Overview of Randomized algorithm

(Example: Hire and Fire Problem)

```
Hire assistant (n) →
{
  best ← 0 // candidate 0 is a least qualified
           dummy candidate
  for i ← 1 to n
    do interview candidate i
    if candidate i is better than candidate best
    Then best ← i
         hire candidate i
}
```

- Cost $O(nC_i + mC_h)$
- Worst Case :- hire all n candidates
- Probablistic analysis :-
  * Use Knowledge of or make assumption about the distribution of inputs
- Randomized algorithm :- use when input distribution is not-known or can not be modeled computationally
- Enforce random order if required.

Probabilistic Analysis :-

For the hiring problem. We can assume that candidates come in Random order

→ * Assign a rank(i), a unique integer in the range 1 to n to each candidate

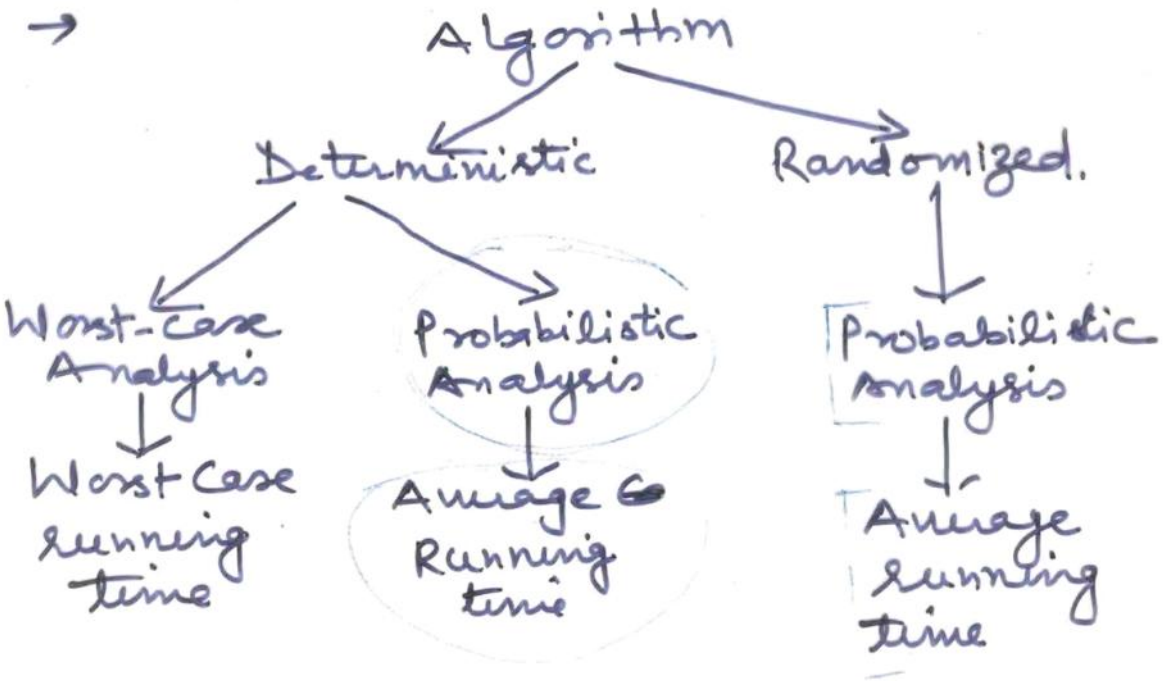
→ * The order list <rank(1), rank(2), ... rank(n)> is a permutation of the candidate numbers <1 2 .. n>

→ * Determine the no. of candidates who will be hired on an average assuming that the permutation of their ranks is (uniformly distributed).

Randomized-Hire assistant (n)

Randomly permute the list of candidate
best ← 0 // dummy variable
for i ← 1 to n
    do interview candidate i
    if candidate i is better than
                candidate best
    Then best ← i
    hire candidate i

→



Order statistics

- i^{th} order statistics :
  i^{th} smallest element of a set of n elements
- Minimum :- first order statistics
- Maximum :- n^{th} order statistics
- Median :- "half-way point" of the set
  * Unique :- when n is odd occur at $i = (n+1)/2$
  * Two medians when n is even
    - lower median, at $i = n/2$
    - upper median, at $i = n/2 + 1$

Note: for consistency, "median" will refer to the lower median

Selection Problem :-

Input :- A set A of n distinct numbers and a number i, with $1 \le i \le n$

Output :- the element $x \in A$ that is larger than exactly $i-1$ other elements of A

[illegible]

→ General selection problem –:
(selection in Expected linear time)

→ Modeled after randomized Quicksort

– Exploits the abilities of Randomized-partition (RP)

>> RP returns the index k in the sorted order of a randomly chosen element (pivot)

– If the order statistics, we are interested in i, equals k, then DONE

– Else reduce the problem size using it's other ability

>> RP rearranges the other elements around the random pivot

– if i < k, selection can be narrowed down to A[1..k-1]

– Else select the $(i-k)^{th}$ element from A[k+1..n]

Else select the $(i-k)^{th}$ element from $A[k+1..n]$

— { Randomized - Select } :-

```
Randomized-Select (A, p, r, i)
    if p = r
      then return A[p]
    q ← Randomized-Partition (A, p, r)
    k ← q - p + 1
    if i = k
      then return A[q]
    else if i < k
      then return Randomized-select(A, p, q-1, i)
    else return
      Randomized-select (A, q+1, r, i-k)
```

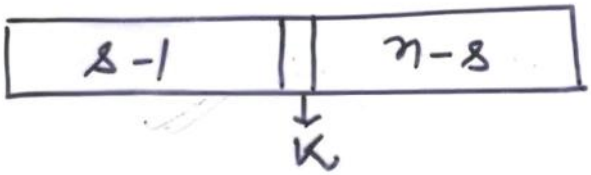
Analysis :-

Worst Case : $\Theta(n^2)$ ✓

Average case : $\Theta(n)$ ✓

Selection :-

→ In order to find the K^{th} order statistics in a region of size n, use Randomized partition to split the region into two subarray.



if $K = s$, the pivot is the key that's looked for.

→ if $K \leq s-1$, look for the K^{th} element in the left subarray

→ Otherwise look for the $(K-s)^{th}$ one in the right subarray.

Analysis :

Let $T(n)$ be the expected running time $T(n)$.

for each i, $0 \le i \le n-1$

size of left subarray is equal to i with the probability $1/n$

→ Assume that the larger interval is taken, for some $\alpha > 0$.
$T(n)$ is at most

$$\alpha n + \frac{1}{n} \sum_{1 \le k \le n-1,\ k \ne s} T(\max(k, n-k))$$

at most ⇒

$$\alpha n + \frac{2}{n}\left(\sum_{k=\lceil n/2 \rceil}^{n-1} T(k)\right)$$

~~$c_p \leq c_k \; \forall k$~~

→ Assume that there is $c>0$ such that $T(k) \leq ck \; \forall \; k<n$.

then the sum $\sum_{k=\lceil n/2 \rceil}^{n-1} T(k)$ is at

most $\sum_{k=\lceil n/2 \rceil}^{n-1} ck$. this is at most.

$$\sum_{k=2}^{n-1} ck - \sum_{k=1}^{\lceil n/2 \rceil - 1} ck.$$

$$= \frac{cn(n-1)}{2} - c\left(\lceil \tfrac{n}{2} \rceil - 1\right)\lceil \tfrac{n}{2} \rceil$$

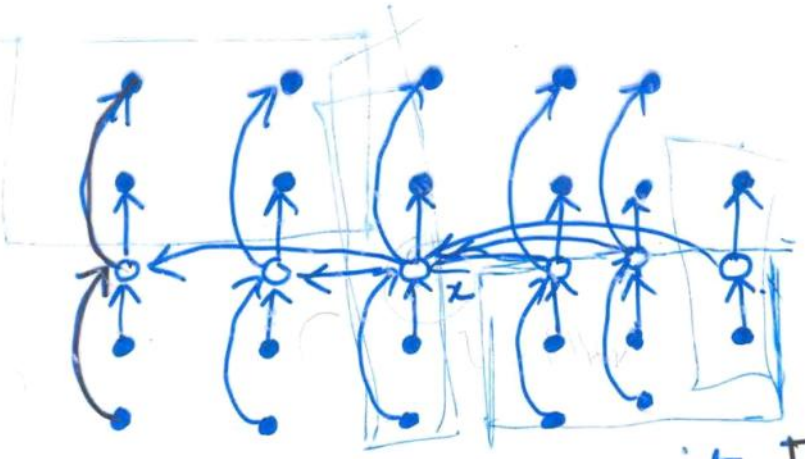
$$\leq \frac{cn(n-1)}{2} - \frac{c}{2}\left(\frac{n}{2} - 1\right)\frac{n}{2}$$

$$= cn\left(\frac{3n}{8} - \frac{1}{4}\right)$$

~~if c is~~

✓ ✓

Selection in Worst Case linear time



Step 1 Divide n elements into $\lceil n/5 \rceil$ group of 5 elements

Step 2 Find the median of each group.

Step 3 Use select ~~seq~~ recursively to find the median x of the above $\lceil n/5 \rceil$ medians

Step 4 Partition n elements around x into S_1, S_2 and S_3

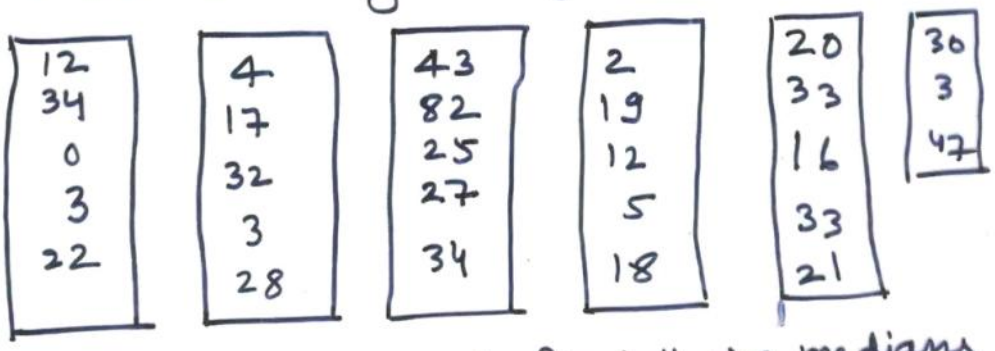
Step 5 if $|S_1| > i$, search ith smallest element in S_1 recursively
else if $|S_1| + |S_2| > i$, then return x (the ith smallest element)
Else search $(i - (|S_1| + |S_2|))$th in S_3

Example

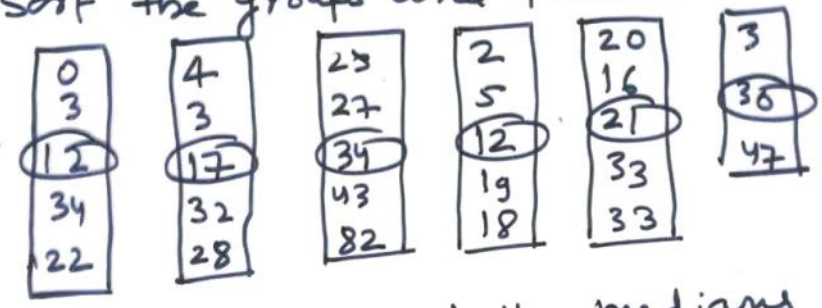
Find the 11th smallest element in the array

A = {12, 34, 0, 3, 22, 4, 17, 32, 3, 28, 43, 82, 25, 27, 34, 2, 19, 12, 5, 18, 20, 33, 16, 33, 21, 30, 3, 47}

1. Divide the array into groups of 5 elements



2. Sort the groups and find their medians



3. Find the median of the medians

12, 12, (17), 21, 34, 30

4. Partition the array around the median of medians (17)

First partition: {12, 0, 3, 4, 3, 2, 12, 5, 16, 3}

pivot = 17 (position of pivot wrt all)

Second partition

{34, 22, 32, 28, 43, 82, 25, 27, 34, 19, 18, 20, 33, 33, 21, 30, 47}

→ To find 6th smallest element, recurse our search in first partition. (10+1)

Step 1, 2 & take $O(n)$ ✓

Step 3 takes $T(\lceil n/5 \rceil)$ ✓

Step 5 :-

* At least half of the medians in step 2 are $\geq x$. Thus at least $\frac{1}{2}\lceil n/5 \rceil - 2$ groups contribute 3 elements which are $\geq x$ i.e

$$3\left(\lceil \tfrac{1}{2}\lceil n/5 \rceil \rceil - 2\right) \geq \frac{3n}{10} - 6$$

* Similiarly the number of elements $\leq x$ is also at least $(3n/10) - 6$

* Thus, $|S_1|$ is at most $\left(\frac{7n}{10}\right) + 6$, similiarly for $|S_3|$ $\quad \{n - (\frac{3n}{10} - 6)\}$

* Thus Select in step 5 is called Recursively on at most $(7n/10) + 6$ elements

— Recurrence is

$$T(n) = \begin{cases} O(1) & \text{if } n < \text{some value} \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) & \text{if } n \geq \text{some value.} \end{cases}$$

solution

- suppose $T(n) \leq cn$ for some c
- $T(n) \leq c\lceil n/5 \rceil + c(7n/10 + 6) + 9n$

$$\leq \frac{cn}{5} + c + \frac{7}{10}cn + 6c + 9n$$

$$= \frac{9}{10}cn + 9n + 7c$$

$$= cn + (-cn/10 + 9n + 7c)$$

- which is at most cn

iff $-\frac{cn}{10} + 9n + 7c < 0$ ✓

$\Rightarrow c \geq 10 \cdot 9\left(\frac{n}{n-70}\right)$ when $n > 70$

- so select $n = 140$ and then

$c \geq 20 \cdot 9$ ✓