CSE 201 2-4-21

Quicksort (sec. 7.1 - 7.4)

Again $A[p \ldots r]$ is sorted recursively.

Quicksort(A, p, r)

1. if $p < r$
2. $\quad q = \text{Partition}(A, p, r)$
3. $\quad \text{Quicksort}(A, p, q-1)$
4. $\quad \text{Quicksort}(A, q+1, r)$

Partition arranges $A[p \ldots r]$ so that

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

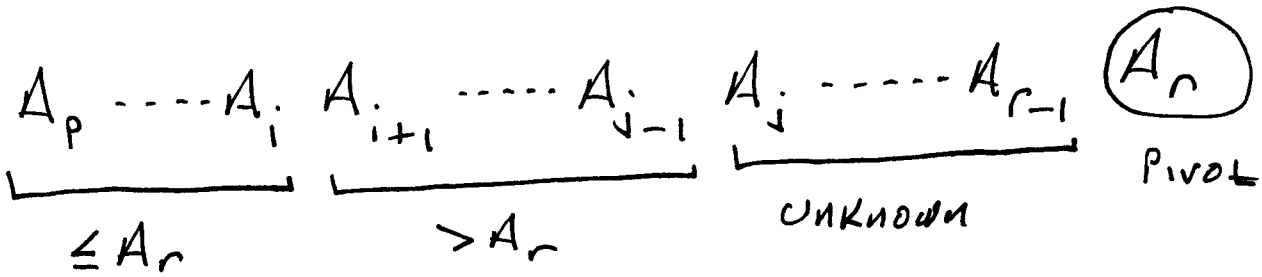
where: $p \leq q < r$

Partition (A, p, r)

1. $i = p - 1$
2. for $j = p$ to $(r-1)$
3. if $A[j] \leq A[r]$
4. $i = i + 1$
5. $A[i] \leftrightarrow A[j]$ // swap
6. $A[i+1] \leftrightarrow A[r]$ // swap
7. return $i+1$

note

- if $A_j \leq A_r$: swap $A_j \leftrightarrow A_{i+1}$, i++, j++
- if $A_j > A_r$: j++



loop 2-5 maintains following invariants

(1) $i < j < r$

(2) $A[p \ldots i] \leq A[r]$

(3) $A[r] < A[(i+1) \ldots (j-1)]$

(4) $A[j \ldots (r-1)]$ has not been processed.

It follows that

$$A[p \ldots (q-1)] \leq A[q] < A[(q+1) \ldots r]$$

is true when Partition returns.

note: if $n = \text{length}(A[p \ldots r]) = r-p+1$, then Partition$(A, p, r)$ does exactly $n-1$ comparisons. (best, worst, avg. cases.)

Top level: Partition$(A, 1, n)$ does $n-1$ comparisons.

worst case occurs when $A[p \ldots r]$ is already sorted:

$$A[p \text{----} (q-1)] \leq A[q] < \ldots \text{empty} \ldots$$

(q = r)

Let $T(n)$ = worst case # of comparisons by Quicksort$(A, 1, n)$, i.e. when $A[1 \ldots n]$ is sorted

$$T(n) = \begin{cases} 0 & n = 0, 1 \\ T(n-1) + (n-1) & n \geq 2 \end{cases}$$

Iteration method:

$$\begin{aligned} T(n) &= (n-1) + T(n-1) \\ &= (n-1) + (n-2) + T(n-2) \\ &= (n-1) + (n-2) + (n-3) + T(n-3) \\ &\vdots \\ &= (n-1) + (n-2) + (n-3) + \cdots + 3 + 2 + 1 + 0 \end{aligned}$$

$\therefore T(n) = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n = \Theta(n^2)$

What's quick about Quicksort?

Average case!!

Assume all $n!$ permutations of input array $A[1 \ldots n]$ are equally likely.

Let $t(n)$ = average case # of comparisons of Quicksort$(A, 1, n)$. Then

$$t(n) = \frac{\sum_{\text{all Permutations}} (\#\text{ of comp. Perf. on given Perm.})}{n!}$$

Goal: find a recurrence for $t(n)$.

Our assumption implies that the return value q of Partition is equally likely to be any of the values $q = 1, 2, 3, \ldots, n$.

$\therefore$ $\text{Prob}(q = i) = \frac{1}{n}$ for $i = 1, \ldots, n$.

Also

- Partition$(A, 1, n)$ does $n-1$ comparisons
- length$(A[1 \ldots q-1]) = q - 1 - 1 + 1 = q - 1$
- length$(A[q+1 \ldots n]) = n - (q+1) + 1 = n - q$

Thus

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

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

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

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

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

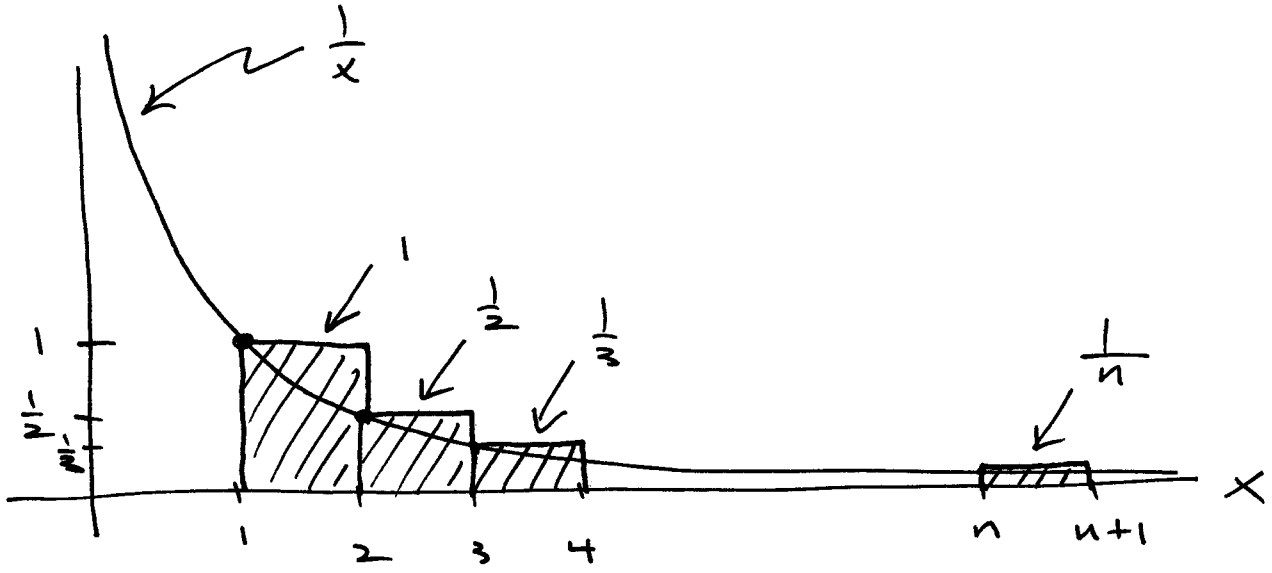
Solution :

$$\boxed{t(n) = 2(n+1)H_n - 4n}$$

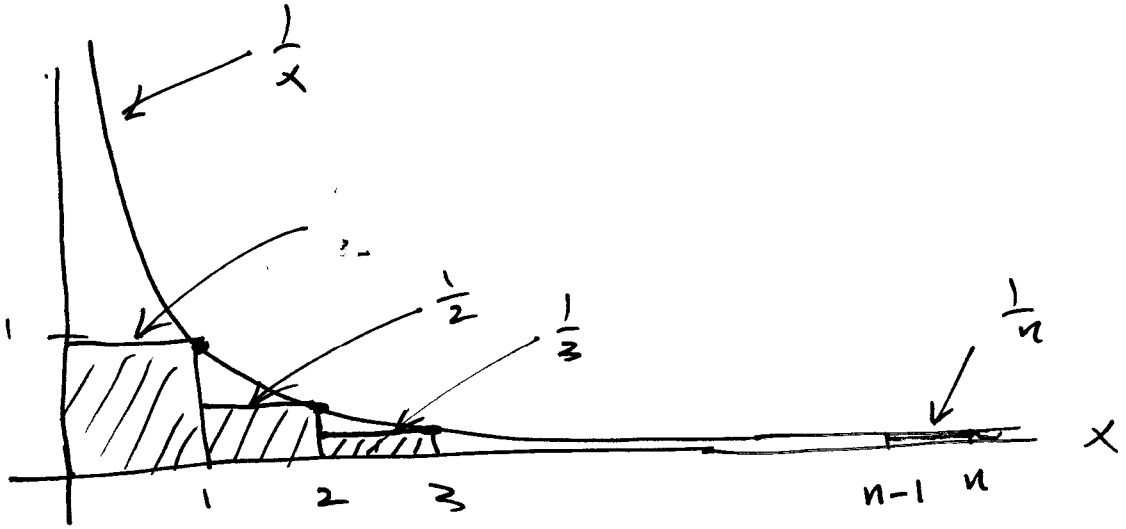
where $H_n = \sum_{r=1}^{n} \frac{1}{r} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

is the n^{th} harmonic number.

estimate size of H_n



$$\int_1^{n+1} \frac{1}{x}\,dx \le H_n \le 1 + \int_1^{n} \frac{1}{x}\,dx$$



Thus

$$\Omega(\ln n) = \ln(n+1) \leq H_n \leq 1 + \ln(n) = O(\ln n)$$

$\therefore \; H_n = \Theta(\ln(n))$

in fact:

$H_n \sim \ln(n)$

$\therefore \; t(n) = 2(n+1)\Theta(\log n) - 4n$

$\therefore \; \boxed{t(n) = \Theta(n \log n)}$ ← asymptotic solution

$t(n) \sim 2n\ln(n)$

In practice we randomize Quicksort to elicit average case runtime.

RandPartition(A, p, r)

```
1. i = Random(p, r)
2. A[i] ↔ A[r]  // swap
3. return Partition(A, p, r)
```

RandQuicksort(A, p, r)

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

Sec. 4.2 (P. 75-83)

Strassen's algorithm

Problem: $C = A \cdot B$

$n \times n$ $\quad n \times n$ $\quad n \times n$

let $A = (a_{ij})$, $B = (b_{ij})$, $C = (c_{ij})$.

row col $\quad$ row col

Recall

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$ { n basic ops.

Basic op: multiplication of real numbers.

cost = $\Theta(n^3)$

If n is an exact power of 2

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$\frac{n}{2} \times \frac{n}{2}$ $\frac{n}{2} \times \frac{n}{2}$

note.

$$C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \quad \frac{n}{2} \times \frac{n}{2}$$

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

8 matrix multiplications on $\frac{n}{2} \times \frac{n}{2}$ sub-matrices.

Runtime

$$T(n) = \begin{cases} 1 & n = 1 \\ 8T\left(\frac{n}{2}\right) + 1 & n \geq 2 \end{cases}$$

↑ cost of div. & comb.

By master thm

$$T(n) = \Theta(n^3)$$