6/25/2018 Quick Sort Week 3 ①

- Cool algorithm
- Run time : Average $n \cdot \log n$

How it works? (High Level)

Step 1: Pick element of arrey (Pivot element)

3	8	2	5	1	4	7	6

Pivot

Step 2: Rearrange array so that:
(Partition)

2	1	3	6	7	4	5	8

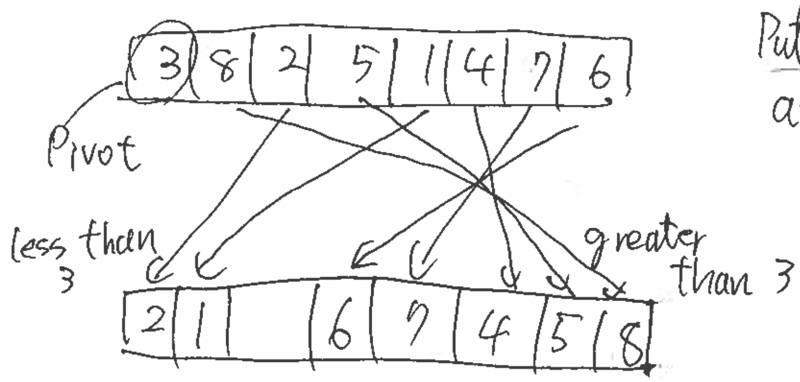
< pivot   > piot

Pivot element automatically comes to the right place.

Step 3: Recurrsively apply partitioning to the left side and the right side.

※ No Merge step!

How to partition? (Easy! But using additional memory)



Put the less on left end and ~~put the greater on right~~ end.

What is the method of partitioning use the least amount of memory? (In-place Implementation) ②

Step1: Swap and bring the chosen pivot element to the front of the array.

Step2: Keep track of the elements that you've already seen, and the elements that you've not seen yet.

p	<p	>p	? (No idea)

$\underbrace{\quad\quad\quad\quad}_{\text{already partitioned}}$ $\underbrace{\quad\quad\quad\quad}_{\text{unpartitioned}}$

Call the thing we scaned, are already partitioned)

E.g. Need two indices to keep track

p	<p	>p	?

i (Partition border)

j (Scaned border)

1

Pivot

3	8	2	5	1	4	7	6

i, j

unseen

2

3	8	2	5	1	4	7	6

i part. j unseen

3

part.

3	2	8	5	1	4	7	6

i j unseen

4

3	2	8	5	1	4	7	6

i j unseen

5

swap

3	2	1	5	8	4	7	6

i+1 j unseen

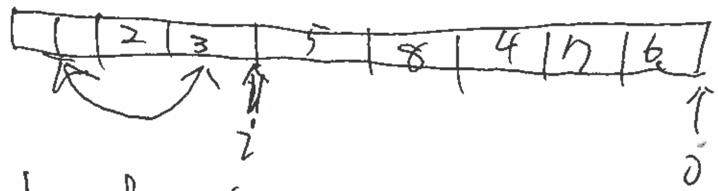
6

3	2	1	5	8	4	7	6

i

no change j

11 Bring the pivot element to the partition border



Pseudocode of partition

※ In divid in conqur algorithm, I should not store Divided array as new memory. Rather, it should be expressed as indices in original array.

```
Partition(A, l, r){
    P := A[l]    # ~~Partition~~ Pivot point.
    i := l+1     # Start of partition border
    for j=l+1 to r {
        if A[j] > P {
            Pass }
        if A[j] < P {
            Swap(A[j], A[i])
            i++ }
    }
    swap (A[l] and A[i-1])
}
```

※ Runtime $O(n)$.

※ No new array required.

What is running time of the quick sort?

⇒ Highly depends on the quality of the pivot element.

Worst case of quick sort

Input: Sorted array with first element as pivot element.

⇒ Unequal split at worst

$$\underbrace{< P}_{\text{empty}} \; | \; P \; | \; \underbrace{\geq P}_{\text{length } n-1 \text{ (still sorted)}} \longrightarrow \text{Pass in to recursion function.}$$

Runtime $\geq n + (n-1) + (n-2) + \cdots + 1$
$= O(n^2)$

Best case of quick sort

Assume every pivot element I get is a median value in that array.

Runtime: $T(n) \leq \underbrace{2T\left(\frac{n}{2}\right)}_{\text{half the size by median.}} + \underbrace{O(n)}_{\text{Change Pivot, Partition}} \Rightarrow O(n \cdot \log n)$

How we can chose good pivot?

⇒ Random pivots!

Pseudo Code of quick Sort

```
quicksort(Arr, left, right)
    if right - left <= 0   # when len of Arr is 1.
        return Arr                          (Base Case)
    else
        pivot = findpivot(Arr, left, right)
   Par= partationing(Arr, left, right, pivot)
        quicksort(Arr, left, par+1)
        quicksort(Arr, par+1, right)
```

6/30/2018 Probability ②

Concept #1 : Sample space Ω = "all possible outcomes"

Each outcome $i \in \Omega$ has a probability $p(i)$

Of course $\sum_{i \in \Omega} P(i) = 1$

e.g. Rolling two dice ($\Omega = \underbrace{\{(1,1), (1,2) \cdots (5,6), (6,6)\}}_{36 \text{ posibilities}}$)

$p(i) = 1/36$ for all $i \in \Omega$

Concept #2 : Events S = "Subset of Ω"

$S \subseteq \Omega$ ∴ probability of an event S is $\sum_{i \in S} p(i)$

e.g. Rolling two dice and sum of both become 7.

$$\sum_{S \in \Omega} P(i) = 6/36 = 1/6$$

$S = [(1,6)(6,1)(2,5)(5,2)(3,4)(4,3)]$

Concept #3 : Random Variables = "The value of output"

X : Random Variables

$X : \Omega = \mathbb{R}$

e.g. Rolling two ~~dies~~ dice and sum value

$X : \Omega = (2, 3, \cdots, 12)$

Concept #4 : Expectation = "Average of X"

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

Concept #5 : Linearity of Expectation

Given $X_1, \cdots X_n$ random variables exist on same sample space Ω,

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

※ Works even X_i's are not indepentent each other.

※ $E\left[\prod_{j=1}^{n} X_j\right] \neq \prod_{j=1}^{n} E[X_j]$

Only works when summing. Products not applicable.

e.g. Rolling two dice and expect the sum of two.

One way is

$E\left[\sum_{j=1}^{n=36} X_j\right]$ --- Take the averge of all possible (36 pattern) outcome

$\Rightarrow$ Don't want to do it.

$E\left[\sum_{j=1}^{n} X_j\right] = \sum_{j=1}^{n} E[X_j]$ --- Linearity of Expectation

$\sum_{j=1}^{n} E[X_j] = E[X_1] + E[X_2]$

$E[X_j] = \sum_{i \in \Omega} \underbrace{X_j(i)}_{\text{Dice value}} \cdot \underbrace{P(i)}_{\text{Probability}}$

$E[X_1] = \frac{1}{6}(1+2+3+4+5+6) = 3.5$

$E[X_2] = 3.5$

$E[X_1] + E[X_2] = 3.5 + 3.5 = 7.0$ //

e.g. Server problem

Have n number of server.
Have n number of process to assign to servers.
Q: What is the expectation number of processes assign to server?

Ω: all n^n assigments of processes to servers.
equally likely ($1/n^n$)

Goal: Compute $E[Y]$
where Y = total number of processes assigned to the first server.

Technically, we can do

$$\sum_{j=1}^{n^n} \underbrace{X_j}_{\substack{\text{no. of}\\ \text{assigned}\\ \text{process}}} \cdot \underbrace{p(j)}_{1/n^n} \Rightarrow \text{Hell no!}$$

Use Linearity of Ex.

Focus on single process assign on first server

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

Note: $\underbrace{Y}_{\substack{\text{total}\\ \text{number that}\\ \text{assigned to}\\ \text{first server}}} = \sum_{j=1}^{n} \underbrace{X_j}_{\substack{1 \text{ if } j\text{-th}\\ \text{process is assigned}}}$

process=[0, 0, 0, 0, 1, 0, 0, 0 ... 0, 0]
$\uparrow$ jth

$$E[Y] = E\left[\sum_{j=1}^{n} X_j\right]$$
$$= \sum_{j=1}^{n} E[X_j] \quad \text{(Lin-Exp.)}$$
$$= \sum_{j=1}^{n} (\underbrace{\cancel{P[X_j = 0]}}_{\frac{n-1}{n}} \cdot \underbrace{0}_{\text{value}} + \underbrace{P[X_j = 1]}_{1/n} \cdot \underbrace{1}_{\text{value}})$$
$$= n \cdot \frac{1}{n} = 1$$

Eg. Birthday problem

How many people n, need that expectation of the number of the pair of people who has same birthday is at least one?

Process (Birthday) $= [1, 2, 3, \cdots, 365]$

$\Rightarrow$ People $= [1, 2, \cdots n]$

Random variable X_{ij}

$$X_{ij} = \begin{cases} 1 & \text{if } i\text{-th and } j\text{-th people have same birthday} \\ 0 & \text{if otherwise} \end{cases}$$

Ω : Combination of i-th and j-th person.

$$nC_2 = \frac{n\cdot(n-1)}{2\cdot 1} = n(n-1)/2$$

$$E[Y] = E\Big[\sum_{i,j\in\Omega}^{\cancel{n(n-1)/2}} X_{ij}\Big]$$

$$= \sum_{i,j\in\Omega}^{\cancel{n(n-1)/2}} E[X_{ij}]$$

$$= \underbrace{\sum_{i,j\in\Omega}}_{n(n-1)/2} \Big(\underbrace{\cancel{P[X_{ij}=0]}}_{\text{Prob. that Birth unmatch}} \cdot \underbrace{0}_{\text{Value}} + \underbrace{P[X_{ij}=1]}_{\substack{\text{Prob. that Birth match} \\ \Downarrow \\ \frac{1}{365}}} \cdot \underbrace{1}_{\text{value}}\Big) = \underbrace{1}_{\substack{\text{At least} \\ \text{one pair} \\ \text{of matching}}}$$

$$\frac{n(n-1)}{2}\cdot\frac{1}{365} = 1$$

$$n^2 - n - 730 = 0$$

$$n = -26.523,\ 27.523$$

At least one pair when $n > 28$

~~Comp~~ Concept #6: Conditional probability

$$P[X|Y] = \frac{P[X \cap Y]}{P[Y]} \quad (X \text{ given } Y)$$

Concept #7: Independence of Events

$$P[X \cap Y] = P[X] \cdot P[Y]$$

iff $X, Y \subseteq \Omega$ are independant.

$E[A \cdot B] = E[A] \cdot E[B]$ only if A, B are independent.