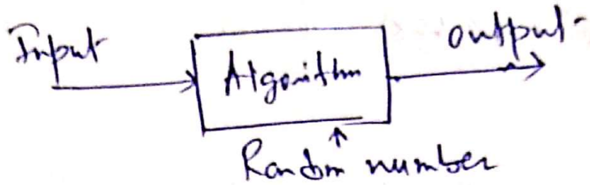
RANDOMIZED ALGORITHM

A randomized algorithm is one that makes use of randomizer (such as an ~~randomizer~~ random number generator).



- The output of a randomized algorithm can vary/differ from run to run for the same input.
- The execution time of a randomized algorithm could also vary from run to run for the same input.

Randomized algorithm can be categorized into two classes:

Las Vegas

Algorithm that always produce same (correct) output for the same input. The execution time of a Las Vegas algorithm depends on the output of the randomizer. The running time may vary between execution.

Example: Randomized Quicksort Algorithm

Monte Carlo

Outputs might differ from run to run for the same input. If a Monte Carlo algorithm is employed to solve such a problem, then the algorithm might give incorrect answers depending on the output of the randomizer.

Example: Randomized MINCUT Algorithm

Randomized Quicksort Algorithm

Input: A set S of n integers. $S = \{a_1, a_2, \cdots a_n\}$

Output: The sorted version of S. $a_{i_1} \le a_{i_2} \le a_{i_3} \cdots \le a_{i_n}$

Requirement:

1. Choose a number of y uniformly random from S.
2. (a) Construct the subset S_1 containing all elements of S which are less than y.
   (b) Construct the subset S_2 containing all elements of S which are larger than y.
3. Recursively sort S_1 and S_2. Output ⇒ S_1 followed by y, followed by S_2.

Array → $A[p \ldots r]$, pivot element $x = A[r]$. The pivot element, Randomly chosen element from the array $A[p \ldots r]$.

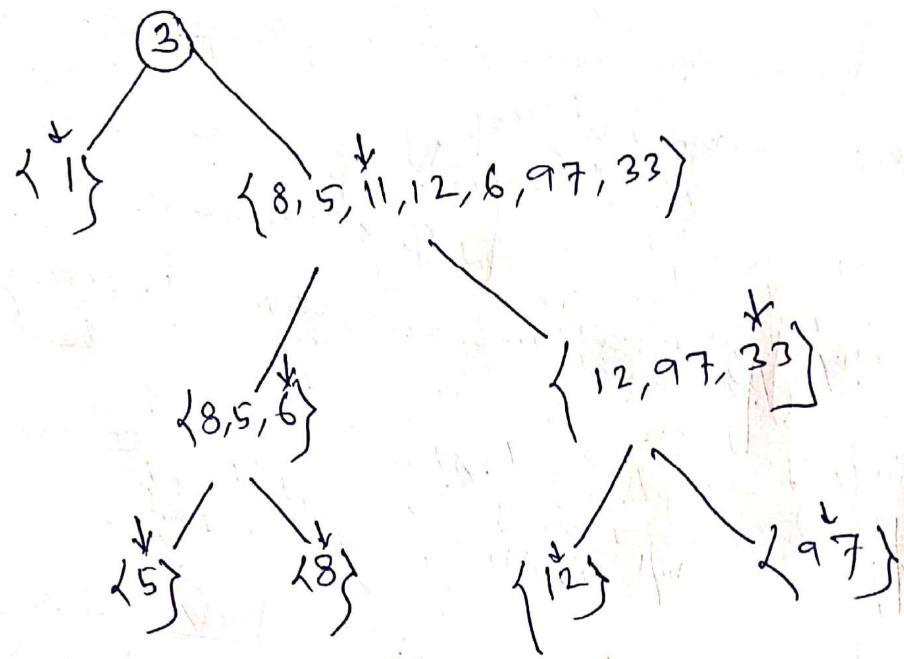
RANDOMIZED-PARTITION (A, p, r)

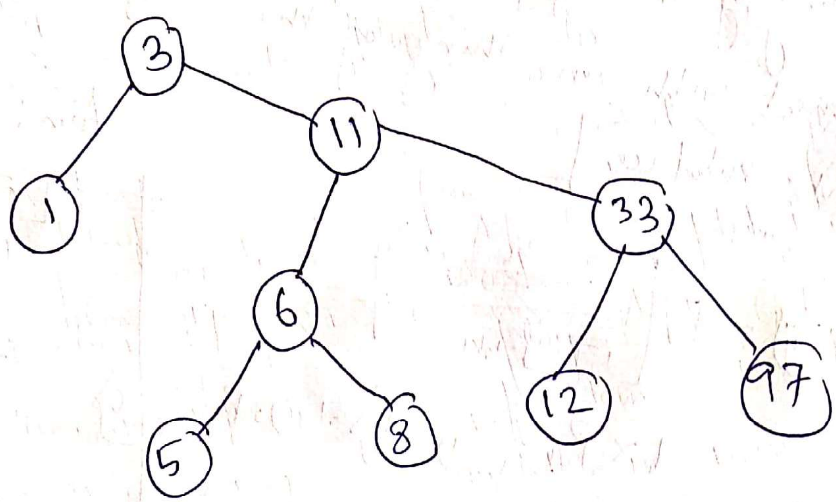
1. i = RANDOM(p, r)
2. exchange $A[r] \leftrightarrow A[i]$
3. return PARTITION (A, p, r)

RANDOMIZED-QUICKSORT (A, p, r)

1. if $p < r$ then
2. q = RANDOMIZED-PARTITION (A, p, r)
3. RANDOMIZED-QUICKSORT $(A, p, q-1)$
4. RANDOMIZED-QUICKSORT $(A, q+1, r)$

Example: {1, 8, 5, 3, 11, 12, 6, 97, 33}





Inorder Traversal of the tree ⇒ elements are sorted order

{1, 3, 5, 6, 8, 11, 12, 33, 97}

Analysis (Expected running-time)

S_i denotes the fixed element in S with rank i

We introduce a random variable

$$X_{ij} = \begin{cases} 1 & \text{if } S_i \text{ and } S_j \text{ are compared} \\ 0 & \text{otherwise} \end{cases}$$

So total no. of comparisions are bound (running time of the algorithm)

$$X = \sum_{i<j} X_{ij} = \sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij} \quad (\text{no. of comparisons})$$

Fact: S_i & S_j are compared iff S_i is a parent of S_j and vice versa

[Any thing of the left subtree is not compared with any thing of the right subtree]

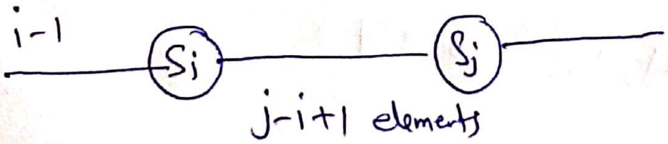
Aim: Compute $E[X]$

$$E[X] = \sum_{i<j} E[X_{ij}] = \sum_{i<j} P_{ij} = \sum_{i=1}^{n} \sum_{j=i+1}^{n} P_{ij}$$

$$\therefore E[X_{ij}] = \cancel{E\left[\sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{ij}\right]}$$

$$= 1 \times Pr\{X_{ij} = 1\} + 0 \cdot Pr\{X_{ij} = 0\}$$

$$= P_{ij}$$



Fact: S_i and S_j are compared iff S_i & S_j is chosen ahead of the elements in between S_i and S_j

We compute probability that this event occurs. $S=(S_i \text{ to } S_j)$

Prior to the point at which an element S has been chosen as a pivot, the whole set S is together in the same partition. Therefore any element of S is equally likely to be the first one chosen as a pivot. Because the set S has $j-i+1$ elements, the probability that any given element is the first one chosen as a pivot is $\frac{1}{j-i+1}$. Thus we have

$$P_{ij}\{S_i \text{ is compared to } S_j\} = P_{ij}\{S_i \text{ or } S_j \text{ is first pivot element chosen from } S\}$$

$$P_{ij} = P_{ij}\{S_i \text{ is first pivot chosen from } S\} + P_{ij}\{S_j \text{ is first pivot chosen from } S\}$$

$$= \frac{1}{j-i+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1}$$

$$E[X] = \sum_{i<j} P_{ij} = \sum_{i=1}^{n}\sum_{j=i+1}^{n} P_{ij} = \sum_{i=1}^{n}\sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

$$\boxed{\sum_{k=2}^{n} \frac{2}{k} \leq H_n}$$

H_n denotes Harmonic number $= \frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\dots\frac{1}{n} \approx \log n$

$$= \sum_{i=1}^{n}\sum_{k=2}^{n} \frac{2}{k}$$

$$\leq \sum_{i=1}^{n} H_n \approx n \log n$$

$$E[X] = O(n \log n)$$