CS412 Algorithms: Design & Analysis

Spring 2024

Dhanani School of Science and Engineering

Habib University

Practice Problems

Week 4

1. Although merge sort runs in $\Theta(n\lg n)$ worst-case time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort can make it faster in practice for small problem sizes on many machines. Thus, it makes sense to **coarsen** the leaves of the recursion by using insertion sort within merge sort when subproblems become sufficiently small. Consider a modification to merge sort in which $n = k$ sublists of length k are sorted using insertion sort and then merged using the standard merging mechanism, where k is a value to be determined.

   (a) Show that insertion sort can sort the $n = k$ sublists, each of length k, in $\Theta(nk)$ worst-case time.

   (b) Show how to merge the sublists in $\Theta(n\lg(n/k))$ worst-case time.

   (c) Given that the modified algorithm runs in $\Theta(nk + n\lg(n/k))$ worst-case time, what is the largest value of k as a function of n for which the modified algorithm has the same running time as standard merge sort, in terms of, $\Theta-$notation?

   (d) How should we choose k in practice?

   Solution: a) The time for insertion sort to sort a single list of length k is $\Theta(k^2)$, so, n/k of them will take time $\Theta(\frac{n}{k}k^2) = \Theta(nk)$.

   b) Suppose we have coarseness k. This meas we can just start using the usual merging procedure, except starting it at the level in which each array has size at most k. This means that the depth of the merge tree is $\lg(n) - \lg(k) = \lg(n/k)$. Each level of merging is still time cn, so putting it together, the merging takes time $\Theta(n\mathbf{lg}(n/k))$.

   c) Viewing k as a function of n, as long as $k(n) \in O(\lg(n))$, it has the same asymptotics. In particular, for any constant choice of k, the asymptotics are the same.

   d) If we optimize the previous expression using our calculus 1 skills to get k, we have that $c_1 n - \frac{nc_2}{k} = 0$ where c_1 and c_2 are the coeffients of nk and $n\lg(n/k)$ hidden by the asymptotics notation. In particular, a constant choice of k is optimal. In practice we could find the best choice of this k by just trying and timing for various values for sufficiently large n.

2. What does FIND-MAXIMUM-SUBARRAY return when all elements of A are negative?

   Solution: It will return the least negative position. As each of the cross sums are computed, the most positive one must have the shortest possible lengths. The algorithm doesn't consider length zero sub arrays, so it must have length 1.

3. Write pseudocode for the brute-force method of solving the maximum-subarray problem. Your procedure should run in $\Theta(n^2)$.

Solution:

```
left = 1
right = 1
max = A[1]
curSum = 0
for i = 1 to n do // Increment left end of subarray
    curSum = 0
    for j = i to n do // Increment right end of subarray
        curSum = curSum + A[j]
        if curSum > max then
            max = curSum
            left = i
            right = j
        end if
    end for
end for
```

4. Prove by induction that the *ith* Fibonacci number satisfies the equality $F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$ where ϕ is the golden ratio and $\hat{\phi}$ is its conjugate.

Solution: First, we show that $1 + \phi = \frac{6+2\sqrt{5}}{4} = \phi^2$. So for ever i, $\phi^{i-1} + \phi^{i-2} = \phi^{i-2}(\phi+1) = \phi^i$. Similarly for $\hat{\phi}$. For $i = 0$, $\frac{\phi^0 - \hat{\phi}^0}{5} = 0$. For $i = 1$, $\frac{\frac{1+\sqrt{5}}{2} - \frac{1-\sqrt{5}}{2}}{\sqrt{5}} = \frac{\sqrt{5}}{\sqrt{5}} = 1$. Then, by induction, $F_i = F_{i-1} + F_{i-2} = \frac{\phi^{i-1}+\phi^{i-2}-(\hat{\phi}^{i-1}+\hat{\phi}^{i-2})}{\sqrt{5}} = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$.

5. Use the master method to give tight asymptotic bounds for the following recurrences.
   1. $T(n) = 2T(n/4) + 1$
   2. $T(n) = 2T(n/4) + \sqrt{n}$
   3. $T(n) = 2T(n/4) + n$
   4. $T(n) = 2T(n/4) + n^2$

Solution: The master theorem is a tool used to analyze recurrence relations of the form:

$$T(n) = aT(n/b) + f(n)$$

Here, a and b are positive constants, and $f(n)$ is an asymptotically positive function.

The master theorem provides a framework for determining the asymptotic behavior of the recurrence relation in terms of Big O notation. The theorem has three cases, and the solution depends on comparing $f(n)$ with a certain function related to $n^{\log_b^a}$

The master theorem is stated as follows:

(a) If $f(n) = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$ then $T(n) = \Theta(n^{\log_b a})$.

(b) If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.

(c) If $f(n) = \Omega(n^{\log_b a+\epsilon})$ for some $\epsilon > 0$ and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta f(n)$.

Now, let's analyze each of the given recurrences using the master theorem:
1. $T(n) = 2T(n/4) + 1$
Here, $a = 2$, $b = 4$, and $f(n) = 1$. The recurrence follows the form of the master theorem with $\log_b a = \log_4 2 = 1/2$.
Comparing $f(n) = 1$ with $n^{\log_b a-\epsilon} = n^{1/2-\epsilon}$ for any $\epsilon > 0$, we see that $f(n) = O(n^{1/2-\epsilon})$.
Therefore, the first case of the master theorem applies.
The solution is $T(n) = \Theta(n^{\log_b a}) = \Theta(n^{1/2})$.

2. $T(n) = 2T(n/4) + \sqrt{n}$
Here, $a = 2$, $b = 4$, and $f(n) = \sqrt{n}$. The recurrence also follows the form of the master theorem with $\log_b a = \log_4 2 = 1/2$.
Comparing $f(n) = \sqrt{n}$ with $n^{\log_b a} = n^{1/2}$, we see that $f(n) = \Theta(n^{\log_b a})$. Therefore, the second case of the master theorem applies.
The solution is $T(n) = \Theta(f(n) \log n) = \Theta(\sqrt{n} \log n)$.

3. $T(n) = 2T(n/4) + n$
Here, $a = 2$, $b = 4$, and $f(n) = n$. The recurrence follows the form of the master theorem with $\log_b a = \log_4 2 = 1/2$.
Comparing $f(n) = n$ with $n^{\log_b a+\epsilon} = n^{1/2+\epsilon}$ for any $\epsilon > 0$, we see that $f(n) = \Omega(n^{1/2+\epsilon})$.
Therefore, the third case of the master theorem applies.
The solution is $T(n) = \Theta(f(n)) = \Theta(n)$.

4. $T(n) = 2T(n/4) + n^2$
Here, $a = 2$, $b = 4$, and $f(n) = n^2$. The recurrence follows the form of the master theorem with $\log_b a = \log_4 2 = 1/2$.
Comparing $f(n) = n^2$ with $n^{\log_b a} = n^{1/2}$, we see that $f(n) = \Omega(n^{1/2+\epsilon})$ for any $\epsilon > 0$.
Therefore, the third case of the master theorem applies.
The solution is $T(n) = \Theta(f(n)) = \Theta(n^2)$.