Binary Search · Binary Trees · Dynamic Programming (DP) · +2

Why and how does the O(n log n) longest increasing subsequence algorithm using a binary search work?

This question previously had details. They are now in a comment.

Answer · Follow · 28 · Request · 4+

Add a comment... Comment

Elynn Lee
Jan 5, 2015
You specifically want it with a binary search *tree*, not just binary search?
Reply · Upvote · Downvote · Report

Wasim Thabraze: Sorry! I meant Binary Search itself. I just wanted to know the f...

Raziman T.V.
Jan 15, 2015
Which part is confusing you?
Reply · Upvote · Downvote · Report

Quora Question Details Bot
Aug 8, 2017
I have gone through few tutorials. I'm feeling hard to understand the algorithm. Can someone describe in brief to me what actually happens in the O(n log n) algorithm? The Wikipedia article is cumbersome to understand.

Thanks in advance.
Reply · Upvote · Downvote · Report

Ravi Reddy
Jan 9, 2015
Could you share the wikipedia link?
Reply · Upvote · Downvote · Report

Wasim Thabraze
Jan 9, 2015
Longest increasing subsequence
Reply · Upvote · Downvote · Report

Ad by datascience@berkeley
Earn a Master's in Data Science online from UC Berkeley.
Learn to analyze complex data with UC Berkeley's Master's in Data Science online. GRE/GMAT required.
Learn more at datascience.berkeley.edu

6 Answers

Eugene Yarovoi, Tech Interviews and Competitive Programming Meetup organizer
Updated Dec 28, 2017

The starting idea for the algorithm is that we will scan the input from left to right, maintaining at any given time a representation of all the possible increasing subsequences that can be formed with the elements seen so far, *such that* the sub-sequences we track are ones that could contribute to the final solution. We will aggressively discard subsequences that can never influence the solution, and this will allow the algorithm to become efficient.

What subsequences can contribute to the solution? First, we note that if we have two subsequences of equal length, say,

3 -> 9 -> 11 -> 13

and

Related Questions

- How can you count all the duplicates in a binary search tree in O(n) time and O(1) space?
- How can we find the longest continuous subsequence of 0's in binary representation of an integer in O(log n) time complexity?
- Can you prove $O(\log(n!)) = O(n \log n)$?
- How do you compute longest increasing subsequence?
- What's the expected length of the longest increasing subsequence?
- How do top n search algorithms work?
- Other than binary search, are there other code examples that have a running time of O(log n)?
- Given an array of N integers, how can we find the number of all increasing subsequences in O(N*Log N) time?

+ Ask New Question

More Related Questions

Question Stats

27 Public Followers
8,798 Views
Last Asked May 10, 2016
Edits

4 -> 6 -> 10 -> 15

Then, it only makes sense to track the one that has the smaller ending number (13 in this case). This is because if we ever extend either of these sequences with another number, even if we validly extended the second one, for example if we see the number 17 and do:

4 -> 6 -> 10 -> 15 -> 17,

we could have also extended the first one:

3 -> 9 -> 11 -> 13 -> 17

So, we lose nothing by forgetting all about the sequence ending with 15: as a general principle, the sequence ending with the smallest number can always replace the sequence ending with the larger number.

(However, the sequence ending with the smaller number might be able to be extended with numbers that the other can't be extended with, such as 14 in this example, so the reverse argument does not work.)

As such, we now know it's safe to maintain only one sequence per distinct sequence length: the one that ends in the smallest number.

Already, this paves the way to an $O(N^2)$ algorithm. As you process each new number in the input (let N be how many numbers are in the input), iterate through all sequences you are maintaining (at most N, one per distinct sequence length), and see to which sequences you can append that number. For each sequence where you can append the number, generate a new candidate sequence with the append. After generating the new candidate sequences, compare each one with the (at most one) other sequence you had of the same length, and keep only the one ending with the smaller number.

(To make sequences of the same length efficient to find, it's best to have an array of pointers to sequences, with arr[k] representing the current best sequence of length k.)

Implemented sloppily, the above might be $O(N^3)$ due to copying of the sequences, but a sequence can be represented as simply a single number plus a pointer to the previous sequence it extends. Implemented this way, and because all the sequence comparisons look at only the last number in each sequence, we reach $O(N^2)$ time complexity (O(N) work, to look at the last number in each of up to N sequences, for each of the N numbers in the input).

To improve it further, we make another observation, which allows us to discard even more sequences!

We note that if we have two sequences of different length, and the longer one ends with a smaller or equal number than the shorter one, the shorter one can be discarded. If you have:

3 -> 9 -> 11 -> 12 -> 13

and

4 -> 6 -> 10 -> 15

You would never use the second one, by the same kind of reasoning as before. Importantly, this is true even when sequences end in the **same** number: 3 -> 9 -> 11 -> 12 -> 15 allows for 4 -> 6 -> 10 -> 15 to be discarded.

Related Questions

How can you count all the duplicates in a binary search tree in O(n) time and O(1) space?

How can we find the longest continuous subsequence of 0's in binary representation of an integer in O(log n) time complexity?

Can you prove $O(\log(n!)) = O(n \log n)$?

How do you compute longest increasing subsequence?

What's the expected length of the longest increasing subsequence?

How do top n search algorithms work?

Other than binary search, are there other code examples that have a running time of O(log n)?

Given an array of N integers, how can we find the number of all increasing subsequences in O(N*Log N) time?

+ Ask New Question

This means that now, since we will commit to discarding shorter sequences that exhibit the above pattern, we also have the guarantee that shorter sequences that we're tracking end with smaller numbers than longer sequences.

That means that, if we have an array where arr[k] refers to the best sequence of length k, when we look at the ending numbers of the sequences in this array, we see something that looks like this (for some arbitrary example):

```
sequence ending num: unset, 2, 5, 10, 16, 26, unset, unset
array index:             0, 1, 2,  3,  4,  5,     6,     7
```

The important feature here is that the sequence end numbers are in ascending order.

Now, let's say we read the next number from the input and it is 12. Which sequences can we append it to? Everything 10 and below. But, appending it to anything but 10 is useless, because all the sequences after the append will end in 12, and be discarded unless they are the longest sequence ending in 12. So, we need only append to the longest sequence to which we are able to append, 10 in this case.

After appending, we have a new candidate sequence ending in 12 whose length is 4. This sequence has a smaller or equal ending number than the previous best sequence of that length, because otherwise we would have extended *that* sequence. In this case, 12 is better than 16. We do the update and the new state becomes:

```
sequence ending num: unset, 2, 5, 10, 12, 26, unset, unset
array index:             0, 1, 2,  3,  4,  5,     6,     7
```

The array is really maintaining a pointer to the sequence object and not just the ending number so that you can recover the sequence afterwards.

The only remaining question is how, given the number 12, we can find the position of 10, in other words, the rightmost index whose value is less than the target value of 12. Remember the sequence ending numbers are in ascending order, so we can binary search. This allows each number of the input to be processed in O(log N), which results in a complexity of O(N log N).

1.9k Views · View Upvoters · View Sharers

Upvote · 13 Share · 1

Add a comment... Recommended All

Promoted by Coding Dojo

Learn to program in LA. Become a developer in 14 weeks.

No experience required. Part time courses and scholarships available. Average starting salary is $76k.

Learn more at codingdojo.com

Varun Mahajan, BTech Computer Science, SRM Institute of Science and Technology (2019)
Answered Jul 28

best explanation here

http://lightoj.com/article_show....

Let the sequence be
Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
There are 10 elements, so, I will contain 11 elements with index from 0 to 10. So, as described, the elements of I[] will be initially,

Related Questions

How can you count all the duplicates in a binary search tree in O(n) time and O(1) space?

How can we find the longest continuous subsequence of 0's in binary representation of an integer in O(log n) time complexity?

Can you prove $O(\log(n!)) = O(n \log n)$?

How do you compute longest increasing subsequence?

What's the expected length of the longest increasing subsequence?

How do top n search algorithms work?

Other than binary search, are there other code examples that have a running time of O(log n)?

Given an array of N integers, how can we find the number of all increasing subsequences in O(N*Log N) time?

+ Ask New Question

I [-i, i, i, i, i, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [n, n, n, n, n, n, n, n, n, n]

Here i denotes 'infinite' and n denotes 'not calculated yet'.

Now let's insert the numbers from sequence into I[].
1) At first we have 8, so the position for 8 in I[] is 1. Since the left item is –i which is smaller than 8. So, we will assign I[1] = 8. Since 8 is inserted in 1st place, so the L[] value of 8 will be 1. After the first number we will get

I [-i, 8, i, i, i, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [1, n, n, n, n, n, n, n, n, n]

2) Now we have 1, the position for 1 in I[] is 1. So, we will assign I[1] = 1 and since 1 is inserted in 1st place, the L[] value of 1 will be 1. Observe that I[1] was 8, but after this iteration 8 will be replaced by 1.
I [-i, 1, i, i, i, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [1, 1, n, n, n, n, n, n, n, n]

3) The next number is 9. The position for 9 in I[] is definitely the 2nd position. So, we will assign I[2] = 9, and since 9 is inserted in the second position, so the L[] value of 9 will be 2.

I [-i,1,9, i, i, i, i, i, i, i, i]

index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]

L [1, 1, 2, n, n, n, n, n, n, n]

4) Now we have 8, and the position of this 8 in I[] is the second position. So, we put 8 in the second position of I[] (thus replacing 9). And the L[] value of 8 will be 2.

I [-i, 1, 8, i, i, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [1, 1, 2, 2, n, n, n, n, n, n]

5) The next item is 3, and the position in I[] is still the second position. So, we place 3 in the second position of I[]. And the L[] value of 3 is 2.

I [-i, 1, 3, i, i, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [1, 1, 2, 2, 2, n, n, n, n, n]

6) Now we have 4, and the position in I[] is 3. So, the L[] value of 4 is 3.

I [-i, 1, 3, 4, i, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Related Questions

- How can you count all the duplicates in a binary search tree in O(n) time and O(1) space?
- How can we find the longest continuous subsequence of 0's in binary representation of an integer in O(log n) time complexity?
- Can you prove $O(\log(n!)) = O(n \log n)$?
- How do you compute longest increasing subsequence?
- What's the expected length of the longest increasing subsequence?
- How do top n search algorithms work?
- Other than binary search, are there other code examples that have a running time of O(log n)?
- Given an array of N integers, how can we find the number of all increasing subsequences in O(N*Log N) time?

+ Ask New Question

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [1, 1, 2, 2, 2, 3, n, n, n, n]

7) Next item is 6, it will be inserted in the 4th position of I[]. So, the L[] value of 6 will be 4.

I [-i, 1, 3, 4, 6, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [1, 1, 2, 2, 2, 3, 4, n, n, n]

8) Next item is 1, it will be inserted in the 1st position of I[]. So, the L[] value of 1 will be 1.

I [-i, 1, 3, 4, 6, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [1, 1, 2, 2, 2, 3, 4, 1, n, n]

9) Next item is 5, it will be inserted in the 4th position of I[]. So, the L[] value of 5 will be 4.

I [-i, 1, 3, 4, 5, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [1, 1, 2, 2, 2, 3, 4, 1, 4, n]

10) Last item is 2, it will be inserted in the 2nd position of I[]. So, the L[] value of 2 will be 2.

I [-i, 1, 2, 4, 5, i, i, i, i, i, i]
index 0 1 2 3 4 5 6 7 8 9 10

Sequence [8, 1, 9, 8, 3, 4, 6, 1, 5, 2]
L [1, 1, 2, 2, 2, 3, 4, 1, 4, 2]

Now, see that throughout the process the values in I[] are in ascending order. And since we place a number in such a place where all the numbers in left are strictly less than the number so, the L[] value will be correct for all the numbers. One more thing, after all the iterations are over, don't think that I[] will also contain a sequence. Because in the example above, see that 1, 2, 4, 5 is not the correct sequence. See pdf for reconstruction method
Now since the values in I[] will always be in ascending order, so, to find a position of a number we can definitely use binary search. So, if the final LIS length is k and there are n numbers then to insert a number using binary search the complexity will be O(logk). Thus the overall complexity will be O(nlogk), since we are inserting n items.

217 Views

Upvote Share

Add a comment... Recommended All

Related Questions

How can you count all the duplicates in a binary search tree in O(n) time and O(1) space?

How can we find the longest continuous subsequence of 0's in binary representation of an integer in O(log n) time complexity?

Can you prove $O(\log(n!)) = O(n \log n)$?

How do you compute longest increasing subsequence?

What's the expected length of the longest increasing subsequence?

How do top n search algorithms work?

Other than binary search, are there other code examples that have a running time of O(log n)?

Given an array of N integers, how can we find the number of all increasing subsequences in O(N*Log N) time?

+ Ask New Question

Promoted by Hired

Hired. Your answer to job searching.

Join the career matching site built just for software engineers so you can leave the job searching to us.

Apply now at hired.com

Chandan Mittal, Algorithms are my hobby
Answered Jan 11, 2015

Longest Monotonically Increasing Subsequence Size (N log N) - GeeksforGeeks

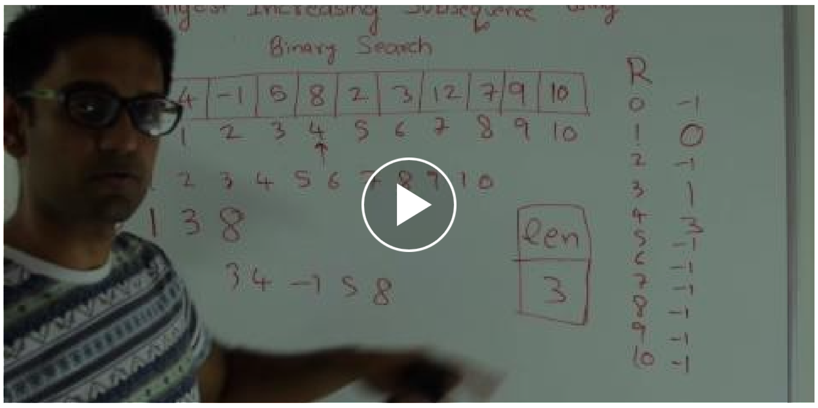
The O(NlogN) calculation of LIS is effectively explained in the tutorial referred by the above link. This one's possibly the best article available on Internet that suffices your query.
I suggest you read the whole article till the end.
PS. If you have any doubt, you can comment reagarding the same below.

5.1k Views · View Upvoters

Upvote · 2 Share

Add a comment... Recommended All

Akhilesh Joshi, studied Computer Science & Data Science at The University of Texas at Dallas (2018)
Answered Mar 24, 2017



best explanation to algo with perfect illustration.. Happy Learning !

926 Views · View Upvoters

Upvote · 1 Share

Add a comment... Recommended All

Ashit Panda(ପଣ୍ଡା), B-Trees are not Binary tree.
Answered Mar 5, 2017

To A2A your question,i will just suggest you to refer this Longest Increasing Subsequence Size (N log N) - GeeksforGeeks for your better understanding purpose.I came across this post few days back because it had not been posted back then.

1.3k Views · View Upvoters · Answer requested by Alokik Pathak

Upvote · 1 Share

Add a comment... Recommended All

Gopal Das, Principal Consultant at Jardine Lloyd Thompson Group
Updated Jan 20

Related Questions

How can you count all the duplicates in a binary search tree in O(n) time and O(1) space?

How can we find the longest continuous subsequence of 0's in binary representation of an integer in O(log n) time complexity?

Can you prove $O(\log(n!)) = O(n \log n)$?

How do you compute longest increasing subsequence?

What's the expected length of the longest increasing subsequence?

How do top n search algorithms work?

Other than binary search, are there other code examples that have a running time of O(log n)?

Given an array of N integers, how can we find the number of all increasing subsequences in O(N*Log N) time?

+ Ask New Question

Please see my post Solution – Scoring Weight Loss

627 Views

Upvote Share

Add a comment...

Recommended All

3 Answers Collapsed (Why?)

Related Questions

How can you count all the duplicates in a binary search tree in O(n) time and O(1) space?

How can we find the longest continuous subsequence of 0's in binary representation of an integer in O(log n) time complexity?

Can you prove $O(\log(n!)) = O(n \log n)$?

How do you compute longest increasing

Top Stories from Your Feed

How do top n search algorithms work?

Other than binary search, are there other code examples that have a running time of O(log n)?

Given an array of N integers, how can we find the number of all increasing subsequences in O(N*Log N) time?

+ Ask New Question