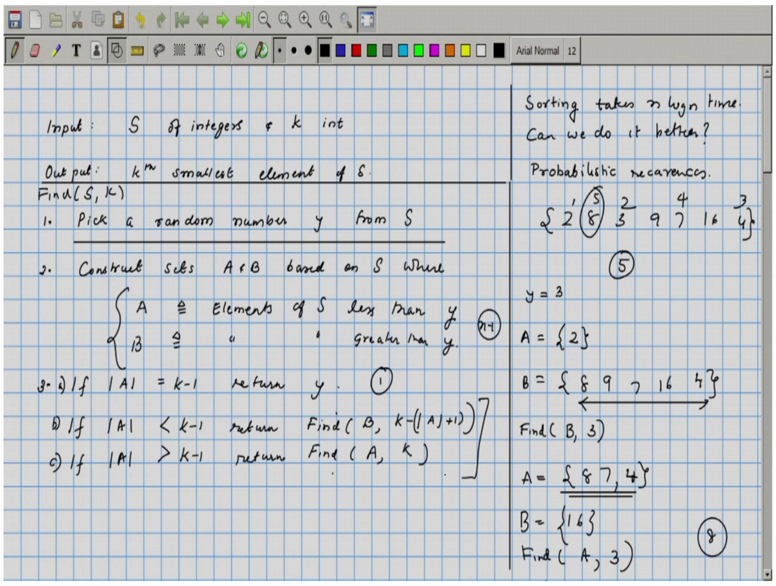
Randomized Algorithms

Prof. Benny George Kenkireth
Department of Computer Science & Engineering
Indian Institute of Technology, Guwahati

Lecture – 03
Randomized Find

So, in this lecture we will learn about find algorithm, is the algorithmic issue that we are interested in is as following.

(Refer Slide Time: 00:38)



Input : S of integers & k int

Output : k^{th} smallest element of S.

```
Find(S, k)
1. Pick a random number y from S
2. Construct sets A & B based on S where
     { A ≜ Elements of S less than y
     { B ≜    "        "  Greater than y.      (n-1)
3. a) If |A| = k-1 return y.   (1)
   b) If |A| < k-1 return Find(B, k-(|A|+1))
   c) If |A| > k-1 return Find(A, k)
```

Sorting takes n log n time.
Can we do it better?

Probabilistic recurrences.

$\{2 \ \overset{1}{} \ \overset{5}{8} \ \overset{2}{3} \ 9 \ \overset{4}{7} \ 16 \ \overset{3}{4}\}$

5

$y = 3$

$A = \{2\}$

$B = \{8 \ 9 \ 7 \ 16 \ 4\}$

Find(B, 3)

$A = \{8 \ 7, 4\}$

$B = \{16\}$

Find(A, 3)

8

The input is a set S of integers and a number k which is also an integer positive integer. And we have to find the kth smallest element of S. The straight forward approach for solving this would be to (Refer Time: 01:17) and output the kth element, but that would take and log n time. Assorting takes n log n time question is can we do it better? Of course, there are linear time algorithms; linear time deterministic algorithms to solve this problem.

We will see a simple randomization. The linear time algorithm which the deterministic linear time algorithm is little more complicated, we will see that a simple randomized algorithm can do this job. And while doing we will also learn about probabilistic recurrences ok. So, let us look at that. So, our algorithm is the steps are as follows, first we pick a random number Y from S and based on this randomly chosen number we split the entire set into 2 parts. So, we are (Refer Time: 02:45) to do the set of numbers which are smaller than Y and the set of number which is greater than Y and based on the number of elements in these set we recurs, that is the basic idea of the algorithm ok.

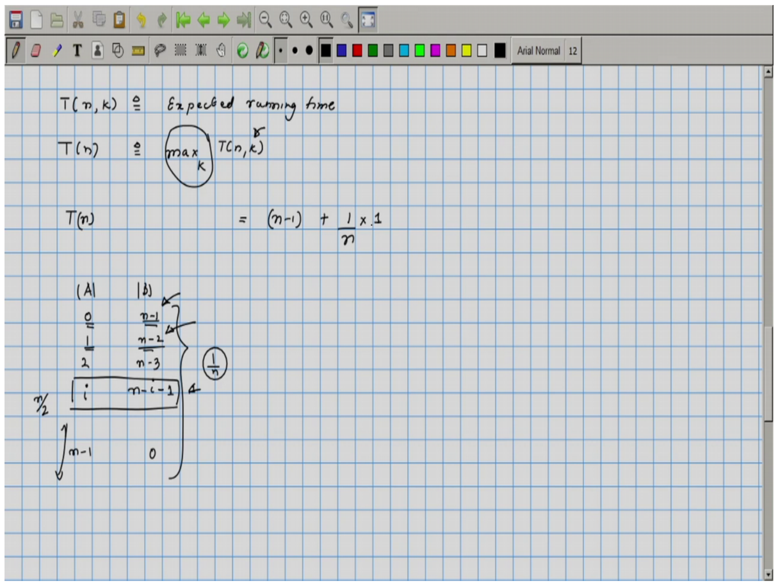
So, we will construct sets A and B based on S. So, A consist of elements less than elements of S less than say y and B is the set of elements of S greater than y and the third step would be. So, if size of A is equal to k minus 1. So, we will assume for the time being that S contains distinct element there no repetition of elements. If A contains k minus 1 elements, then return y as the answer. The next condition will be if size of A is less than k minus 1 that would mean that the kth smallest element is not the element we have chosen it is in the set of elements which are greater than y. So, return we will call this algorithm as find S comma k.

So, here what we will return is return find B. So, in B we will find the k minus size of A plus first element. So, already we have identified small a which are less than the kth smallest elements and the element Y chosen is also not the Kth smallest. So, all those elements have gone will return from d the third condition if A is greater than k minus 1 then we know that our element basically lies in side A. So, will return find A comma k simple natural algorithm. So, far collection was say 2 8 3 9 ok. So, this contain 7 elements and if we wanted to find the fifth smallest element, suppose we randomly choose 3 then. So, so suppose y is equal to 3 and based on 3 we split we have our A is equal to the set consisting of 2, and B consist of every other element so 8 9 7 16 and 4 ok.

So, we know that the fifth largest must be in this collection and B and it must be the. So, we will basically find the third smallest, because the first 2 small elements are basically in A and Y find the third smallest in B. So, B comma 3 would be the call and again let us say we choose a random element, suppose we had chosen let us say 9 ok. So, in that case our A would essentially be 8 7 4 and B would just contain 16 and there already 3 elements in this collection. So, we can skip all the other element. So, we will basically find A raise now 8 7 4. So, find the third element in A and the recursive (Refer Time: 07:23) would return 8 as the fifth largest element the answer is correct 1 2 3 4 and this is the fifth element.

So, that is our algorithm, how do we analyze this algorithm? The worst case running time of these of course, be very bad, but what we want is the expected running time of these algorithm that could depend on k.

(Refer Slide Time: 07:58)



So, what we will do is we will define T n comma k as the expected running time on any input, I mean the worst case expected running time for an input of size n and we are interested in finding the kth smallest element and we can define T n to be max over all with possible values of k T n k ok. So, amongst all possible k s whichever k is the worst k for n the value of T n k at that point is T n ok.

So, let us write down a recurrence for T n if you had an input of size n how much time does it take in an expected sense? So, we had the following situations, if the element that we pick is the kth smallest element we could be very lucky and we could just pick the middle element, all those element have to be compared with all the other elements. So, after this step after this pick this construction of A and B could take n minus 1 comparisons.

In all cases it takes n minus 1 comparisons and after that we require additional comparisons, one of these comparisons. And the cost of those comparisons will be based on our choice y and we have to find the average value over all possible choices. So, with probability 1 by n we could just require one comparison and each of these other choices could happen based on the number that we have picked. So, suppose.

So, let us look at these sets A and B in their sizes. So, based on the split we could have either the size of A as 0 and B as let us say n minus 1. Both of them are mean this will depend upon which is the k, which is the y that you have pick. It could also be 1 and let us say n minus 2 2 and n minus 3 n minus 1 and 0.

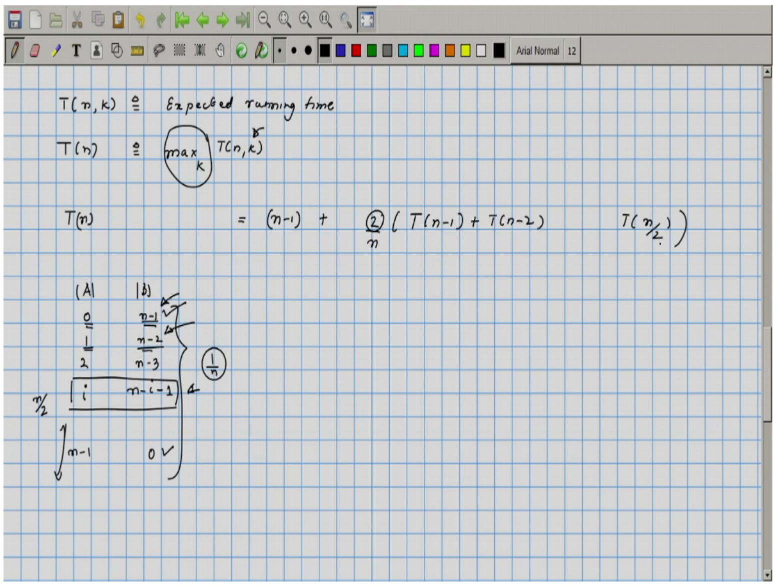
So, all these could be equally likely with probability 1 by n. Once again the pivot that we pick or the element y that we pick will uniquely split the entire set into A and B. And size of A could be 0 to n minus 1, size of B also could be 0 to n minus 1 once the size of A is fixed the size of B is automatically fixed and all these are equally likely. Let us look at the case where this is of size i and this I mean A is of size i and B is of size n minus i minus 1 ok.

Now, in this case whether recurs in i or whether we recurs in n minus i minus 1 depends on the value of k. If k was smaller than i then we would have recurs in A otherwise we would have recurs in B. But since we having the taking the average of all possible I mean we are interested in finding the worst case behavior over k, because here we are looking max over all possible values of k, we could assume that we will be very pessimistic and we could say that always we will be recursing on the larger set ok.

So, this can be and therefore, whenever we are confronted with the choice is 0 n minus 1, we will basically be recursing in n minus 1. Whenever we are confronted with the choice of 1 minus 2, this is where we will be. And all the way of 2 lecture of n by 2, we would be recursing in the other half and if you look at it carefully the terms basically repeat.

So, we can say that T n is equal to n minus 1, this is for the fixed comparison that you have to make to separate the elements into sets of size I mean sets which is smaller than Y and greater than Y plus if you are lucky you could take let us say and with probability 1 by n you have to take just return the correct answer. So, let us say that is constant time. So, 1 by n into 1 or if you are looking at the number of comparison we could just say that this does not happen at all plus 2 by n into T n minus 1.

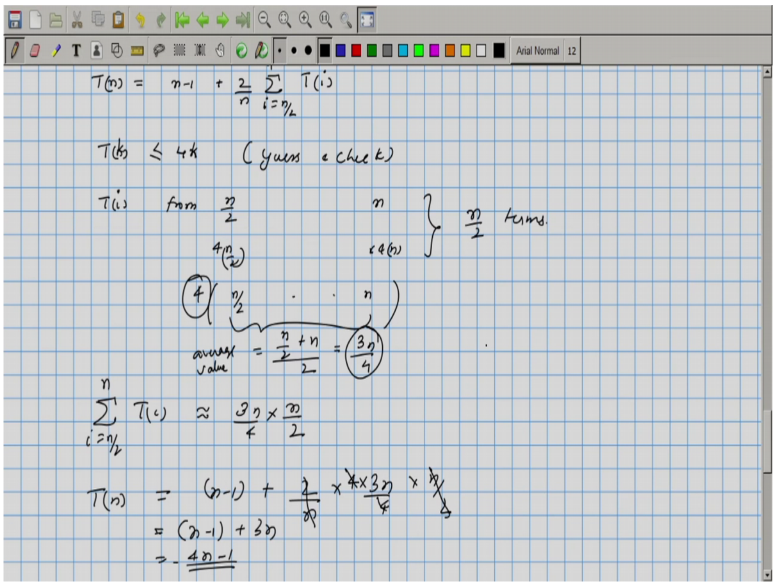
(Refer Slide Time: 13:10)



T(n, k) ≙ Expected running time
T(n) ≙ max_k T(n, k)
T(n) = (n-1) + 2/n (T(n-1) + T(n-2) ... T(n/2))
(A) (B)
0 n-1
1 n-2
2 n-3
i n-i-1
n/2
n-1 0
1/n

So, T n minus 1; so, T n minus 1 appears in both these situations so, that is why this 2 is coming. So, all these could have appear in the possibilities for A and B each of them appears with 1 by n probability, but 0 n minus 1 in that case we recurs on this set of n minus 1 and n minus 1 0 again we recurs in the set of size n minus 1. So, that accounts with 2 into T n minus 1 plus T n minus 2 all the way of to T n by 2 ok.

(Refer Slide Time: 13:50)



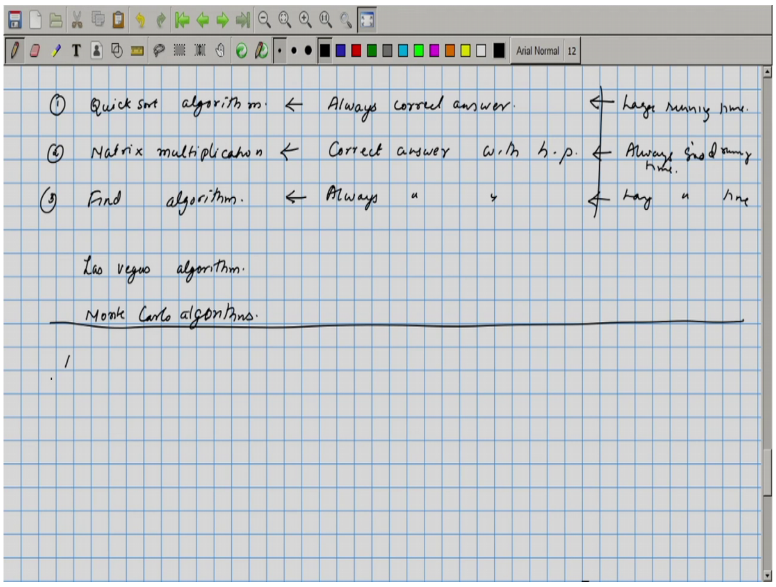
T(n) = n-1 + 2/n Σ_{i=n/2} T(i)
T(k) ≤ 4k (guess & check)
T(i) from n/2 ... n } n/2 terms
4(n/2) ... ≤ 4(n)
4(n/2 . . n)
average value = (n/2 + n)/2 = 3n/4
Σ_{i=n/2}^{n} T(i) ≈ 3n/4 × n/2
T(n) = (n-1) + 2/n × 4×3n/4 × n/2
= (n-1) + 3n
= 4n-1

So, this is the recurrence that we have, T n is equal to n minus 1 plus 2 by n times summation T i i varying from n by 2 to n ok. So, if you look at these the summation there are precisely n by 2 terms or n by 2 plus 1 term from together and they vary from T i to T i plus I mean from n by 2 to n ok. So, we been to guess that we could guess and solve. So, let us check whether T n is less than or equal 4 n ok. So, suppose we plug in the value that T n of T k is equal to 4 k is less than or equal to 4 k we can see if this recurrence is satisfied now if this recurrence is satisfied then we know that T n is less than or equal to 4 k ok.

So, when we plug that n T i from n by 2 to n takes value from let us say 4 n by 2 to 4 n ok. So, this is precisely I mean this is roughly n by 2 terms and their values are varying from 2 n, let us take the 4 out 4 times n by 2 to n and the average value of the these is n by 2 plus n by 2 ok. So, that is going to be 3 n by 4 ok. So, these are n by 2 terms with an average value 3 n by 4. So, we can say that summation i equals n by 2 to n T i is approximately equal to 3 n by 4 into n by 2 because there are n by 2 terms, you can do the calculation more precisely and check it, but this itself is sufficient for us. So, this tells us that T n is equal to n minus 1 plus 2 by n into 3 n by 4 into n by 2 ok. So, this cancels.

So, over all you will get this as that is the 4 here 4 into 3 n by 4. So, this is 4 times 3 n by 4 so 3 n. So, n minus 1 plus 3 n which is 4 n minus 1 ok. So, therefore, we can say that the average time taken is going to be less than or equal to 4 n its a linear time algorithm ok. So, this particular week the things we have covered.

(Refer Slide Time: 17:01)



① Quick sort algorithm.	←	Always correct answer.	←	Large running time.
② Matrix multiplication	←	Correct answer with h.p.	←	Always good running time.
③ Find algorithm.	←	Always " "	←	Large " time

Las vegas algorithm.

Monte Carlo algorithms.

First we looked at quick sort algorithm and then we had looked at another algorithm for matrix multiplication, this was the third algorithm called find algorithm ok. So, all these algorithms used randomness in a certain way, but if we look at these algorithm closely there are some differences between them. Quick sort for example, always get the correct answer. This find algorithm also always get the correct answer whereas, matrix multiplication we had this property that the correct answer is given with high probability, but the quick sort algorithm could have large running time in certain runs. Find algorithm also that problem the running time could be high.

So, large running time is the drawback of this algorithm where as matrix was giving correct answer with high probability, but the running time was to say always good running time. In the sense it would never take more than I mean the algorithm will always run in on square. So, this basically is the 2 types of randomized algorithm. The first algorithm which always gives the correct answer is called a Las Vegas algorithm whereas, the algorithms which sometimes give an incorrect answer they are called as Monte Carlo algorithms which one is better will depends upon our needs.

If we wanted an algorithm which will always give the correct answer. If the correctness of the answer is (Refer Time: 19:29) then we will have to go with Los Vegas algorithm. But if we want the algorithm to be first and we can live with a small we can take a small chance of error then Monte Carlo algorithms are usefully used ok.

That is all that we will have for this week. In the next week we will look at axiomatic definitions of probability. We will have review of probability that we would be needing in the later on lectures.