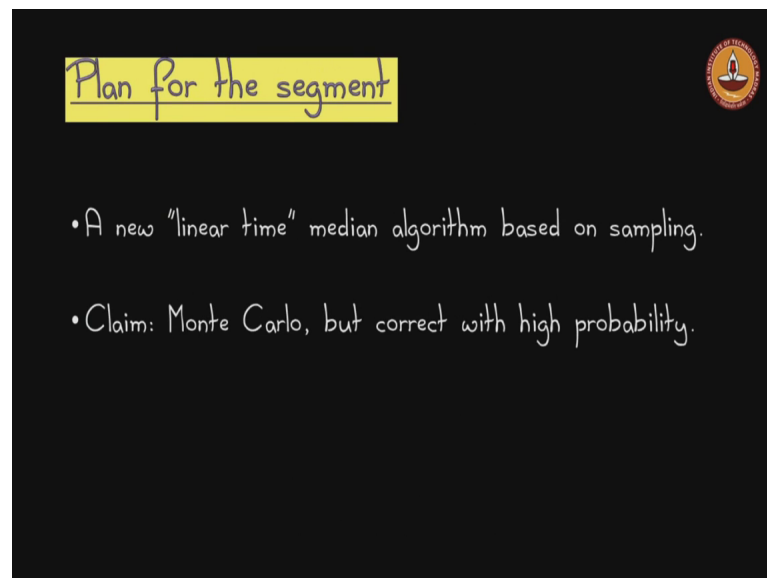


Probability & Computing
Prof. John Augustine
Department of Computer science and Engineering
Indian Institute of Technology Madras

Module – 3
Tail Bounds I
Lecture - 17
Segment 3: Median via sampling

So, we are now starting segment 3 in module 3 where were going to be talking about a median algorithm, algorithm to find the median we are going to the technique where algorithmic technique we are going to use this is sampling ok.

(Refer slide Time: 00:29)



Plan for the segment

- A new "linear time" median algorithm based on sampling.
- Claim: Monte Carlo, but correct with high probability.

So, this algorithm is going to be a linear time algorithm meaning it is going to be O of n where n is the size of the array it is going to be a Monte carol algorithm it may as stated, we can talk about how to convert it into a Las Vegas algorithm ah, so let us just state the problem.

(Refer slide Time: 00:52)

Finding the Median

Given Input

- An arbitrarily ordered array S of n numbers. (repetitions OK)

Required Output

- Find the median element in S .

The slide includes a logo in the top right corner and a small video inset of a speaker in the bottom right corner.

Now, I am going to focus only on the median ah, but extending it to the selection problem is not going to be difficult. So, you are given a set S of n numbers arbitrarily ordered and our goal is to find the median element in S .

(Refer slide Time: 01:16)

Sampling based Randomized Median Alg.

S [array of 15 cells]

Unsorted

↓

A multiset R of $\lceil \frac{3}{4}n \rceil$ elements chosen indep. & UAR from S

The slide includes a logo in the top right corner and a small video inset of a speaker in the bottom right corner.

So, here how the algorithm works you take the unsorted array S and we sample a multiset R of this n to the 3 fourths number of elements ok. So, each one of them is chosen independently and uniformly at random from S ok, just pick a random number from 1 to n whatever the element index it lands on drag it into the set R .

Student: It is with replacement.

It is it is with replacement. So, basically you can that is why this R will be a multi set , it you could be sampling the same element more than once and this is often the case because it makes the analysis simpler, you can take advantage of independents and things like that and you could probably do a little bit better if you use if you do it without replacement, but it is probably not worth.

It is not worth it for a few reasons, it is not a it is not worth it from them from an algorithmic point of view because, now if you have to do this without replacement; that means, this S has to be suitably updated after you pick an element to avoid picking it again ok.

So, there is an algorithmic complication that comes with it, there is also an analysis complication that comes with it. So, now dependencies can creep in, so you need to be a little bit careful. So, with all of that the a lot of times if we can do away with it we will just do the sampling with replacement ok. So, then life becomes very simple both algorithmically and from the analysis point of view.

(Refer slide Time: 03:31)

The slide is titled "Sampling based Randomized Median Alg." and features a logo in the top right corner. It contains the following elements:

- A blue cloud-shaped box containing the text: "A multiset R of $\lceil n^4 \rceil$ elements chosen indep. & UAR from S ".
- A purple arrow pointing downwards with the text "Sort into array" next to it.
- A horizontal array of colored blocks representing a sorted array. The array is divided into three sections: a left section of blue blocks, a middle section of a single orange block labeled d , and a right section of blue blocks. Below the array, the sizes of these sections are given as $\lceil n^4 - \sqrt{n} \rceil$, $\lceil n^4 \rceil$, and $\lceil n^4 + \sqrt{n} \rceil$ respectively.
- A purple rounded rectangle on the right side containing the text "Intuition:" followed by two bullet points:
 - d is less than but close to the median
 - u is greater than but close to the median

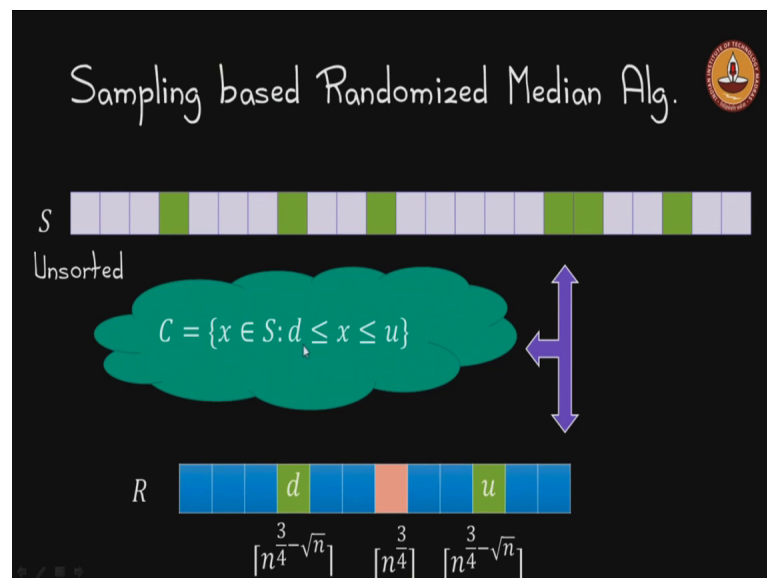
So, now we have the sample set art what are we going to do with it, now what we do is we take that sample set R and we sort it and so immediately you should be worried

sorting is expensive, but look the size of the this the array that we are sorting is only n to the 3 fourths.

So, now if you use an $n \log n$ time algorithm to sort it you are still going to be only taking little O of n amount of time ok. So, will sort it and our hope is the following that will be able to spot elements in this sample that are close to the median, but guaranteed to be less than one guaranteed to be less than the median and one guaranteed to be little to the right of the median ok.

So, that is let us let and well see why that is the case. So, once we sort it this is the sorted array R and I have showed it over here. So, this is a sorted array R , this is the middle element in this in the array R this is just a sampled elements ok. So, now from middle element you walk to the left square root of n steps, you get the element whatever is there we call it the element d . From the middle element you walk to the right for some square root of n steps and whatever element you get over there, you call it you u and our rest of the algorithm is going to hinge on the kind on the condition, if you will that d is less than the median in the in the original array, but it is close to the median and similarly u is greater than the median, but so close to the median.

(Refer slide Time: 05:20)



If that is the case then what can you do? Well you can scan through the entire array S and pick all the elements. So, create this list of elements C ok, it is all the elements in S that are lying between the values of d and u . Remember d and u are these elements d and u

which and this will require a full scan that is O of n time, but that is still O of n ok. What is the advantage with this? Now you collected a set of elements C that is and if this the good event occurred, d is less than the median and u is greater than the median and this C itself is not too large, then what can we do in order to find the median?

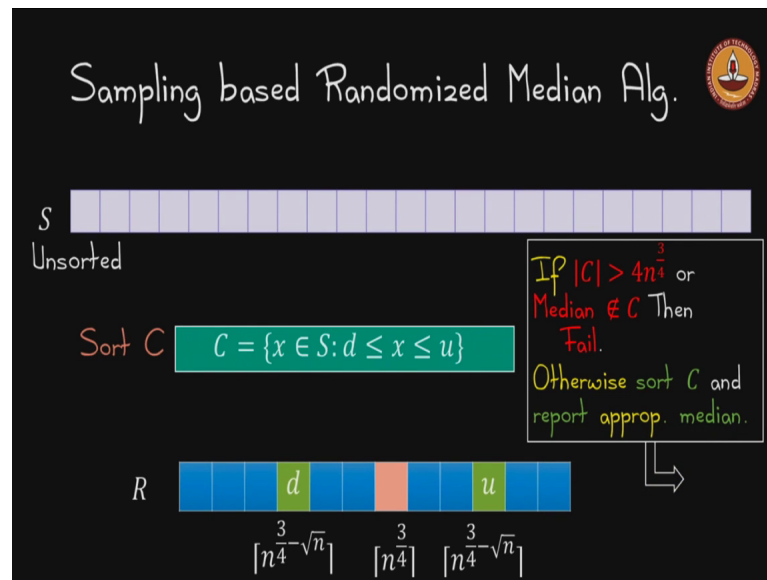
How do we find the median? Well let us lets it is not very difficult to see that for now important difference.

(Refer slide Time: 06:24)

The slide is titled "Sampling based Randomized Median Alg." and features a logo in the top right corner. It shows a horizontal array representing a set S . A segment of length ℓ is highlighted in green and labeled "Sorted $C = \{x \in S: d \leq x \leq u\}$ ". A purple arrow points from this segment to the text: "Sorted view (for intuition/analysis only)." and "The $\left(\frac{n}{2} - \ell + 1\right)$ th element in C is median in S ." A small video inset in the bottom right shows a man speaking.

So, I am taking S , I am just going to ask you to view S from the sorted viewpoint. Remember we do not have the sorted array S because, sorting this will require theta of $n \log n$ time ok, but just for thinking about how this problem this algorithm works, view it in the sorted case ok. Now, this C over here that you have. If you just sort it and this is let us say this is sufficiently small this c , this you take the sorted C and you superimpose it on S , where will the median lie? Remember what is the smallest element in C that is the element d , the largest element is u and remember we are at least hoping at the moment, that d is less than the median but close to the median, u is greater than the median, but close to the median. So, where will the median lie in this it is somewhere in this sorted in this C right?

(Refer slide Time: 07:48)

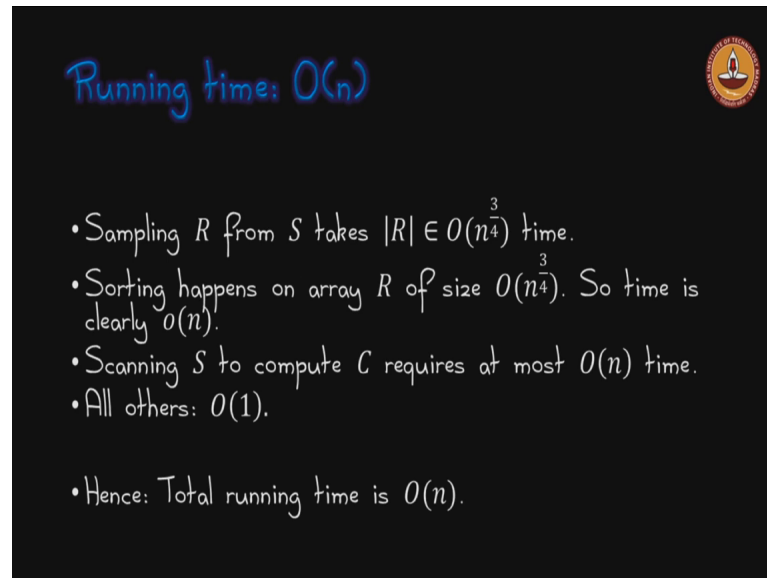


So, now what we know, that we know that that is the case, what do we do? We sort C and just a sanity check to we should be able to again scan and make sure that you know the number, if the number of elements in S that was there are less than d are is more than n over 2. What has happened? That that would have happened if this whole super imposed array is to the to the right of the median element. So, that is a bad event that was the case we have failed ok, similarly the u the number of elements greater than u should not be great also greater than n over 2, again that case also we have failed ok. Just need to make sure that the median is in fact, in C and we also have to make sure and this will happen with high probability that the size of the this set C is also sufficiently small if that is the case then we can sort C ah. If we can sort C then what we do; then we should be able to find the appropriate median element ok, so let us see how that works out ah.

So, again go back to this picture here , so we know that the median is somewhere in the middle of this sorted C that is being superimposed on S , what we can do is in our scan we can count the number of elements that were less than d ok. For simplicity if you are concerned let us make sure that we just think of the case where the elements are not repeated ok. So, the all the elements are not distinct. So, now what you do is count the number of elements to the there are less than d and now we want to spot the element in C , that is exactly the median element ok. So, that is the if you look the median element is going to be the n over 2 of the element.

So, the median element is going to be somewhere over here that is going to be the $\frac{n}{2}$ th element. So, if we want to spot the $\frac{n}{2}$ th element in S , we have to find the appropriate element in C , what is that going to be that is going to be? That going to be the $\frac{n}{2} - 1 + 1$ th element in C that will be the median element in S ok. So, that is it that gives us the ability to spot the median.

(Refer slide Time: 10:56)



Running time: $O(n)$

- Sampling R from S takes $|R| \in O(n^{\frac{3}{4}})$ time.
- Sorting happens on array R of size $O(n^{\frac{3}{4}})$. So time is clearly $O(n)$.
- Scanning S to compute C requires at most $O(n)$ time.
- All others: $O(1)$.

• Hence: Total running time is $O(n)$.

So let us just make sure that this is you know everything is correct at least from the running time point of view. The sampling of R from S takes O of n to the 3 fourths time because we are just assuming that these are random numbers that we are generating in the range 1 to n and collecting them into this array R ok

And sorting of R is going to take again little O of n time because it is going to take n to the 3 fourths times $\log n$ ok.

Student: What is the (Refer Time: 11:34) $n^{\frac{3}{4}}$ by 4.

Ah that that is a very good question, a lot of times what you do is you work with an idea. So, this is a this will give me an opportunity to explain how randomized algorithms works you have an intuitions to how the algorithm works and then you have to work through it and see then play with it and basically the worse engineer the parameters. You just need to find the right 1 set of right parameters for which the argument goes through.

Student: (Refer Time: 12:05).

You do not have to it does not have to be empirically, you can also in this case what you have to do is actually kind of try 1 set of values and work through the analysis, at the end if the fits required a proof then you are done, but if it does not, you can actually work your way backwards in the analysis and say oh at this point you know the probability is not good enough. So, I need to you know bring down the probability a little bit which means then working backwards in the analysis which means I should have sampled a little bit more over here. So, increase the sample a little bit over here. So, it is a reverse engineering process. So, you kind of have to work your way backwards and get and finally, play with the values until you get a 1 set of parameters for which the entire an algorithm and analysis go through.

So.

Student: (Refer Time: 12:59) say it is 1 minus some C between 0.

Yes that is a good point. So, you can keep things general, but keeping things general sometimes works sometimes ah.

Student: Ya it will become messy.

It will become very messy. So, what I end up doing and this is a personal thing is that I just put in specific values. So, it is a little bit also putting in specific values means that the intuition is not lost. If his if it is lot of parameters floating around the intuition can get lost. So, if you put specific values, then you still have this intuition and you kind of are playing with your intuition until you get some values. So, one thing you can do is after you play with your intuition get one set of values then you can generalize it and optimize the constants also if you want.

Student: Because you take lot of values between 0 and 1, let say you have to you have to prove that it is order of n right. So, intuitively you know that this power has to be between 0 and 1.

Correct ya.

Student: If I run a vigorous experiment will not be that exactly 3 by 4, I might get a special output.

Ya you could get. So, for example, you can even make this as long as the size of R is O of n over $\log n$ you are fine because you are doing a sorting and when you do the sorting you do not want to take seed O of n that is the that is important thing, but these are just one set of values for which it works, and when we have that we just kind of satisfied ok.

So sorting the array R can be sorted in little O of n time because it is of size O of n to the 3 fourths ah. So, from there we need to compute this set C where the elements lie between d and u that require one full scan of S right, but that is again still O of n all other operations can check in conditions here and there.

Student: sorting again right.

Ah yes. So, there will be one more sort, but again on oh ya the C will also be sorted, but C again is a size you are right. So, C again is of size at most n to the 3 fourths because it C is the subset of R oh sorry no I take it back. So, C is not a subset of R , but this is this is.

Student: (Refer Time: 15:22) bounded by.

Correct. So, that you are right. So, C is what we need to prove is that d and u are actually very close to the median, and what is that if they are very close to the median the number of elements.

Student: (Refer Time: 15:34).

Ya the number of elements that get dropped into C will also be little over and so, I have to add that 1 point over here you are right. So, thanks for pointing that out.

Student: (Refer Time: 15:43).

Ya.

Student: (Refer Time: 15:44) 1 before.

Ya 1 before

Student: I will ask him here we seeing that C is greater than.

Ya

Student: (Refer Time: 15:55).

Ya this is this is where this shows appropriate you are absolutely right. So, here what we need to show is with high probability C is going to be small and were how will C be small? If the d and u are close to the median, but still to the left and to the right of the median. If d and u are far away from the median then in the set of elements that you are collecting in C can become large right and you what we will do is, we prove in the next segment will prove that C is actually going to be less than or equal to $4n^{3/4}$ with high probability ok.

In the unfortunate event that it exceeds it is its a it is a bad event, actually you can this condition is a little too strong. As long as C is little o of or O of n over $\log n$ itself we can actually proceed because we can still afford a sorting on that set of elements.

Student: Is there any (Refer Time: 16:57).

Oh god I want you to work for your points.

So, all right. So, let us let us continue that and so, then now when you add up all the running times it is at most O of n .

(Refer slide Time: 17:26)

The slide features a yellow title box at the top with the text "Concluding Remarks". Below the title, there are three bullet points written in white on a black background. The first bullet point states: "Described an $O(n)$ time sampling based algorithm to find the median." The second bullet point states: "Analysis pending." The third bullet point states: "Output is either 'fail' or 'correct'" followed by "Claim: $\Pr(\text{fail}) \leq n^{-1/4}$ " with handwritten annotations n^{-1} and $n^{1/4}$ next to the exponent. Below the text, there is a small video inset showing a man with glasses speaking in front of a green chalkboard.

ah And ya as I said there are pending things here, what we know what we have seen is the algorithm hopefully the algorithm the idea is clear it is a very simple algorithm, there

are analysis issues that are pending though. For example, we need to ensure that I mean what is clear is that it is only going to fail or we are going to be correct when you, but and when it fails we know that it is failed ok. What we need to be able to show is that the probability of failure that is how can it fail, there it can fail for a few reasons as was pointed out it can fail because the set C the array C somehow is too large ok.

Student: I could mean that I would not fail do you just say that running time is larger than O of n

That is correct ya ah.

Student: So.

But if it is really large, then you are not no longer going to be in the O of n region.

Student: Ya so, but it is are you saying that if it exceeds that size.

Hm.

Student: Then the median that is finds will not close to the real median I think.

No the other failure is more egregious, the other failure is where the median. So, there are 2 conditions.

Student: Ya

Which can fail one is just a running time issue.

Student: Ya.

The other the median not being in C is the real.

Student: (Refer Time: 18:43) Monte Carlo.

Ya. So, currently the wait state of the algorithm is a Monte Carlo algorithm, meaning that it can fail to even produce a correct answer, but we can work with that to convert we can convert it into a Las Vegas algorithm I will discuss that shortly now.

Student: But currently also it is its having a problem with realistic running time.

Yes ya.

Student: Other one is the median is exactly median of the co or close to them.

No it is exactly the median, if the algorithm is in this conditioning is and we actually proceed here and basically median is in the C then we are going to find the exact median ok.

So, ah. So, we know that the output is either fail or correct. So, what will prove and this is I am going to leave it as a claim for now is that the probability of failure is at most $n^{-1/4}$ to the 1 fourth and. So, $n^{-1/4}$ ah. So, this is just as fixed constant in the exponent $n^{-1/4}$ and if you recall in the previous cases, when we wanted to prove something with high probability the bad event has $1/n$ probability ah. So, how do you I mean. So, this is just $n^{-1/4}$ would that be an issue what can we do to avoid this?

Student: (Refer Time: 20:21).

Ah we want to avoid this $n^{-1/4}$, which seems a little awkward we want to say let us say we want to be able to get this to n^{-1} what do we do? We repeat it to simply 4 times, and if it fails all 4 times then let us say we give up and that the 4 repetitions will ensure that it is n^{-1} the probability of failure ah. But in general for any $n^{-\alpha}$ you just have to repeat this sum what $4/\alpha$ times or something like that right and you will be able to ensure that you get $n^{-\alpha}$, you will be able to get that particular probability of failure

(Refer slide Time: 21:12)



So, with that we conclude this segment, hopefully the algorithm is very clear the analysis will require us to work with chebyshevs inequality. So, exercises on that which we will see in the next segment.