Merge Sort

Merge

$[x_1, x_2, \ldots x_m]$ ← Both sorted

$[y_1, y_2, \ldots, y_n]$

$O(m+n)$

Output: $m+n$

if $x_1 \leq y_1$

append x_1 to output

merge $([x_2, \ldots, x_m], [y_1 \ldots, y_n])$

else

append y_1 to output

merge $([x_1, \ldots x_m], [y_2, \ldots y_n])$

```
mergesort(l)
    l1 = mergesort(lefthalf(l))
    l2 = mergesort(righthalf(l))
    return(merge(l1, l2))
```

DIVIDE & CONQUER

(Base cases: mergesort([]) = [], mergesort([x]) = [x])

$T(0) = 1$

$T(1) = 1$

$T(n) = 2T(n/2) + \overset{\times}{c}n$

$$T(0) = T(1) = 1$$

$$T(n) = 2T(n/2) + n$$

$$T(n) = 2\left[2T(n/4) + n/2\right] + n$$

$$= 2^2 T(n/2^2) + 2n$$

$$= 2^2\left[2T(n/2^3) + n/2^2\right] + 2n$$

$$= 2^3 T(n/2^3) + 3n$$

$$\vdots$$

$$= 2^k T(n/2^k) + kn$$

After k unwindings

$$T(n) = 2^k T(n/2^k) + kn$$

Assume n is a power of 2 $\Rightarrow n = 2^l$

At $\log_2 n$

$$T(n) = 2^{\log_2 n} T(1) + (\log_2 n) \cdot n$$

$$= n + n \log_2 n$$

$$= O(n \log n)$$

Quicksort

- Why merge? No guarantee that all elements on left are smaller than right
- Divide by value
  - lower [below median]
  - upper [above median]

> Ideally

Getting the median is not easy

Instead, split w.r.t. some value in list

$[x_1, \ldots, x_n]$

Say x_1 is splitter

lower = $[x_j \mid x_j \leq x_1, \; j > 1]$

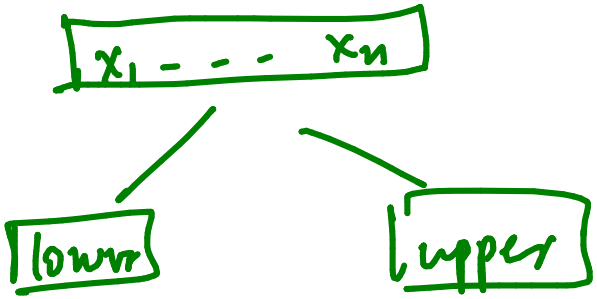
upper = $[x_j \mid x_j > x_1, \; j > 1]$

(sort lower) ++ [splitter] ++ (sort upper)

Quicksort: CAR Hoare early 1960's

Base Case: $T(0) = T(1) = 1$

Splitter is not the median, in general



splitter is min value

" " max "

$T(n) = T(n-1) + n$ $\sum_i i = O(n^2)$

Sorting:

Space of all inputs are permutations of 1..n

Uniform distribution on n! permutations &

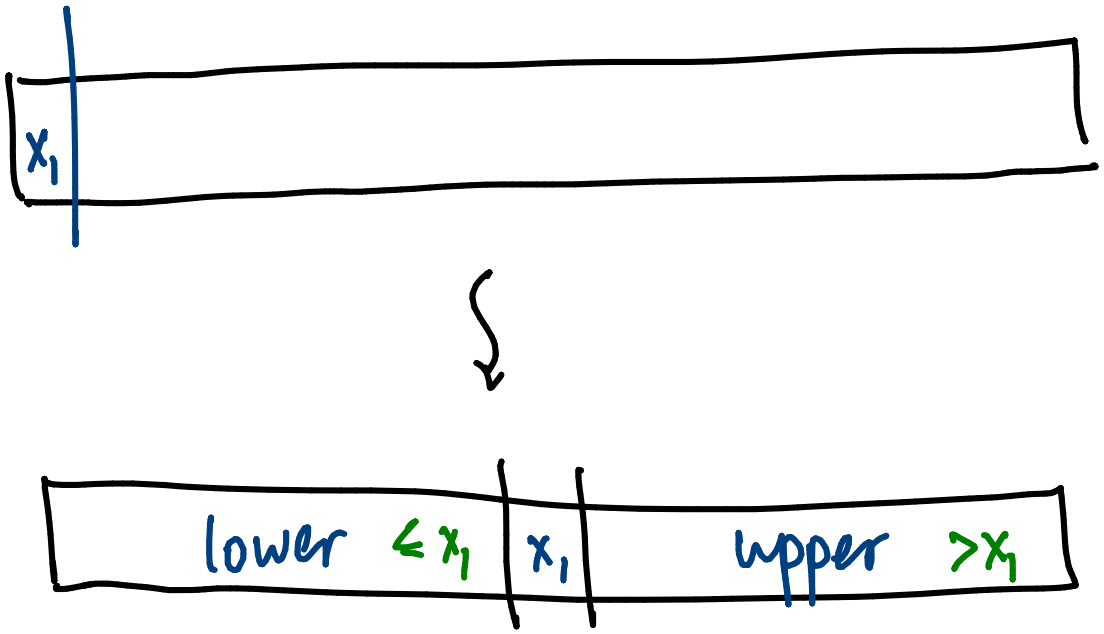
calculate expected running time

Quicksort has expected running time $O(n \log n)$

Use this fact?

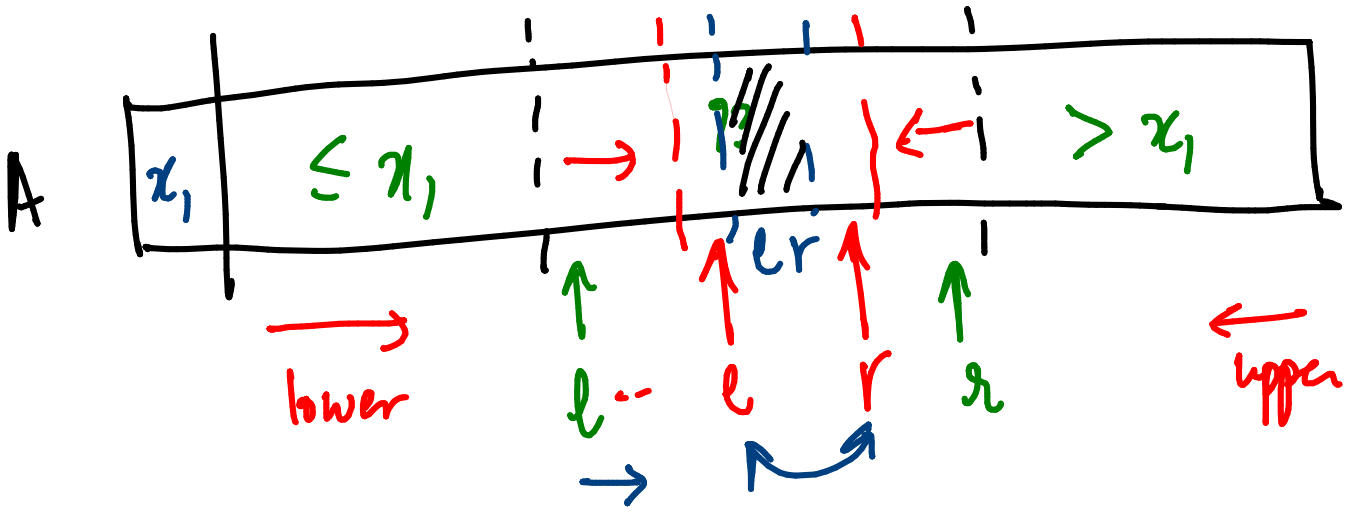
Randomize choice of pivot

Quicksort in place Partitioning



Two strategies

Hoare's partitioning strategy (arrays)

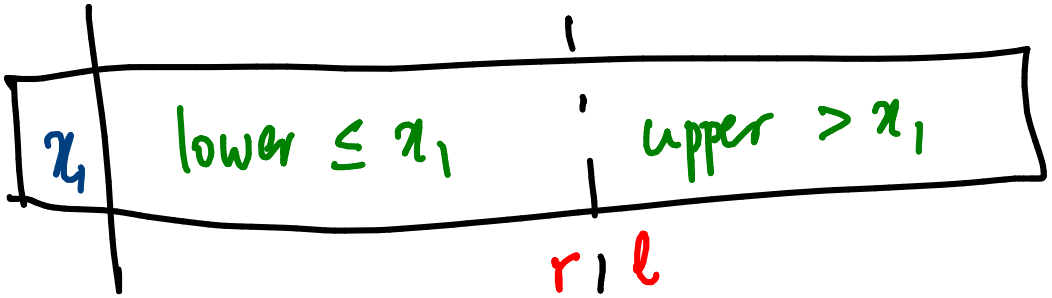


Initially $l = 2, r = n$

Increment l till $A[l] > x_1$

Decrement r till $A[r] \leq x_1$

Exchange, increment l, decrement r



"Invariant"

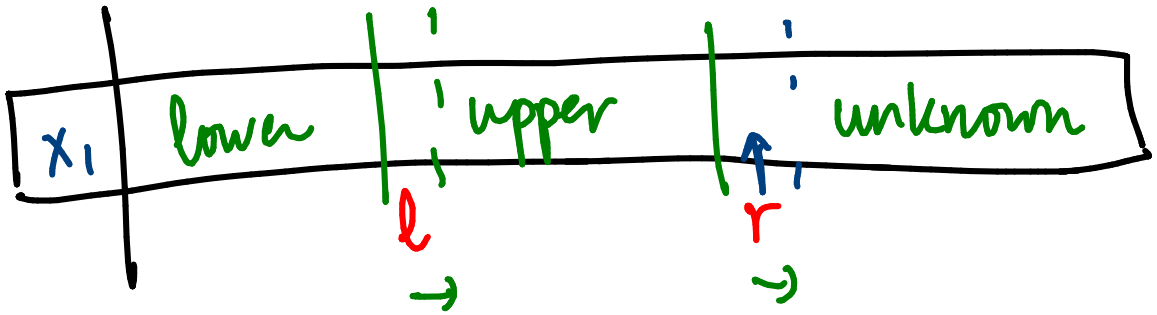
At all points A[2] .. A[l-1] are in lower

A[r+1] .. A[n] are in upper

A[l] .. A[r] is "unknown"

Swap x_1, A[r]

Another partitioning strategy



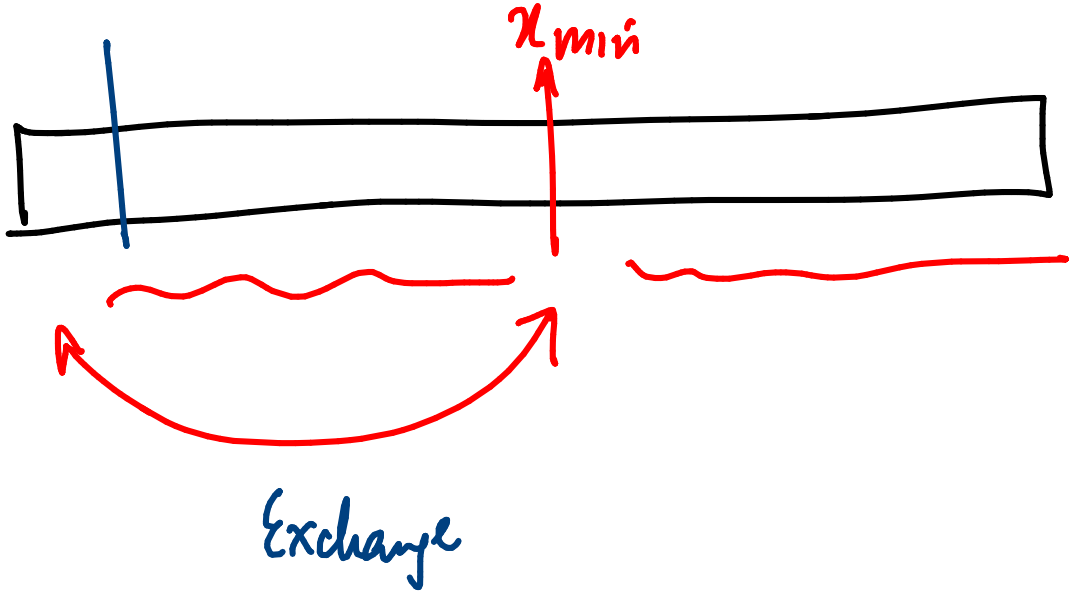
Initially $l = r = 2$

while $(r <= n)$

if $A[r] > x_1$, $r = r + 1$

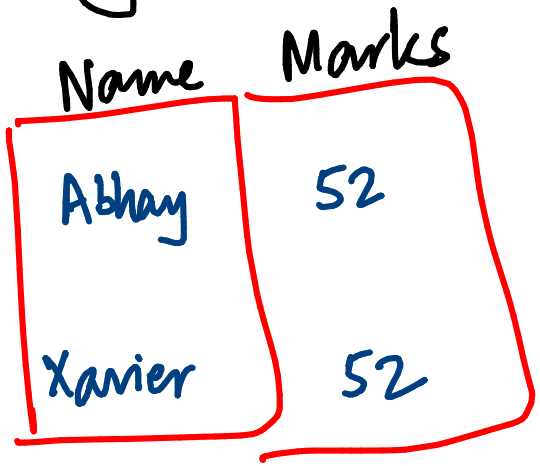
if $A[r] \leq x_1$, swap $(A[l], A[r])$, $l = l + 1$, $r = r + 1$

Selection sort in place



Stable sorting

Spreadsheet



Originally, rows are sorted by name

Now sort by marks

Want

Abhay 52
Xavier 52

long distance swaps disturb stability

Quicksort
Selection sort
] not stable as described

Mergesort, Insertionsort stable if we handle $\leq$ case sensibly