WEEK 3

L1 | L1 | L1 | L1

QuickSort

- Definitely a "greatest hit" algorithm
- Prevalent in practice
- Beautiful Analysis
- O(n log n) time "on average", works in place (not much extra memory needed)

Back at the Sorting Problem.

Input: | 3 | 8 | 2 | 5 | 1 | 4 | 7 | 6 |

Output: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Assume all array elements are distinct
(extend to handle duplicate as exercise)

Key idea: partition around a pivot element.

- pick first element of array(3)(for now) | 3 | 8 | 2 | 5 | 1 | 4 | 7 | 6 |
- rearrange array s.t. all elements left of the pivot are smaller than the pivot, all elements to the right are greater than the pivot

| 2 | 1 | 3 | 6 | 7 | 4 | 5 | 8 |

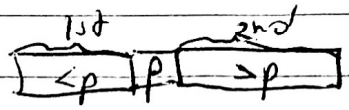
Then Note: pivot is in the right position now
1) This all happens in (O(n)) linear time, and requires no extra memory
2) It reduces the problem size

Quicksort High Level Description

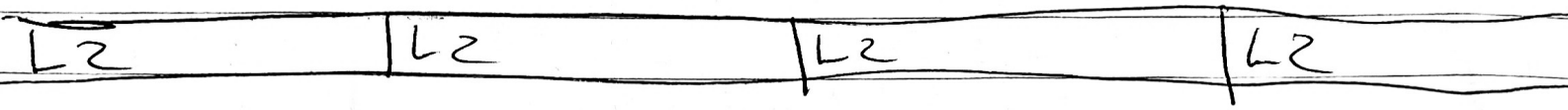
D&C

Quicksort (Array A of length n)

- if n=1 return
- p = ChoosePivot(A, n) (implemented later)
- Partition A around p



- recursively sort 1st & 2nd parts



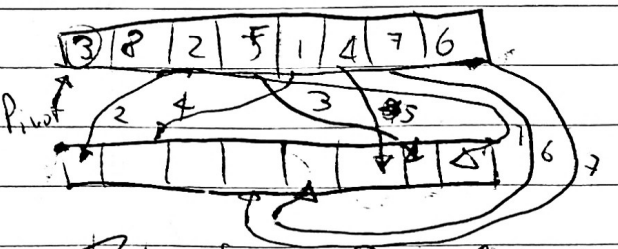
Partitioning around a pivot in detail

Two cool facts about partition:

① linear time, no extra memory

② Reduces problem size

Easy way out (not in place), using O(n) extra memory



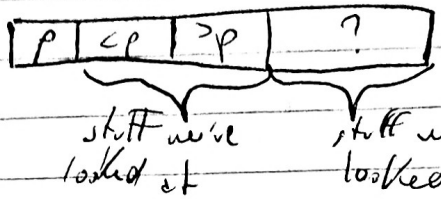
Scan LTR, ~~check~~ if n < Pivot {fill from left}, else fill from right

We can do it without additional memory!

Assume: pivot = 1st element of array

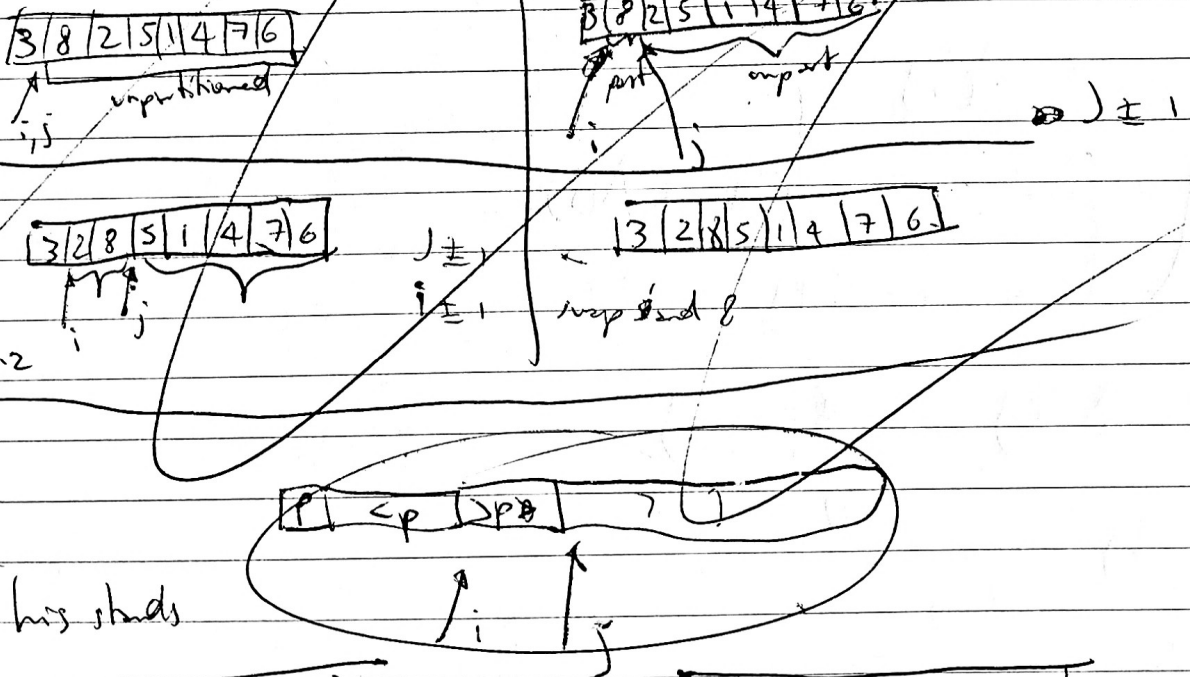
[if not, swap pivot with 1st element of preprocessing step]

High-Level Idea:

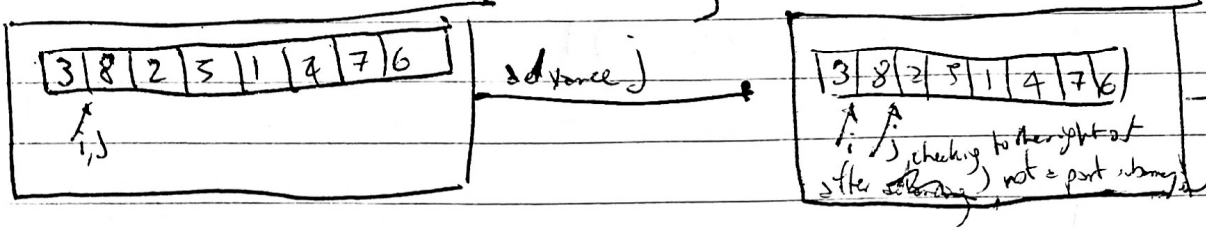


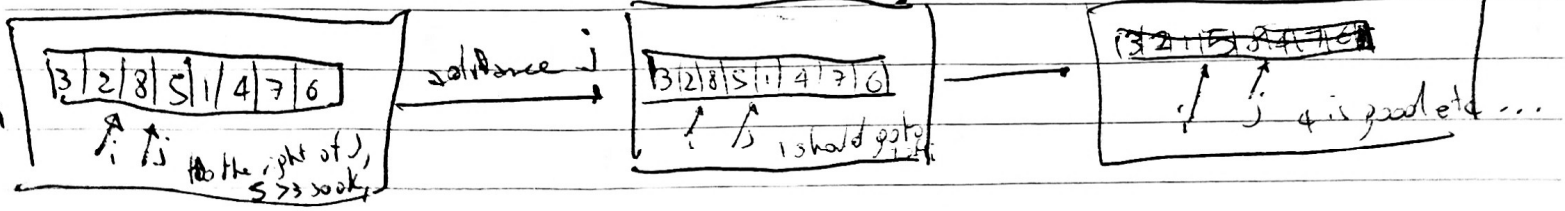
- single scan through array
- invariant: everything looked at so far is partitioned

Partition Example

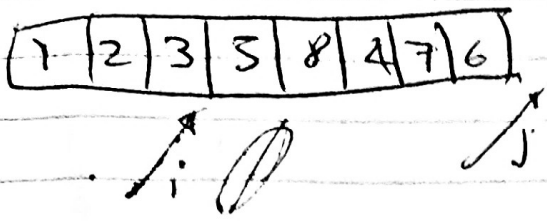


This stinks





Swap with left pivot to the right of i:



Pseudocode for this:

Partition (A, ℓ, r) input = $A[\ell \ldots r]$

- pivot = $A(\ell)$
- $i = \ell + 1$
- for $j = \ell + 1$ to r:
  - if $A[j] < p$:
    - swap $A[j]$, $A[j]$
    - $i = i + 1$
- Swap $A[\ell]$ and $A[i-1]$

$O(n)$ (wow) $n = r - \ell + 1$

$O(1)$ work per array entry (len(array) = n)

Also works in place!

Also correct

Choosing a good Pivot.

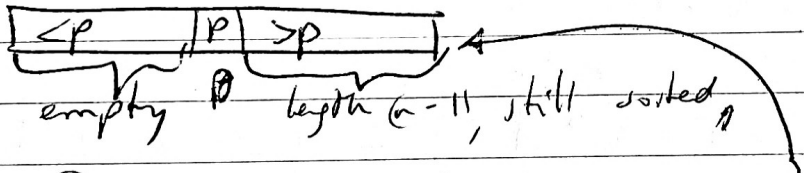
ChoosePivot(A, n) (Array A of length n)

The importance of the Pivot.

Q: running time of Quicksort? Can't discuss now, depends crucially on quality of the pivot chosen

For a Choosepivot = 1st elem, $O(n^2)$ for sorted array

Why?



: QuickSort calls itself again on array of len (n-1), then (n-2) etc $n(n-1) < O(n^2)$

If ChoosePivot found Median element at each subarray, $O(n \log(n))$ (like merge sort), split in two etc)

BIG IDEA: Random Pivots!

Randomised algorithm, will run differently even with the same input because randomness is intrinsic to it

Why could it possibly work?

① Hopefully a random pivot is "pretty good" often enough

② Intuition: approximately balanced split will give good res too. If always get 25-75 ~~%~~, still $O(\lg n)$

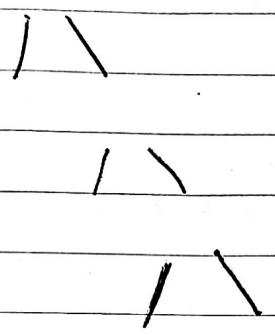
1st + 3rd
$a = 1$
$b = 1$
$d = 1$

→

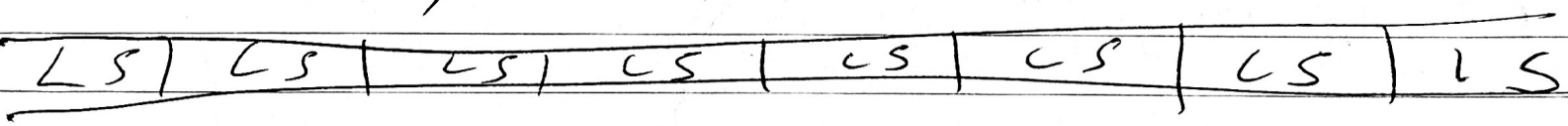
$a = 2$
$b = \left(\frac{1}{0,25} + \frac{1}{0,75}\right)/2 =$
$d = 1$

$0,75^5 = 0,25$, after 5 levels, $5+2$, $b = 1,5$

$b \geq 1,4$ for $O(n \log(n))$



? does it actually work?



Proof that Θ of quicksort is $\Theta(n \log(n))$

Fix input Array of length n

* Sample space Ω = all possible outcomes of random choice in Quicksort

Key notion: for $\sigma \in \Omega$, $C(\sigma)$ = # of comparisons made by quicksort between two input elements

Lemma: running time of Quicksort = # comparisons

Note: Can't apply master method (random, unbalanced subp b/ten

BUILDING BLOCKS

Notation: z_i = ith smallest element of A

z_i is not the element in the ith position of the array (usually)

For $\sigma \in \Omega$, indices $i < j$, let $X_{ij}(\sigma)$ = # of times z_i, z_j get compared with pivot separ

Can be 0 or 1, if ever pivot then 1, one's if not, then 0.

$$C(\sigma) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}(\sigma), \text{ by linearity of expectation}$$

$$E[C] = \sum_{i=1}^{n-1} \sum_{j=i+1} E[X_{ij}] =$$

Complicated! Simple!

$$E[X_{ij}] = 0 \cdot \text{blah blah blah} + 1 \cdot P[X_{ij} = 1] = P[X_{ij} = 1]$$

General Decomposition Principle (might I say)

1 Identify RVAR Y
2 Express Y in terms of indicator RVARS
3 Apply linearity of expectation

3e

LG | LG | LG | LG

Quicksort Theorem: $O(n \log(n))$ for every input.

Key claim: $\forall i < j$, $Pr(z_i, z_j \text{ get compared}) = x_{ij} = \frac{2}{(j-i+1)}$

Proof of Key claim:

fix z_i, z_j, $i < j$

consider $z_i, z_{i+1}, z_{i+2} \ldots\ldots z_{j-1}, z_j$

Inductively: as long as none of these misled:

if pivot smaller than all of these will end up on the right

" " bigger " " left

⇒ do no comparisons

comparison happens when pivot is picked within these.

These $(j-i+1)$ elements

Consider pivot is z_i or z_j, they will get compared

IF not, no comparison. EVER.

$\frac{2}{j-i+1}$ chances of this happening.

$E[C]$

So $= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(\frac{2}{j-i+1}\right) =$

$= \left(2\sum_{i=1}^{n-1}\right) \cdot \left(\sum_{j=i+1}^{n} \frac{1}{(j-i+1)} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots\right) \leq$

$= So. E[C] \leq 2 \cdot n(\text{choices for } i) \cdot \sum_{k=2}^{n} \frac{1}{k} \quad \int \frac{1}{x} = \log(x), \text{ smaller than integral} \leq 2n \log n$

Scribbles from Problems WK3

$\frac{\alpha}{0.5}$ $\frac{0.5}{\alpha}$

$\frac{\alpha}{0.5} = \frac{1}{2}$

$2\alpha = 1 \Rightarrow p$

$1 - 2\alpha$

$n(1-\alpha)$ will take longer for $\alpha < 0.5$

$n(1-\alpha)^d = 1$

$\log(n) + d\log(1-\alpha) = 0$

$d = \frac{-\log n^b}{\log(1-\alpha)} = -\log_{(1-\alpha)}(n)$

$\alpha = 0.5$,

$X_{ij} = \begin{cases} 0 \end{cases}$

$E[X] = \sum_{i=1}^{n-1} \sum_{j=i}^{n}$ Pr (i and) have same birthday)

$= \left(\frac{1}{365} \sum_{i=1}^{n-1}\right) \sum_{j=i+1}^{n}$ $= b$ i+1 + i+2 + ... + i+k

$\frac{2 \pm \sqrt{2900}}{2}$

34

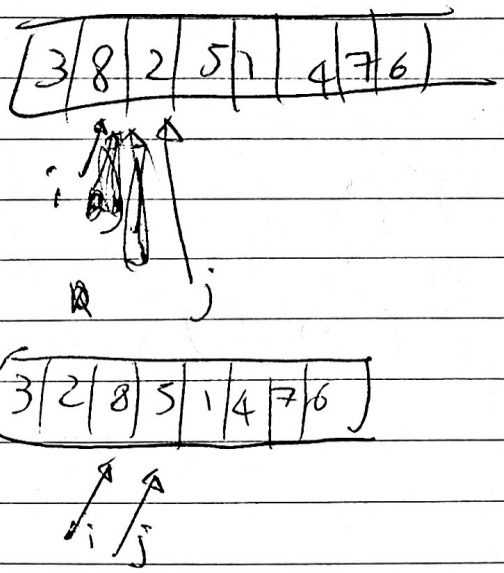
X product of x_1 and x_2
x_1 and x_3

not independent (consider $x_1 = 1$)

$E[YZ] = E[Y] \cdot E[Z]$

1
2 . 2
3 . 2
4 . 2
5 . 2
6 . 2
8

Fixing x_1, yes, but no!



WEEK 3 PROBABILITY REVIEW 1

Sample Spaces

Sample space Ω is a collection of all the things that could happen, all possible outcomes

will be dealing with this as finite set in algos

Also each outcome $i \in \Omega$ has a probability $p(i) \geq 0$

Constraint $\sum_{i \in \Omega} p(i) = 1$ Example: 2 dice rolls $\Omega \{(1,1),(1,2) \dots$

EVENTS

An event is a subset $S \subseteq \Omega$

Example: sum of rolls is 11, $S \subseteq \Omega$, $S = \{(5,6)(6,5)\}$

RANDOM VARIABLES

A random variable X is a real-valued function $X: \Omega \to \mathbb{R}$

Example: sum of the two dice

EXPECTATION

Average value, weighted by probability

$E[X]$ of $X = \sum_{i \in \Omega} X(i) \cdot p(i)$

5) LINEARITY OF EXPECTATIONS

Let $X_1, \ldots, X_n$ be RANDVARS, defined on Ω.

$$E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} E[X_i]$$

Sum of two slice ✓

Product X

Also
~~Simple~~

LOADING PROBLEM

Sample space Ω = all n^n possible assignments of processes to servers, each equally likely

Let Y = # of processes assigned to the first server.

Goal: compute $E(Y)$

Let $X_j = \begin{cases} 1 & \text{if } j\text{th process assigned to first server.} \\ 0 & \text{otherwise} \end{cases}$

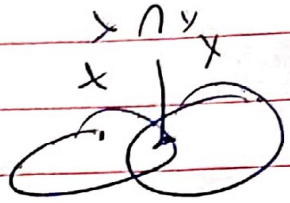
$E[Y] = \sum_{j=1}^{n} X_j$, since linearity, we just have to find X_j, $\frac{1}{n}$

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

WEEK 3 PROBABILITY REVIEW 2

6) CONDITIONAL PROBABILITY

Let $X, Y \subseteq \Omega$ be events

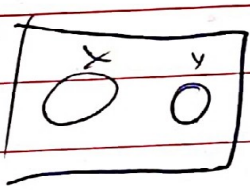


$$P_r[X \mid Y] = \frac{P[X \cap Y]}{P[Y]}$$

Given Y happened, what's the probability of X?

7) INDEPENDENCE OF EVENTS

X, Y are independent if and only if $P_r[X \cap Y] = P_r(X) \cdot P_r(Y)$



8) INDEPENDENCE OF RANDVARS

Definition: for A, B RANDVARS, independent if $P_r[A = a] \cdot P_r[B = b]$

are independent for all a, b

Do not do between sum n product of dice, product has prob of being 1 if sum = 12