CPS 130 Homework Solutions

Guangwei Yuan
Fall 1999

HOMEWORK 2

2.1: By using the iteration method, we have

\[ T(n) = \sqrt{n} I(\sqrt{n}) + n \log n \]
\[ = n^{k=1 \frac{1}{2}} T(n^{\frac{1}{2}}) + n \log n \sum_{i=0}^{k-1} \frac{1}{2^i} \]
\[ = n^{1-\frac{1}{2^k}} T(n^{\frac{1}{2^k}}) + 2(1 - \frac{1}{2^k}) n \log n \]

Let \( n^{\frac{1}{2^k}} = 2 \), we have \( \frac{1}{2^k} = \log_2 2 \), and

\[ T(n) = O(n) + \Theta(n \log n) \]
\[ = \Theta(n \log n) \]

2.2: By using the Master method, we have

\[ T(n) = 2T(n/2) + n^3 \]
\[ = \Theta(n^3) \]

\[ T(n) = 2T(n/4) + \sqrt{n} \]
\[ = \Theta(\sqrt{n} \log n) \]

\[ T(n) = 7T(n/2) + n^2 \]
\[ = \Theta(n \log_2 7) \]

For the other three recurrences, Master method could not be applied. The harmonic number
introduced in page 44 of CLR is needed for the last two.

\[
T(n) = T(\sqrt{n}) + 1 \\
= \Theta(\lg \lg n)
\]

\[
T(n) = 2T(n/2) + n/\lg n \\
= \Theta(n) + n \sum_{k=1}^{\lg n} \frac{1}{k} \\
= \Theta(n) + \Theta(n \ln \lg n) \\
= \Theta(n \lg \lg n)
\]

\[
T(n) = T(n-1) + 1/n \\
= \Theta(1) + \sum_{k=1}^{n} \frac{1}{k} \\
= \Theta(1) + \Theta(\ln n) \\
= \Theta(\lg n)
\]

2.3:

- When \( k = n \), we have \( n \) lists of size 1. Time taken is then:
  \[ O(n) + nO(1) + (1 + 2 + 3 + \cdots + n) = O(n^2) \]

- Let \( T(n) \) be the time complexity of the algorithm for a problem size \( n \).
  The amount of work done for a problem of size \( n \) is:

  1. make the \( k \) lists: \( O(n) \)
  2. sort the \( k \) lists of size \( \frac{n}{k} \): \( kT(\frac{n}{k}) \)
  3. merge the lists:
     \[
     \frac{2n}{k} + \frac{3n}{k} + \cdots + \frac{kn}{k} \\
     = \frac{n}{k}(2 + 3 + 4 + \cdots + k) \\
     = \frac{n}{k}O(k^2) \\
     = O(nk)
     \]

  The total work done is: \( O(n) + kT(\frac{n}{k}) + O(nk) \), which can be expressed as the following recurrence:

  \[ T(n) = kT(\frac{n}{k}) + O(nk) \]

- The solution to the recurrence is: \( O(kn \log_k n) \). Note we cannot assume \( k \) to be constant as it varies from 1 to \( n \).

2.4 We have \( \alpha \leq 1 - \alpha \), given \( 0 < \alpha \leq 1/2 \). The minimum depth of a leaf in the recursion tree is the length of the path always with the \( \alpha \) part of the splits at each level. It is approximately: \( \log_{1/\alpha} n = -\lg n / \lg \alpha \). The maximum depth is the length of the path always with the \( (1 - \alpha) \) part. It is approximately: \( \log_{1/(1-\alpha)} n = -\lg n / \lg(1 - \alpha) \).

2.5 Let \( m \) be the number of inputs which could be sorted in linear time. The problem
is asking if it is possible for $m$ to be more than $n!$, or just more than $n!/n$, or more than $n!/2^n$. Here we consider a decision tree that sorts $n$ elements, in which each leaf represents a permutation of $n$ elements. For the input which could be sorted in linear time, the depth of the leaf representing it is therefore $O(n)$. Then we have: $m \leq 2^{O(n)} = O(2^n)$, since the decision tree is a binary tree. Notice the fact that $n! = \omega(2^n)$, $n!/n = \omega(2^n)$, and $n!/2^n = \omega(2^n)$, so there is no comparison sort whose running time is linear for more than even a fraction $1/2^n$ of the $n!$ inputs.