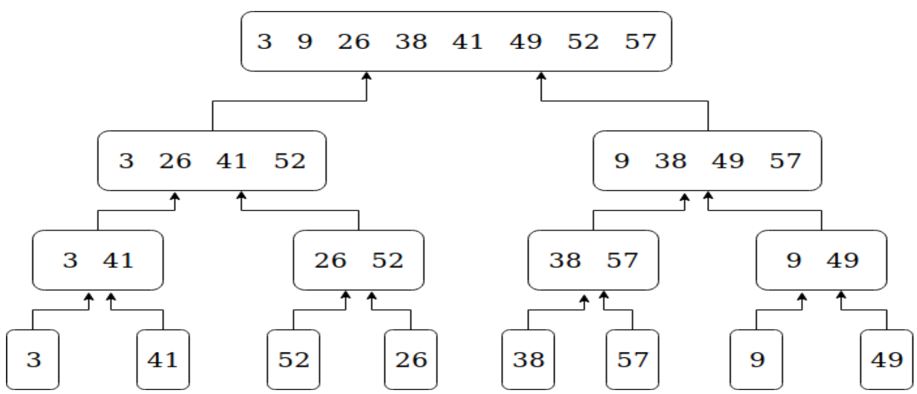
HW 1

Daniel Sawyer

September 8, 2018

Q1. Exercise 2.3-1: Using Figure 2.4 as a model, illustrate the operation of merge sort on the array A = {3, 41, 52, 26, 38, 57, 9, 49}



Q2. Exercise 2.3-6: Observe that the while loop of lines 5 – 7 of the INSERTION-SORT procedure in Section 2.1 uses a linear search to scan (backward) through the sorted subarray A[1...j-1]. Can we use a binary search instead of a linear search to improve the overall worst-case running time of insertion sort to Θ(nlgn)?

You could use a binary search but it would not speed up the sorting process since you would still have to rearrange the values into their proper place. So it would take just as long, if not longer, to do it. So in the end, it would still be O(n^2).

Q3. For the MERGE function, the sizes of the L and R arrays are one element longer than n 1 and n 2 , respectively. Can you rewrite the merge function with the size of L and R exactly equal to n_1 and n_2 ?

Yes you can. You do not need a sentinel value (max or min), however, you would double the amount of comparisons instead of just having 2 extra comparisons.

```
Merge(A, p, q, r):
n1 = q - p + 1
n2 = r - q
let L[1...n1] and R[1...n2]
for i=1 to n1:
   L[i] = A[p + i - 1]
for j=1 to n2:
   R[j] = A[q + j]
i = j = 1
for k=p to r:
   if i ≤ n1 ∧ L[i] ≤ R[j]
      A[k] = L[i]
      i++
   elif j ≤ n2
      A[k] = R[j]
      j++
```

Q4. Prove that $e^{1/n} \in$ O(n^t): t > 0

Assume that value t is an integer and $f(n) = e^{\frac{1}{n}}$, $g(n) = n^t$.

$n^t > e^{\frac{1}{n}}$

$ln(n^t) > ln(e^{\frac{1}{n}})$

$tln(n) > \frac{1}{n}$

$tln(n) \geq ln(n) > \frac{1}{n}$

$\lim_{n\to\infty} \frac{\frac{1}{n}}{ln(n)} = \lim_{n\to\infty} \frac{1}{nln(n)} = 0$

Therefore: Since $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, $f(n)$'s rate of change is less than $g(n)$'s $\forall$ $n \geq e$

Q5. Express the function $\frac{n^3}{100} - 50n - 100lgn$ in terms of Θ notation.

f(n) = $\frac{n^3}{100} - 50n - 100lg(n)$, g(n) = n^3

f(n) = O(g(n)) $\iff$ $\exists$ c $\wedge$ n_0 s.t. f(n) $\leq$ $cg(n)$ $\forall$ n $\geq$ n_0

$$\frac{n^3}{100} - 50n - 100lg(n) < \frac{n^3}{100}$$

$$< n^3$$

Therefore: $\exists$ positive constant $c_1 = 1 \wedge n_0 = 1$ s.t. c_1 n^3 $\geq \frac{n^3}{100} - 50n - 100lg(n)$ $\forall$ n $\geq$ n_0

f(n) = Ω(g(n)) $\iff$ $\exists$ c $\wedge$ n_0 s.t. f(n) $\geq$ $cg(n)$ $\forall$ n $\geq$ n_0

$\frac{n^3}{100} - 50n - 100lg(n) \geq 50n + 100n$

$= 150$n

Therefore: $\exists$ a positive $c_2 = \frac{1}{150}$ $\wedge$ $n_0 = 150$ s.t. $\frac{n^3}{100} - 50n - 100lg(n) \geq$ c_2n^3 $\forall$ n $\geq$ n_0

Since $c_2n^3 \leq$ $\frac{n^3}{100} - 50n - 100lg(n) \leq c_1n^3$ $\forall$ n $\geq$ n_0, we can say that $\frac{n^3}{100} - 50n - 100lg(n) = \Theta(n^3)$ for positive constants $c_1 = 1$, $c_2 = \frac{1}{150}$ $\wedge$ $n_0 = 150$ $\forall$ n $\geq$ n_0

Q6. Exercise 3.1-6 Prove that the running time of an algorithm is Θ(g(n)) if and only if its worst-case running time is O(g(n)) and best-case running time is Ω(g(n)).

The definition of Big O states that f(n) $\leq$ c_1g(n) $\forall$ n $\geq$ n_1 and the definition of Big Ω states that f(n) $\geq$ c_2g(n) $\forall$ n $\geq$ n_2. Therefore we can state the average running time, Big Θ, is g(n) $\forall$ n $\geq$ n_0 since c_2g(n) $\leq$ f(n) $\leq$ c_1g(n) with positive constants $c_1 \wedge c_2$ with $n_0 = \max(n_1, n_2)$ $\forall$ n $\geq$ n_0

Q7. Which is asymptotically larger: lgn or $\sqrt{n}$? Please explain your reason.

$\sqrt{n}$ grows at a faster rate than lg(n), we can prove this using L'Hopital's Rule, which states that taking the limit of the derivatives of the numerator and denominator will give you the limit if you have an indeterminate form. Since $\lim_{n\to\infty} \frac{lg(n)}{\sqrt{n}} = \infty$, we must take both derivatives. lg(n)' $= \frac{1}{nln(2)}$ and $\sqrt{n}$' $= \frac{1}{\sqrt{n}} \to \frac{\frac{1}{ln(2)n}}{\frac{1}{\sqrt{n}}} = \frac{\sqrt{n}}{ln(2)n}$. This limit is also indeterminate, and taking the derivatives once more gives us numerator $\frac{1}{\sqrt{n}}$ and denominator $ln(2)$ which gives us $\frac{1}{ln(2)\sqrt{n}}$. The $\lim_{n\to\infty} \frac{1}{ln(2)\sqrt{n}} = 0$. Therefore we can say that lg(n) $\in$ O($\sqrt{n}$) $\forall$ n $\geq$ some value n_0

Q8. Prove that $n^{lgc} \in \Omega(c^{lgn}$), where c is a constant and c $>$ 1.

There is an identity that states $n^{lgc} = c^{lgn}$, since f(n) and g(n) are equal, we can state that $n^{lgc} \in \Omega(c^{lgn})$ since kc^{lgn} $\leq n^{lgc}$ with k $= 1$, $n_0 = 1$ $\forall$ n $\geq n_0$

Q9. Use the definition of limits at infinity to prove $(lgx)^p \in$ o(x^p).

$(lgx)^p \leq x^p$ using L'Hopitals Rule.
$\lim_{n\to\infty} \frac{lgx^p}{x^p} \to$ p-th root $= \frac{lgx}{x}$
Taking the derivatives of numerator and denominator gives us $\frac{1}{ln(2)n}$
$\lim_{n\to\infty} \frac{1}{ln(2)n} = 0$, therefore we can say that $(lgx)^p \in x^p$ $\forall$ x ≥ 1