CMPS 201 4-26-10

Recall:

BinSearch(A, p, r, t) Pre: A[p..r] sorted

1.) if $p > r$
2.) return 0
3.) $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
4.) if $t = A[q]$
5.) return q
6.) else if $t > A[q]$
7.) return BinSearch(A, q+1, r, t)
8.) else
9.) return BinSearch(A, p, q-1, t)

Runtime?

let $T(n)$ = worst case cost of BinSearch() on Arrays of length n, i.e.

BinSearch$(A, 1, n, t)$

$$T(n) = \begin{cases} c & n = 0 \\ T(\lfloor \frac{n}{2} \rfloor) + d & n \geq 1 \end{cases}$$

To use Master Thm write this as

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

Compare: 1 to $n^{\log_2 1} = n^0 = 1$

By case 2: $\boxed{T(n) = \Theta(\log n)}$.

Recall Case (2): $T(n) = \Theta(f(n) \cdot \log n)$.

NOTE: EXACT SOLUTION:

$$T(n) = (c+d) + d\lfloor \lg n \rfloor$$

CHECK:

$$RHS = T\left(\lfloor \tfrac{n}{2} \rfloor\right) + d$$

$$= \left((c+d) + d\left\lfloor \lg \lfloor \tfrac{n}{2} \rfloor \right\rfloor \right) + d$$

EXERCISE: show $\left\lfloor \lg \lfloor \tfrac{n}{2} \rfloor \right\rfloor = \left\lfloor \lg \left(\tfrac{n}{2}\right) \right\rfloor$

$$= (c+d) + d\left(\left\lfloor \lg \left(\tfrac{n}{2}\right) \right\rfloor + 1 \right)$$

Exercise: $\lfloor x \rfloor + m = \lfloor x + m \rfloor$ if m is an integer

$= (c+d) + d(\lfloor \lg n - \lg 2 + 1 \rfloor)$

$= (c+d) + d \lfloor \lg n \rfloor$

$= T(n)$

$= LHS$

MergeSort:

Recursively sorts A subarray $A[p \ldots r]$ of $A[1 \ldots n]$.

where $n = length[A]$.

MergeSort(A, p, r)

1.) if $p < r$

2.) $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$

3.) MergeSort(A, p, q)

4.) MergeSort(A, q+1, r)

5.) Merge(A, p, q, r)

Merge(A, p, q, r) (Pre: A[p..q], A[q+1..r] both sorted)

sorted
A[p .... q]

sorted
A[(q+1) ... r]

A[p ..... r]

sorted

note: worst case # of comparisons performed by Merge(A, p, q, r) is length[A[p...r]] − 1

Exercise: write pseudo-code for Merge(.) or see p. 29 of 2nd ed. (sec. 2.3).

Theorem:

After MergeSort(A, p, r) is called, the sub-array A[p...r] is sorted. (we Assume Correctness of Merge(.))

Proof: use (strong) induction on $n = length[A[p \ldots r]] = r - p + 1$

I. If $n = 1$ then $r - p + 1 = 1$, so $r = p$, so $p < r$ is false on line (1), so Algorithm does nothing, as it should since $A[p..r]$ is already sorted in this case.

II. Let $n > 1$. Assume that any call to MergeSort on a sub-array of length less than n, results in that sub array being sorted.

now $m > 1 \Rightarrow r - p + 1 > 1 \Rightarrow p < r$

so test on line(1) is true. Upon setting $q = \lfloor \frac{p+r}{2} \rfloor$ we have.

$p \le q < r$ (why? $0 \le \frac{p+r}{2} < r \Rightarrow p \le \lfloor \frac{p+r}{2} \rfloor < r$)

Thus

$$\text{length}[A[p \ldots q]] = q - p + 1 < r - p + 1 = m$$

and

$$\text{length}[A[q+1 \ldots r]] = r - (q+1) + 1$$

$$= r - q \le r - p < r - p + 1 = m$$

By Induction Hypothesis lines (3) & (4) result in $A[p \ldots q]$ and $A[q+1 \ldots r]$ being sorted. $\therefore$ After line(5) $A[p \ldots r]$ is sorted.

Runtime :

Let $T(n)$ = worst case Cost of the call MergeSort(A, 1, n)

Basic Operation : Comparison of Array elements.

$$T(n) = \begin{cases} c & n = 1, 0 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + d(n-1) & n \geq 2 \end{cases}$$

Here d = Cost of 1 Comparison op.

To apply Master Thm, we write

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Compare: n to $n^{\log_2 2} = n^1$

By Case (2): $T(n) = \Theta(n \log n)$.

Exact solution: (in the case $n = 2^k$)

$$T(n) = dn \lg n + cn - \frac{1}{2}d(\lg n)^2 - \frac{1}{2}d \lg n$$

Exercise: check this

Quicksort

Again sort $A[p \ldots r]$ recursively.

Quicksort(A, p, r)

1.) if $p < r$

2.) $\quad q \leftarrow$ Partition(A, p, r)

3.) $\quad$ Quicksort$(A, p, q-1)$

4.) $\quad$ Quicksort$(A, q+1, r)$

Partition(A, p, r)

re-arranges $A[p \ldots r]$ and returns an index q $(p \le q \le r)$ s.t.

$$\underbrace{A[p \ldots (q-1)]}_{\text{not sorted}} \le A[q] \le \underbrace{A[(q+1) \ldots r]}_{\text{not sorted}}$$

$A[q]$: Pivot