$. So $T(n) = \Theta(n^{\log_4 5})$.

10. $T(n) = 9T(n/3) + n^2$
    Solution: Using Master theorem, $f(n) = n^2 = \Theta(n^{\log_3 9}) = \Theta(n^2)$, so $T(n) = \Theta(n^2 \lg n)$.

2.2 (10 points)

Imagine abstract problem A with the input of size n. You and your friends came up with the following four algorithms that solve A:

1. Algorithm X divides A into 5 subproblems of half the size, recursively solves each subproblem and then combines the solutions in quadratic time.

2. Algorithm Y divides A into 1 subproblem of size $n-2$, recursively solves the subproblem and then derives the solution in linear time.

3. Algorithm Z divides A into 2 subproblems of size $n-1$, recursively solves each subproblem and then combines the solutions in constant time.

4. Algorithm W divides A into 100 subproblems of size $n/1000$, recursively solves each subproblem and then combines the solutions in linear time.

Which algorithm you should choose and why?
Solution: The time complexities in recurrence of the four algorithms are:

$$T_X(n) = 5T_X(n/2) + c_1 n^2 \tag{30}$$
$$T_Y(n) = T_Y(n-2) + c_2 n \tag{31}$$
$$T_Z(n) = 2T_Z(n-1) + c_3 \tag{32}$$
$$T_W(n) = 100T_W(n/1000) + c_4 n \tag{33}$$

c_1, c_2, c_3, c_4 are constants.
Solve the four recurrences:

1. For Algorithm X, we can use Master theorem to solve for $T_X(n)$. As $c_1 n^2 = O(n^{\log_2 5 - \epsilon})$ with $\epsilon = 0.1$, apply the Master theorem we get $T_X(n) = \Theta(n^{\log_2 5})$.

2. For Algorithm Y, we use the substitution method to solve the recurrence.

$$\begin{aligned} T_Y(n) &= T_Y(n-2) + c_2 n && (34) \\ &= T_Y(n-4) + c_2(n + (n-2)) && (35) \\ &= c_2(\frac{(n+2)n}{4}) + O(1) && (36) \\ &= \Theta(n^2) && (37) \end{aligned}$$

   In equation (36) we assume n is even and $T_Y(0) = O(1)$.

3. For Algorithm Z, use the substitution method:

$$\begin{aligned} T_Z(n) &= 2T_Z(n-1) + c_3 && (38) \\ &= 2(2T_Z(n-2) + c_3) + c_3 && (39) \\ &= 2^k T_Z(n-k) + c_3(1 + 2 + 2^2 + \cdots + 2^k) && (40) \\ &= 2^{n-1} T_Z(1) + c_3(1 + 2 + 2^2 + \cdots + 2^{n-1}) && (41) \\ &= O(2^{n-1}) + (2^n - 1)c_3 && (42) \\ &= \Theta(2^n) && (43) \end{aligned}$$

4. For Algorithm W, we can use the Master theorem. $f(n) = c_4 n = \Omega(n^{\log_{1000} 100 + \epsilon})$, and choosing $c = \frac{1}{1000}$ satisfies that $100 f(n/1000) \leq c f(n)$ with sufficiently large n. As a result, $T_W(n) = \Theta(f(n)) = \Theta(n)$.

Comparing the time complexities of the four algorithms, we should choose Algorithm W.