Intro to Algorithms, COMP-160, Homework #1

Benjamin Tanen, 02/02/2016

5. **Use the substitution method to get:**

   (a) an upper bound for $T(n) = T(n{-}1) + n$
   Assume $T(n) = O(n^2)$
   For $k < n$, $T(k) \leq c_1 \cdot k^2$
   $T(n) = c_1 \cdot (n-1)^2 + n = c_1 \cdot n^2 - ((2c_1 - 1)n - c_1)$
   $((2c_1 - 1)n - c_1) \geq 0$
   $2c_1 n - n \geq c_1$
   $2c_1 \geq \frac{c_1}{n} + 1$
   In the worst case, where $n = 1$, we then get $c_1 \geq 1$
   Thus, we get $T(n) \leq c_1 \cdot n^2$ for $c_1 \geq 1$
   This proves $T(n) = O(n^2)$

   (b) a lower bound for $T(n) = T(n{-}1) + n$
   Assume $T(n) = \Omega(n^2)$
   For $k < n$, $T(k) \geq c_2 \cdot k^2$
   $T(n) = c_2 \cdot (n-1)^2 + n = c_2 \cdot n^2 - ((2c_2 - 1)n - c_2)$
   $((2c_2 - 1)n - c_2) \leq 0$
   $2c_2 n - n \leq c_2$
   $2c_2 \leq \frac{c_2}{n} + 1$
   $\lim_{n \to \infty} \frac{c_2}{n} = 0$ so in the worst case ($n \to \infty$), $c_2 \leq \frac{1}{2}$
   Thus, we get $c_2 \cdot n^2 \leq T(n)$ for $c_2 \leq \frac{1}{2}$
   This proves $T(n) = \Omega(n^2)$

   (c) an upper bound for $T(n) = T(n{-}1) + \log n$.
   Assume $T(n) = O(n \cdot \log n)$
   For $k < n$, $T(k) = cn \cdot \log n$
   $T(n) = c(n-1)\log(n-1) + \log n$
   We know $\log(n-1) \leq \log(n)$
   Thus $T(n) = c(n-1)\log(n-1) + \log n \leq c(n-1)\log(n) + \log(n)$
   $T(n) \leq cn \cdot \log(n) - [c \cdot \log(n) - \log(n)]$
   For $c \geq 1$, $c \cdot \log(n) - \log(n) \geq 0$
   $T(n) \leq cn \cdot \log(n) - [c \cdot \log(n) - \log(n)] \leq n \cdot \log(n)$ for $c \geq 1$
   Thus, we've shown $\exists c$ such that $T(n) \leq cn \cdot \log(n) - [c \cdot \log(n) - \log(n)] \leq n \cdot \log(n)$
   This shows an upper bound exists so $T(n) = O(n \cdot \log n)$