Homework 1

1. Observe that the while loop of lines 4-7 of the INSERTION-SORT procedure given below uses a linear search to scan (backward) through the sorted subarray A[1..j-1]. Can we use a binary search instead to improve the overall worst-case running time of insertion sort to $\theta(nlgn)$?

   Answer: No. After finding the element, it still needs to shift all elements to the right which is linear in the worst case.

2. Let $f(n)$ and $g(n)$ be asymptotically nonnegative functions. Using the basic definition of θ-notation, prove that $max(f(n), g(n)) = \theta(f(n) + g(n))$

   Answer: Let $f(n) = \max(f(n), g(n))$. We need to prove that

   $c_1 \cdot (f(n)+g(n)) \leq f(n) \leq c_2 \cdot (f(n)+g(n))$ for some positive const $c_1, c_2, n \geq n_0$.

   $$c_1 \cdot \left(\frac{f(n)+g(n)}{f(n)}\right) \leq \frac{f(n)}{f(n)} \leq c_2 \cdot \left(\frac{f(n)+g(n)}{f(n)}\right)$$

   $$c_1 + c_1 \frac{g(n)}{f(n)} \leq 1 \leq c_2 + c_2 \cdot \frac{g(n)}{f(n)} \quad // \text{ as } g(n) \leq f(n), \frac{g(n)}{f(n)} \leq 1$$

   So, $c_2 = 1$, $c_1 = 1/2$, $n \geq n_0 = 1$.

3. Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?

   Answer:

   $2^{n+1} = 2 \cdot 2^n \leq c \cdot 2^n$
   $c = 2$. $n_0 = 1$

   $2^{2n} = 2^n \cdot 2^n \leq c \cdot 2^n$
   $2^n \leq c$
   You cannot find c s.t. $2^n \leq c$. for $n \geq n_0$

4. Use the substituion method to show that the solution of $T(n) = T(n-1) + n$ is $O(n^2)$.

   Answer:

   Guess: $T(n) = O(n^2) \Rightarrow T(n) \leq c \cdot n^2$
   Assume $T(n-1) \leq c(n-1)^2$

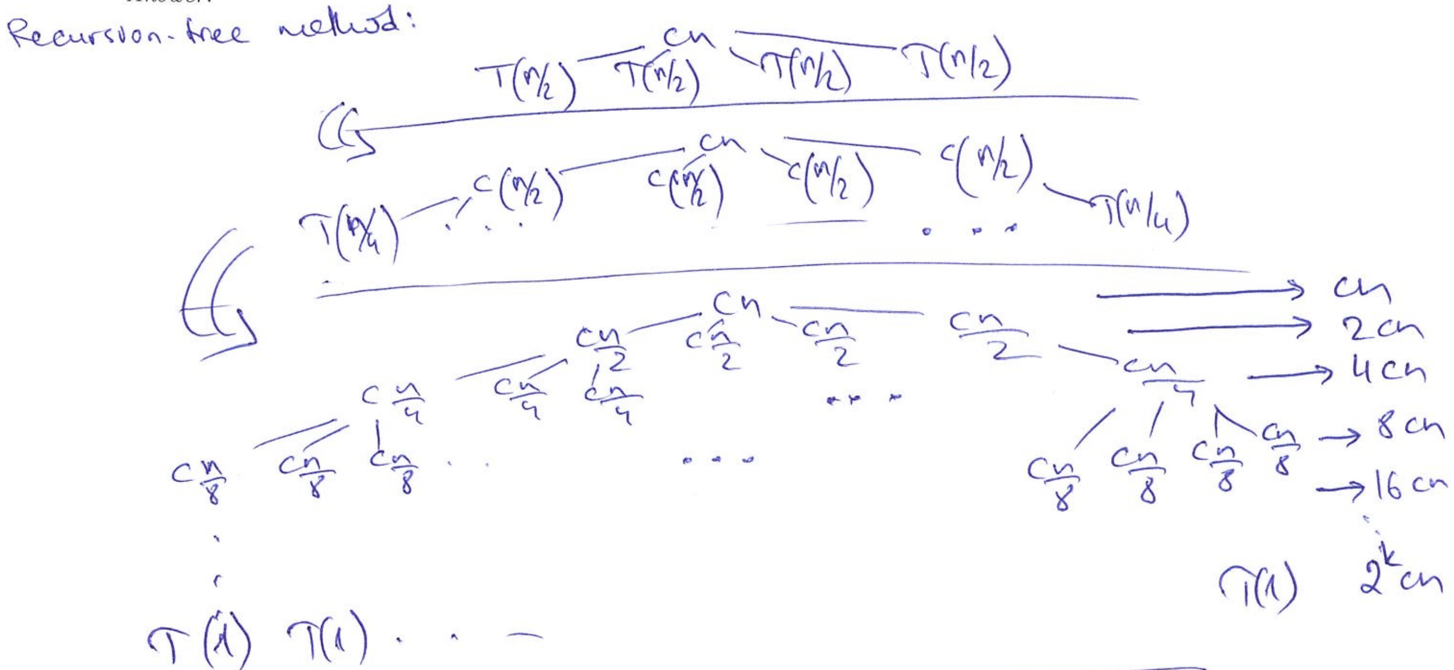
   $$\begin{aligned} T(n) &= T(n-1)+n \\ &\leq c(n-1)^2+n \\ &= c(n^2-2n+1)+n \\ &= cn^2-2cn+c+n \\ &= cn^2-(2cn-n-c) \end{aligned}$$

   $T(n) \leq cn^2$ if $2cn-n-c \geq 0$
   $n(2c-1) \geq c$

   for $n_0=1$ ✓ $2c-1 \geq c$
   $c \geq 1$ ✓

5. Use the recursion-tree method for $T(n) = 4T(n/2) + cn$, where c is a constant, to give a good asymptotic upper bound on its solution. Use the substitution method to verify your bound.
Answer:

Recursion-tree method:



$$T(1) = T\left(\frac{n}{2^k}\right) \Rightarrow 1 = \frac{n}{2^k} \Rightarrow 2^k = n \Rightarrow \boxed{\lg n = k}$$

$$T(n) = cn + 2cn + 4cn + 8cn + \dots \quad 2^{\lg n} cn$$

$$= \sum_{i=0}^{\lg n} 2^i \cdot cn = cn \sum_{i=0}^{\lg n} 2^i = cn \cdot \left(\frac{2^{\lg n + 1} - 1}{2-1}\right) = \frac{cn \cdot (2 \cdot n - 1)}{1} =$$

$$= 2cn^2 - cn = O(n^2)$$

Subs. method: $T(n) = O(n^2) \Rightarrow T(n) \leq dn^2$ for some + c. n_0.

Assume $T(n/2) \leq d\left(\frac{n}{2}\right)^2$

$T(n) \leq 4 \cdot d\left(\frac{n}{2}\right)^2 + cn$

$= 4d\frac{n^2}{4} + cn = 1$

$= dn^2 + cn$

positive

we cannot conclude that

$T(n) \leq dn^2$.

$T(n) \leq d_1 n^2 - d_2 n$

$T(n/2) \leq d_1\left(\frac{n}{2}\right)^2 - d_2\left(\frac{n}{2}\right)$

$T(n) \leq 4\left(d_1\left(\frac{n}{2}\right)^2 - d_2\left(\frac{n}{2}\right)\right) + cn$

$= d_1 n^2 - 2d_2 n + cn$

$= \underbrace{d_1 n^2 - d_2 n}_{\text{we need this}} - d_2 n + cn$

$= d_1 n^2 - d_2 n - \underbrace{(d_2 n - cn)}_{\geq 0}$

$d_2 n - cn \geq 0$

$d_2 \geq c.$