CS 677 Homework Assignment 02

Hai Nguyen

Semtember 11, 2018

1. Consider the following recursive algorithm.

   ALGORITHM *Min1*(A[0..n-1])
   //Input: An array A[0..n-1] of integer number
   if n = 1 return A[0]
   else *temp* $\leftarrow$ *Min1*(A[0..n-2])
       if *temp* $\leq$ A[n-1] **return** *temp*
       else return A[n-1]

   (a) What does this algorithm compute?

   Minimum of an array of integer numbers.

   (b) Set up a recurrence relation for the algorithm's basic operation count and solve it

   ALGORITHM *Min1*(A[0..n-1])
   if n = 1 return A[0] [Constant time c1]
   else *temp* $\leftarrow$ *Min1*(A[0..n-2]) [c2 + T(n-1)]
       if *temp* $\leq$ A[n-1] **return** *temp* [Constant time c3]
       else return A[n-1] [Constant time c4]
   Recurrent relationship:

$$
\begin{aligned}
T(1) &= c1 \\
T(n) &= c + T(n-1)
\end{aligned}
$$

   Solve it:

$$
\begin{aligned}
T(n) &= c + T(n-1) \\
&= T(1) + (n-1)c \\
&= \Theta(n)
\end{aligned}
$$

2. Solve the following recurrences using the method of your choice.

   - $T(n) = 2T(\frac{n}{2}) + n^3$
     Iteration method:
     Assume: $n = 2^k$

$$
\begin{aligned}
T(n) &= 2T(\frac{n}{2}) + n^3 = 2(2T(\frac{n}{4}) + (\frac{n}{2})^3) + n^3 = 4T(\frac{n}{4}) + n^3(1 + \frac{1}{4}) \\
&= 4(2T(\frac{n}{8}) + (\frac{n}{4})^3) + n^3(1 + \frac{1}{4}) = 2^3T(\frac{n}{2^3}) + n^3(1 + \frac{1}{4} + \frac{1}{4^2}) \\
&= ... \\
&= 2^iT(\frac{n}{2^i}) + n^3(1 + \frac{1}{4} + \frac{1}{4} + ... + \frac{1}{4^{i-1}}) \\
&= nT(1) + n^3\sum_{i=1}^{lgn}\frac{1}{4^{i-1}} \\
&= c_1n + c_2n^3 \\
&= \Theta(n^3)
\end{aligned}
$$

- $T(n) = T(\sqrt{n}) + 1$
  Rename: $m = lgn \rightarrow n = 2^m$
  $T(2^m) = T(2^{m/2}) + 1$
  Rename: $S(m) = T(2^m)$

$$
\begin{aligned}
S(m) &= S(m/2) + 1 \\
&= S(m/4) + 2 \\
&= S(m/2^i) + i \\
&= \ldots \\
&= S(1) + lgm \\
&= \Theta(lgm)
\end{aligned}
$$

  $S(m) = S(m/2) + 1 = \Theta(lgm)$

  $T(n) = T(2^m) = S(m) = \Theta(lgm) = \Theta(lglgn)$

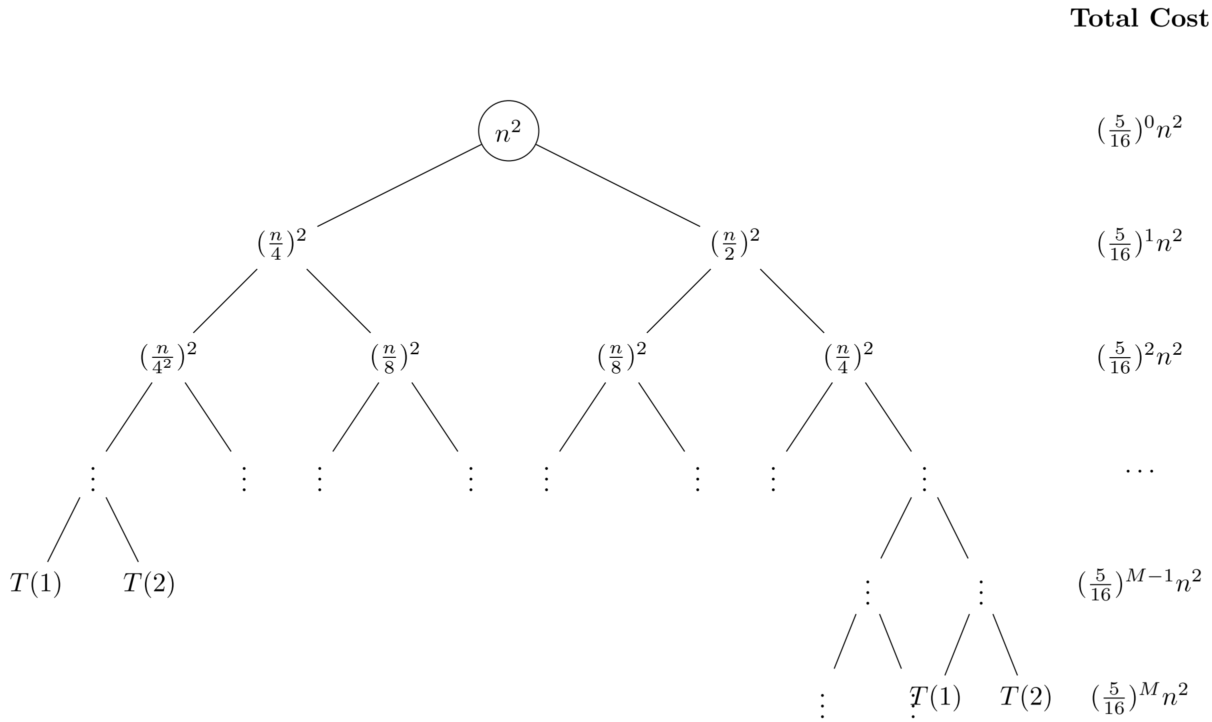
- $T(n) = 3T(\frac{n}{2}) + nlgn$
  Master's method: $a = 3, b = 2, f(n) = nlgn$.
  Compare $n^{\log_b a} \approx n^{1.58}$ and $nlgn$, for some constant $\epsilon > 0$:

$$f(n) = nlgn = O(n^{1.58-\epsilon})$$

  $\rightarrow$ Case 1:

$$T(n) = \Theta(n^{\log_2 3})$$

3. - Draw the recursion tree for $T(n) = T(n/4) + T(n/2) + n^2$ and provide a tight asymptotic bound on its solution.



Unbalanced tree, left branch ends sooner. Tight bound is a new balanced tree with the left branch replaced by the reflection of the old right branch.

$$T(n) \leq n^2(1 + (\frac{5}{16})^2 + ... + (\frac{5}{16})^M)$$

With:

$$\frac{n}{2^M} = 2$$

Because $\frac{5}{16} < 1$, we have:

$$\begin{aligned} T(n) &\leq n^2(1 + (\frac{5}{16})^2 + ... + (\frac{5}{16})^M) \\ &\leq n^2 \frac{1}{1 - \frac{5}{16}} \\ &= \mathbf{O(n^2)} \end{aligned}$$

- Use the iteration method to solve the following recurrence:
  $T(n) = 4T(n/2) + n$

Assume: $n = 2^k$

$$
\begin{aligned}
T(n) &= 4T(n/2) + n \\
&= 4(4T(n/2^2) + n/2) + n \\
&= 4^2T(\frac{n}{2^2}) + 4\frac{n}{2} + n \\
&= 4^3T(\frac{n}{2^3}) + 4^2\frac{n}{2^2} + 4\frac{n}{2} + n \\
&= \dots \\
&= 4^iT(\frac{n}{2^i}) + n(1 + 2^1 + ... + 2^{i-1}) \\
&= \dots \\
&= 4^{lgn}T(1) + n\sum_{i=0}^{lgn-1} 2^i \\
&= n^2T(1) + n(n-1) \\
&= \mathbf{\Theta(n^2)}
\end{aligned}
$$

4. Considering the following recursive algorithm:
   ALGORITHM Q(n)

   //Input: A positive integer n
   if $n = 1$

   return 1

   else

   return **Q(n-1) + 2n - 1**

   - Set up a recurrence relation for this function's value and solve it to determine what this algorithm computes: Recurrence relation:

   $$
   \begin{aligned}
   Q(n) &= Q(n-1) + 2n - 1 \text{ With } \forall n > 1 \\
   Q(1) &= 1
   \end{aligned}
   $$

   Solve it:

   $$
   \begin{aligned}
   Q(n) &= Q(n-1) + 2n - 1 \text{ With } \forall n > 1 \\
   &= 1 + 3 + 5 + ... + 2n - 1 \\
   &= \mathbf{n^2}
   \end{aligned}
   $$

   - Set up a recurrence relation for the number of multiplications made by this algorithm and solve it.
   ALGORITHM Q(n)

     //Input: A positive integer n
     if $n = 1$

     return 1

     else

     return **Q(n-1) + 2n - 1**

Recurrence relation for the number of multiplications:

$$\begin{aligned} M(1) &= 0 \\ M(n) &= M(n-1) + c \end{aligned}$$

Solve it:

$$\begin{aligned} M(n) &= M(n-1) + 1 \\ &= M(1) + (n-1)1 \\ &= \mathbf{\Theta(n)} \end{aligned}$$

The total number of multiplications made: $\mathbf{M(n) = n - 1}$

5. Consider the following algorithm.

   ALGORITHM *Mystery*(n)

   //Input: A non-negative integer n

   $S \leftarrow 0$

   for $i \leftarrow 1$ **to** n **do**

       $S \leftarrow S + i * i$

   return S

   - What does this algorithm compute?

     $$\mathbf{\sum_{i=1}^{n} i^2}$$

   - Compute the running time of this algorithm.

     $S \leftarrow 0$ [Cost: c1, Times: 1]
     for $i \leftarrow 1$ **to** n **do** [Cost: c2, Times: n + 1]
         $S \leftarrow S + i * i$ [Cost: c3, Times: n]
     return S [Cost:c4, Times:1]

     Total cost: $c1 + c2 \times (n+1) + c3 \times n + c4 = \mathbf{\Theta(n)}$