CMPS 201 5-3-10

We've shown for Quicksort:

- worst case $T(n) = \Theta(n^2)$
- avg. case $t(n) = \Theta(n \log n)$

RandPartition(A, p, r)

1.) $i \leftarrow$ Rand(p, r) ← random int i s.t. $p \le i \le r$

2.) $A[i] \leftrightarrow A[r]$

3.) return Partition(A, p, r)

RandQuicksort(A, p, r)

1.) if p < r
2.) q ← RandPartition
3.) RandQuicksort(A, p, q-1)
4.) RandQuicksort(A, q+1, r)

Problem: find extreme elements in an array. i.e. max, min.

Ex. Problem: Given A[1...n] determine both max & min entries & return (min, max).

min (m_1, m_2)

1.) if $m_1 < m_2$ ← Does 1 Comparison.
2.) return m_1
3.) else
4.) return m_2

max (M_1, M_2) ← also 1 Comparison.

mix-max (A, P, r) (Pre: $P \leq r$)

1.) if $P = r$
2.) return (A[P], A[P])
3.) $q \leftarrow \lfloor \frac{P+r}{2} \rfloor$
4.) $(m_1, M_1) \leftarrow$ min-max (A, P, q)
5.) $(m_2, M_2) \leftarrow$ min-max (A, q+1, r)
6.) return (min(m_1, m_2), max(M_1, M_2))

Note: obvious iterative algorithm for this problem do s

$$\#\ \text{comp} = 2(n-1) = 2n-2$$

Let $T(n)$ = # comp. performed by min-max$(A, 1, n)$.
(same in best, worst, avg.)

then

$$T(n) = \begin{cases} 0 & n = 1 \\ T\left(\lfloor \frac{n}{2} \rfloor\right) + T\left(\lceil \frac{n}{2} \rceil\right) + 2 & n \geq 2 \end{cases}$$

Exercise: show that exact soln is $T(n) = 2n - 2$.

Lets do IL:

- note: $T(1) = 0$
- for $n \geq 2$:

$$Rhs = T\left(\lfloor \tfrac{n}{2} \rfloor\right) + T\left(\lceil \tfrac{n}{2} \rceil\right) + 2$$

$$= \left(2\lfloor \tfrac{n}{2} \rfloor - 2\right) + \left(2\lceil \tfrac{n}{2} \rceil - 2\right) + 2$$

$$= 2\left(\lfloor \tfrac{n}{2} \rfloor + \lceil \tfrac{n}{2} \rceil\right) - 2$$

$$= 2n - 2 = T(n) = Lhs.$$

Exercise: Verify $T(n) = \Theta(n)$ by master thm.

EXERCISE:

Design a divide & conquer algorithm that finds (min, max) of an Array $A[1 \ldots n]$ by doing exactly $\lceil \frac{3n}{2} \rceil - 2$ (array) comparisons.

Hint: Section 9.1 (2^{nd} e.d. Give an iterative version of this algorithm.)

Problem: determine the i^{th} order statistic of an array $A[1 \cdots n]$. i.e. the

- i^{th} smallest element in A.
- the element which is $\geq$ exactly i other elements
- the element which is $>$ exactly $i-1$ other elements.
- the element $A[i]$ after A is sorted.

we Assume A consists of distinct elements.

One Approach: Sort $A[1 \ldots n]$, then return $A[i]$. In general this costs $\Omega(n \log n)$.

RandSelect(A, p, r, i) (Pre: $1 \le i \le r-p+1$)

1.) if $p = r$
2.) return $A[p]$
3.) $q \leftarrow$ RandPartition(A, p, r)
4.) $k \leftarrow q - p + 1$ // k = length$[A[p \ldots q]]$
5.) if $k = i$
6.) return $A[q]$
7.) else if $i < k$
8.) return RandSelect$(A, p, q-1, i)$
9.) else // $k < i$
10.) return RandSelect$(A, q+1, r, i-k)$

Does $r-p$ comp.

Recall: RandPartition(A, p, r)
returns q in range $p \le q \le r$
And re-arranges $A[p \ldots r]$
so that

$$A[p \ldots q-1] \le A[q] < A[q+1 \ldots r]$$

length = k

Let $t(n)$ = average # of comparisons performed by RandSelect(A, 1, n, i)

Note our use of RandPartition( )

Guarantees that q is equally likely to be any of integers $1 \ldots n$, i.e. Prob $= \frac{1}{n}$. So

$$t(n) = \frac{\sum_{q=1}^{n} \left((n-1) + t(q-1) \cdot P(i<q) + t(n-q) \cdot P(i>q) \right)}{n}$$

note: $P(i<q) = P(i<q\leq n) = \frac{n-i}{n}$

and $P(i>q) = P(1\leq q<i) = \frac{i-1}{n}$

≥ 0

$$t(n) = (n-1) + \frac{1}{n} \cdot \sum_{q=1}^{n} \left(\left(\frac{n-i}{n}\right) t(q-1) + \left(\frac{i-1}{n}\right) t(n-q) \right)$$

$$= (n-1) + \frac{1}{n^2} \left[(n-i) \sum_{q=1}^{n-1} t(q) + (i-1) \cdot \sum_{q=1}^{n-1} t(q) \right]$$

$$= (n-1) + \left(\frac{n - i + i - 1}{n^2} \right) \cdot \sum_{q=1}^{n-1} t(q)$$

$$\therefore \boxed{t(n) = (n-1) + \left(\frac{n-1}{n^2}\right) \sum_{q=1}^{n-1} t(q)}$$

note: $t(n) \geq n-1 = \Omega(n)$

Hw 4: $t(n) \leq 2n = O(n)$.

$\therefore$ $t(n) = \Theta(n)$.

Midterm review:

- Let $h_1(n) = O(g_1(n))$, $h_2(n) = O(g_2(n))$.
  Prove that $h_1(n) \cdot h_2(n) = O(g_1(n) \cdot g_2(n))$.
- Same for Ω, Θ.
- hw Problems on Stirling's

$$\binom{2n}{n} = \Theta(?)$$

- Write a divide & conquer algorithm that returns true/false according to whether $A[1 \ldots n]$ is sorted.
- Proof by induction of correctness of a recursive algorithm.