REMARKS.

LOOP 2-5 MAINTAINS THE FOLLOWING INVARIANTS

(i) $i \le j < r$

(ii) $A[p..i] \le A[r]$

(iii) $A[r] \le A[(i+1)..(j-1)]$

(iv) THE ELEMENTS IN $A[i..(r-1)]$ HAVE NOT BEEN PROCESSED AND MAY BE $\ge A[r]$ OR $\le A[r]$.

EXERCISE

- VERIFY THESE CLAIMS ON EXAMPLES
- PROVE THEM.

IT FOLLOWS THAT WHEN Partition returns:

$$A[p..(q-1)] \le A[q] \le A[(q+1)...n]$$

THE RUN TIME OF Partition (IN TERMS, WORST, AND AVERAGE CASES) IS $\Theta(m)$ WHERE $m = r - p + 1$, SINCE LOOP 2-5 STEPS THROUGH THE ENTIRE SUBARRAY $A[p..(r-1)]$, THEN THE PIVOT IS SET IN PLACE.

EXERCISE.
PROVE THE CORRECTNESS OF QUICKSORT BY INDUCTION ON THE LENGTH OF THE SUB-ARRAY $A[p..r]$: $n = r-p+1$.

THE RUNTIME OF QUICKSORT DEPENDS HEAVILY ON THE VALUE q RETURNED BY PARTITION. IF THE SUBARRAYS $A[p..(q-1)]$ AND $A[(q+1)..r]$ ARE NOT BALANCED (i.e. OF ROUGHLY EQUAL SIZE) THEN PERFORMANCE IS INHIBITED. IN THIS CASE ONE RECURSIVE CALL TO QUICKSORT IS ON A SUBARRAY WHICH IS INORDINATELY LONG.

THE WORST CASE OCCURS WHEN THE ARRAY IS ALREADY SORTED. THEN PARTITION RETURNS

$$\underbrace{A[p\ldots(r-1)]}_{\text{SORTED}} \leq A[r] \leq \ldots \text{EMPTY} \ldots$$

$\uparrow$ q

LET $T(n)$ DENOTE THE WORST CASE RUN TIME OF QUICKSORT (i.e. WITH $A[1..n]$ ALREADY SORTED.)

$T(n)$ = WORST CASE # of COMPARISONS By QUICKSORT ON $A[1 \ldots n]$.

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

$$\begin{aligned} T(n) &= (n-1) + T(n-1) \\ &= (n-1) + (n-2) + T(n-2) \\ &\vdots \\ &= \sum_{i=1}^{K} (n-i) + T(n-K) \end{aligned}$$

CHOOSE K S.T. $n-K = 1 \quad \therefore K = n-1$

$$\begin{aligned} T(n) &= \sum_{i=1}^{n-1} (n-i) + T(1) \\ &= n(n-1) - \frac{n(n-1)}{2} + 0 \\ &= \frac{1}{2} n(n-1) = \frac{1}{2} n^2 - \frac{1}{2} n = \Theta(n^2). \end{aligned}$$

THEN $T(n)$ SATISFIES

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

SIMPLIFY THIS TO $T(n) = T(n-1) + cn$ FOR DEFINITENESS. BY THE ITERATION METHOD

$$\begin{aligned} T(n) &= cn + T(n-1) \\ &= cn + c(n-1) + T(n-2) \\ &= cn + c(n-1) + c(n-2) + T(n-3) \\ &= c \sum_{i=0}^{k-1} (n-i) + T(n-k) \\ &= cnk - \tfrac{1}{2} ck(k-1) + T(n-k). \end{aligned}$$

CHOOSE k SUCH THAT $n-k=1$, i.e. $k = n-1$. THEN

$$\begin{aligned} T(n) &= cn(n-1) - \tfrac{1}{2} c(n-1)(n-2) + \text{const.} \\ &= \Theta(n^2). \end{aligned}$$

SO WHAT'S SO QUICK ABOUT QUICKSORT?

THE ADVANTAGE OF QUICKSORT IS IN ITS AVERAGE CASE.

WE ASSUME THAT ALL $n!$ PERMUTATIONS OF THE INPUT ARRAY $A[1 \ldots n]$ ARE EQUALLY LIKELY, i.e. THAT ANY GIVEN PERMUTATION OCCURS WITH PROBABILITY $\frac{1}{n!}$.

WE CHOOSE AS BASIC OPERATION (i.e. BAROMETER) THE COMPARISON OF NUMERICAL VALUES ON line 3 OF Partition. LET $t(n)$ DENOTE THE AVERAGE NUMBER OF COMPARISONS PERFORMED BY QUICKSORT ON AN INPUT ARRAY $A[1 \ldots n]$ OF LENGTH n, i.e.

$$t(n) = \frac{\sum_{\text{ALL PERMUTATIONS}} (\#\text{ OF COMPARISONS PERFORMED ON GIVEN PERMUTATION})}{n!}$$

WE WISH TO DETERMINE A RECURRENCE FOR $t(n)$. OUR ASSUMPTION IMPLIES THAT THE PIVOT $A[q]$ IS EQUALLY LIKELY TO BE PLACED IN ANY OF THE n LOCATIONS IN $A[1 \ldots n]$.

THUS THE RETURN VALUE q OF PARTITION HAS PROBABILITY $\frac{1}{n}$ OF BEING ANY ONE OF THE n VALUES: $q = 1, 2, \ldots, n$.

OBSERVE THAT PARTITION ITSELF DOES $(n-1)$ COMPARISONS ON $A[1 \cdots n]$. ALSO

$$\text{length}\left[A[1 \cdots (q-1)]\right] = q-1$$

AND

$$\text{length}\left[A[(q+1) \cdots n]\right] = n-q.$$

THUS

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

SO

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

THE INITIAL VALUES $t(0)=0$, $t(1)=0$, $t(2)=1$ CAN BE SEEN BY INSPECTION. THUS

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

WHENCE

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

TO SOLVE THIS RECURRENCE WE RESORT TO SOME TRICKS. LET

$$x_n = \sum_{q=1}^{n-1} t(q) \quad , \quad x_1 = 0 .$$

THEN

$$x_{n+1} - x_n = \sum_{q=1}^{n} t(q) - \sum_{q=1}^{n-1} t(q) = t(n) ,$$

AND SO

$$x_{n+1} - x_n = (n-1) + \frac{2}{n} \cdot x_n$$

$$\therefore \quad x_{n+1} - \left(\frac{n+2}{n}\right) x_n = n-1$$

MULTIPLY BY THE MAGIC NUMBER $\frac{1}{(n+1)(n+2)}$

$$\therefore \quad \frac{x_{n+1}}{(n+1)(n+2)} - \frac{x_n}{n(n+1)} = \frac{n-1}{(n+1)(n+2)} = \frac{3}{n+2} - \frac{2}{n+1}$$

REPLACE n BY r:

$$\frac{x_{r+1}}{(r+1)(r+2)} - \frac{x_r}{r(r+1)} = \frac{3}{r+2} - \frac{2}{r+1}$$

SUM FOR $r = 1$ TO $n-1$:

$$\sum_{r=1}^{n-1}\left(\frac{x_{r+1}}{(r+1)(r+2)} - \frac{x_r}{r(r+1)}\right) = \sum_{r=1}^{n-1}\left(\frac{1}{r+2} + \frac{2}{r+2} - \frac{2}{r+1}\right)$$

$$\therefore \quad \sum_{r=2}^{n}\frac{x_r}{r(r+1)} - \sum_{r=1}^{n-1}\frac{x_r}{r(r+1)} = \sum_{r=3}^{n+1}\frac{1}{r} + \sum_{r=3}^{n+1}\frac{2}{r} - \sum_{r=2}^{n}\frac{2}{r}$$

$$\therefore \quad \frac{x_n}{n(n+1)} - \frac{x_1}{2} = \sum_{r=3}^{n+1}\frac{1}{r} + \frac{2}{n+1} - 1$$

$$= \sum_{r=1}^{n}\frac{1}{r} + \frac{1}{n+1} - 1 - \frac{1}{2} + \frac{2}{n+1} - 1$$

$$= \sum_{r=1}^{n}\frac{1}{r} + \frac{3}{n+1} - \frac{5}{2}$$

RECALL $x_1 = 0$. DEFINE $H_n = \sum_{r=1}^{n}\frac{1}{r}$ (CALLED THE n^{TH} HARMONIC NUMBER.) THEN

$$\frac{x_n}{n(n+1)} = \frac{3}{n+1} - \frac{5}{2} + H_n$$

$$\therefore \quad x_n = 3n - \frac{5}{2}n(n+1) + n(n+1)H_n$$

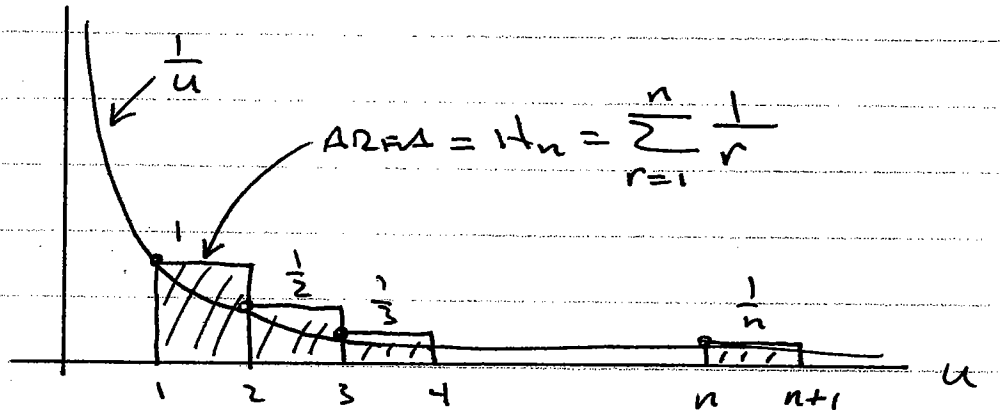
$$\therefore \quad t(n) = (n-1) + \frac{2}{n}x_n$$

$$= n - 1 + 6 - 5(n+1) + 2(n+1)\cdot H_n$$

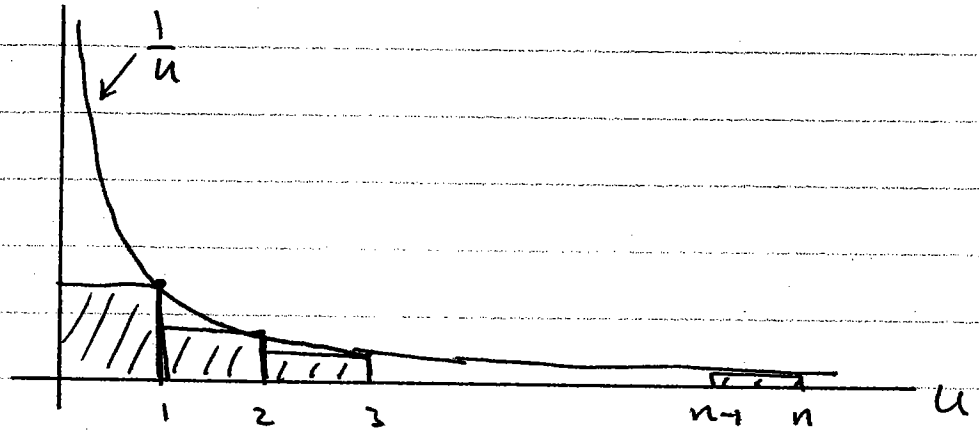
AND SO

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

WE MUST ESTIMATE THE SIZE OF H_n TO DETERMINE THE ASYMPTOTIC ORDER OF $t(n)$



$$\int_1^{n+1} \frac{1}{u}\,du \le H_n \le 1 + \int_1^{n} \frac{1}{u}\,du$$



THUS

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

$$\therefore \quad H_n = \Theta(\ln(n)) = \Theta(\lg(n))$$

THUS

$$t(n) = \Theta(n \lg(n)),$$

WHICH IS BETTER THAN THE WORST CASE $\Theta(n^2)$ RUN TIME.

OUR ASSUMPTION THAT ALL PERMUTATIONS OF $A[1 \cdots n]$ ARE EQUALLY LIKELY MAY NOT BE WELL FOUNDED IN PRACTICE. WE MAY WISH TO SORT A PRE-SORTED ARRAY, OR ONE THAT IS NEARLY SORTED MORE OFTEN THAN NOT.

THERE ARE SEVERAL WAYS TO RANDOMIZE QUICKSORT TO COMPENSATE FOR THIS.

ONE WAY IS TO SIMPLY APPLY A RANDOMLY CHOSEN PERMUTATION TO $A[1 \cdots n]$ BEFORE CALLING QUICKSORT.

A SIMPLER WAY IS TO CHOOSE A RANDOM ELEMENT IN A[p...r] AS PIVOT, SWAP THAT WITH A[r], THEN PARTITION AS USUAL.

RandPartition(A, p, r)

1.) i ← Rand(p, r)
2.) A[i] ↔ A[r]
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)

THIS IS CONSIDERED BY SOME TO BE THE ALGORITHM OF CHOICE FOR SORTING LARGE INPUTS.

Ex

MaxMin(A,p,r) FINDS THE MAXIMUM AND MINIMUM ELEMENTS IN THE SUBARRAY A[p..r].

min(m_1, m_2)

1.) if $m_1 < m_2$
2.) return m_1
3.) return m_2

max(M_1, M_2) is similar. EACH DOES ONE COMPARISON.

MaxMin(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.) $(m_1, M_1) \leftarrow$ MaxMin(A, q+1, r)
5.) $(m_2, M_2) \leftarrow$ MaxMin(A, p, q)
6.) return (min(m_1,m_2), max(M_1,M_2))

LET T(n) DENOTE THE NUMBER OF COMPARISONS PERFORMED BY MaxMin(A,p,r) ON ARRAYS OF LENGTH n.

THEN

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

EXERCISE:

SHOW THAT THE EXACT SOLUTION IS $T(n) = 2n - 2$.

THIS IS NO BETTER THAN THE OBVIOUS ITERATIVE ALGORITHM.

EXERCISE:

DESIGN A DIVIDE AND CONQUER ALGORITHM WHICH FINDS MAXIMUM AND MINIMUM IN EXACTLY $\lceil \frac{2n}{3} \rceil - 2$ COMPARISONS. (HINT: SECTION 9.1 DESCRIBES AN ITERATIVE ALGORITHM TO DO THIS.)

Ex. The Selection Problem

The i^{th} order statistic of an array $A[1 \cdots n]$ consisting of n distinct elements is the i^{th} smallest element. Equivalently the i^{th} order statistic is the unique element in A which is greater than exactly $i-1$ other elements (where $1 \le i \le n$).

e.g. $i = 1$ Gives the minimum
$i = n$ Gives the maximum

The i^{th} order statistic is greater than or equal to exactly i elements of A.

Problem: Given $A[1 \cdots n]$, where all elements are distinct, determine the i^{th} order statistic.

One approach would be to sort $A[1 \cdots n]$ then just return $A[i]$. In general this takes time $\Omega(n \lg n)$.

RandSelect is a randomized algorithm which finds the i^{th} order statistic in linear time, on average.

RECALL THAT RandPartition(A,p,r) SPLITS THE SUBARRAY A[p..r] INTO TWO SUBARRAYS SATISFYING

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

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

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

RandSelect is similar in some respects to both Quicksort and BinarySearch. Like Quicksort it randomly splits the subarray A[p..r] in order to exploit a good average case running time. Like BinarySearch it recurs on only one subarray. Unlike BinarySearch we seek not an index, but an array element.