6.006 Recitation 9 4 March 2009

Note Title 3/4/2009

QUICKSORT

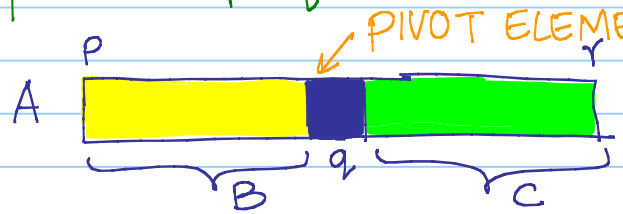
C.A.R. Hoare. Quicksort. Computer Journal, 5(1):10-15, 1962.

DIVIDE-AND-CONQUER

Divide: Partition $A[p..r]$ into two subarrays $B = A[p..q-1]$ and $C = A[q+1..r]$ s.t.

$\forall i,\ p \le i \le q-1 \quad B[i] \le A[q]$

$\forall j,\ q+1 \le j \le r \quad A[q] > C[j]$

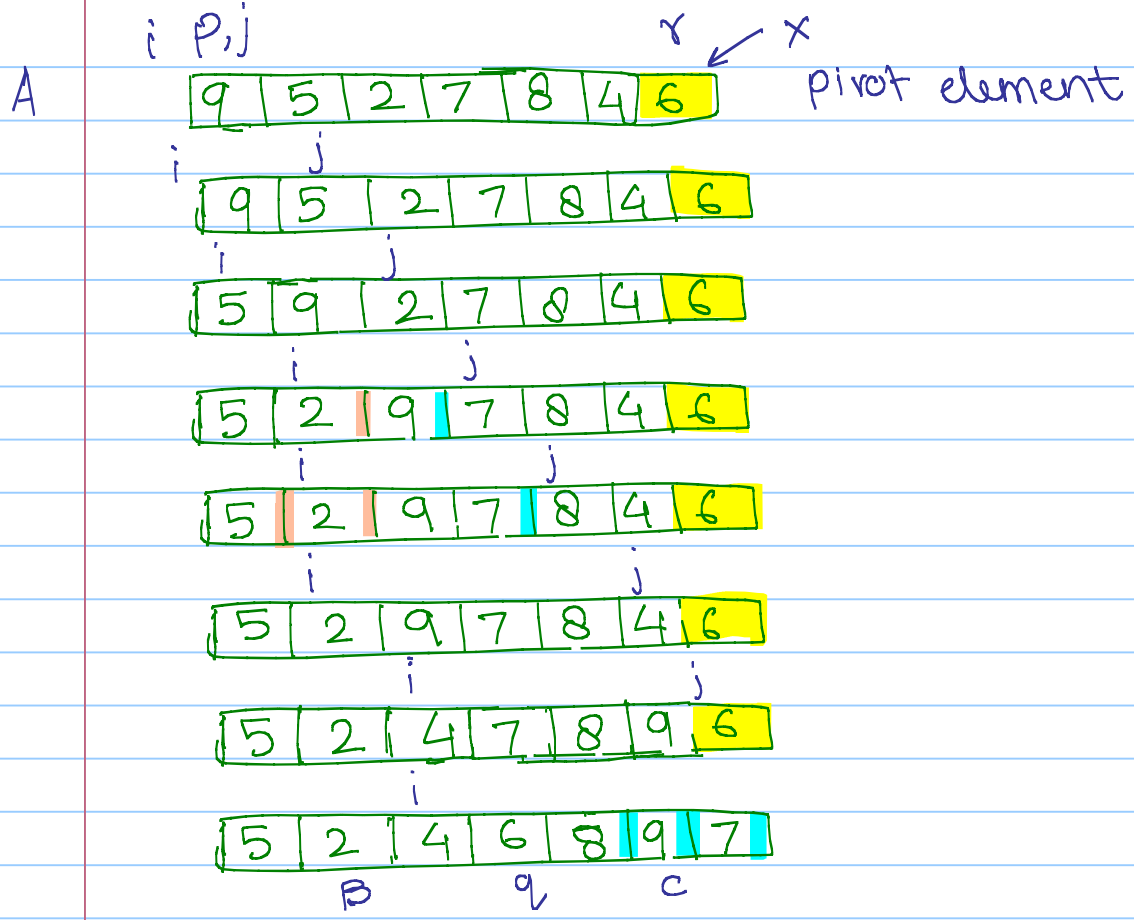


Conquer: Sort B and C recursively using quicksort

Combine: Not needed! A is sorted.

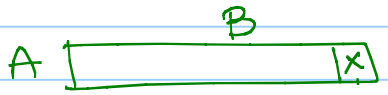
```
QUICKSORT (A,p,r)      // A: array , p : StartIndex , r : EndIndex
if p<r
   q = PARTITION (A,p,r)    // q : index of A[r] pivot in sorted array A
   QUICKSORT (A,p,q-1)      // call Quicksort on B
   QUICKSORT (A,q+1,r)      //    on C


PARTITION (A,p,r)
   x = A[r]              // pivot element
   i = p-1
   for j = p to r-1
       if A[j] ≤ x
          i = i+1
          swap A[j] ↔ A[i]
   swap A[i+1] ↔ A[r]
   return i+1
```


i p,j
r
x
A
9 5 2 7 8 4 6
pivot element
i j
9 5 2 7 8 4 6
i j
5 9 2 7 8 4 6
i j
5 2 9 7 8 4 6
i j
5 2 9 7 8 4 6
i j
5 2 9 7 8 4 6
i j
5 2 4 7 8 9 6
i
5 2 4 6 8 9 7
p q r

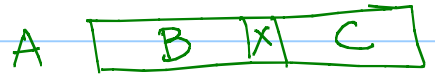
Performance

* Worst - case Partitioning



p is minimum or maximum element

$$T(n) = T(n-1) + T(0) + \Theta(n)$$
$$= n + n-1 + n-2 + \cdots + 1$$
$$= \frac{n(n+1)}{2}$$
$$= \Theta(n^2)$$



* Best - case partitioning

$$T(n) \leq 2T(n/2) + \Theta(n)$$
$$= O(n \log n)$$

Balanced Partitioning

Average case running time much closer to best case than the worst case.

Say, we always get 9-1 split

$$T(n) \leq T\left(\frac{9n}{10}\right) + T\left(\frac{n}{10}\right) + cn$$

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

Even with 999-1 split
$O(n \log n)$

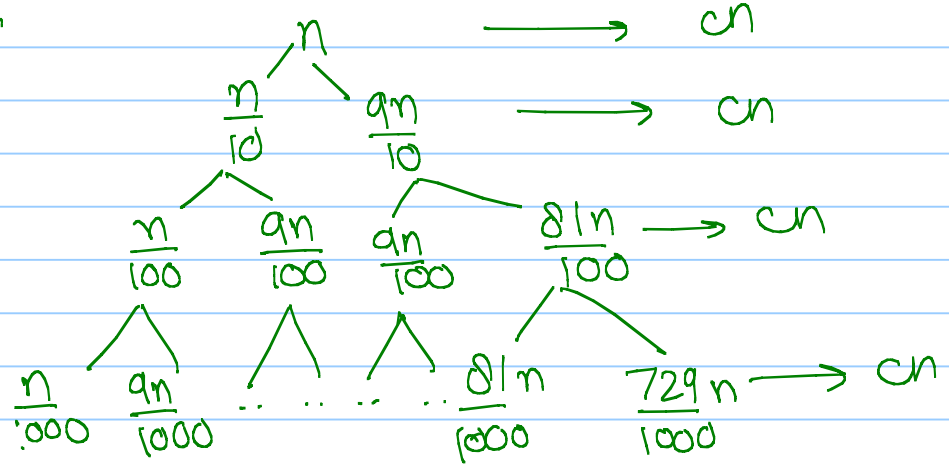
Any split with constant proportionality (independent of n)

$$\left(\frac{9}{10}\right)^d n = 1$$

$$\left(\frac{10}{9}\right)^d = n$$

$$d = \log_{10/9} n$$

$d = \log_{10/9} n$

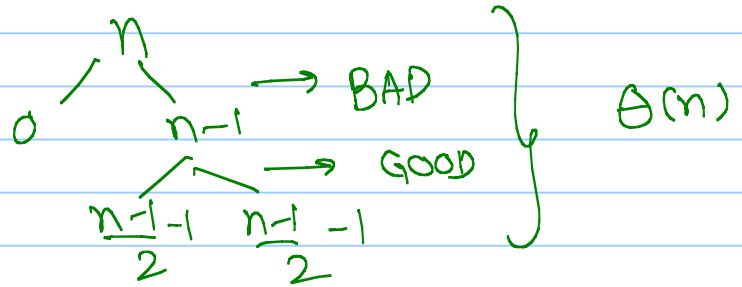


AVERAGE CASE

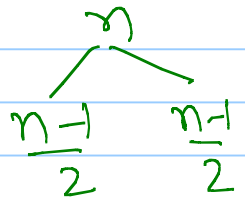
→ various kinds of inputs
→ Ordering of values in A matters, not the values

→ Previous analysis assume partition always occur in same proportion (Not very likely)

→ Mix of GOOD and BAD splits



$\Theta(n)$



$\Theta(n)$

likely complexity $\Theta(n \log n)$

RANDOMIZED QUICKSORT

```
RANDOMIZED-PARTITION(A,p,r)
  i = RANDOM(p,r)
  swap A[i] ↔ A[r]
  return PARTITION(A,p,r)
```

Choose the pivot node at random

(Extremely hard to select bad pivot node randomly all the time)!

Expected running time : $\Theta(n \log n)$

{ not that hard to show }