Mathematics and Algorithms Algorithms

Why does 4 * N space have to be allocated for a segment tree, where N is the size of the original array?

Answer Follow · 6 Request 1

4 Answers

Brian Bi, in love with algorithms

Updated Oct 23, 2018

Let $S(n)$ denote the size of the array needed to build a segment tree on an interval of size n. I assume that if n is odd, we put the middle element in the left subtree.

There is an easy upper bound: $S(n) \leq 4n$. To derive this upper bound, observe that the height of the tree is $\lceil \log_2 n \rceil$, and the rightmost node on the level of depth h is located at position $2^{h+1} - 1$ in the array. Therefore

$$\begin{aligned} S(n) &\leq 2^{\lceil \log_2 n \rceil + 1} - 1 \\ &< 2 \cdot 2^{\lceil \log_2 n \rceil} \\ &= 4 \cdot 2^{\lceil \log_2 n \rceil - 1} \\ &\leq 4 \cdot 2^{\lfloor \log_2 n \rfloor} \\ &\leq 4n \end{aligned}$$

You might wonder whether you really *need* $4n$ space though, or whether kn would suffice for some $k < 4$. At one point I made the foolish assumption that $k = 2$ would suffice, and this was embarrassingly recently. (Exercise for the reader: find a counterexample.)

To explore further, we should derive an explicit formula for $S(n)$:

$$S(n) = \max\left\{S(\lceil n/2 \rceil) + 2^{\lceil \log_2 \lceil n/2 \rceil \rceil}, S(\lfloor n/2 \rfloor) + 2^{\lceil \log_2 \lfloor n/2 \rfloor \rceil + 1}\right\}$$

(with the base case $S(1) = 1$.) This formula might look scary, but it's just saying that the largest position in the array occupied by a tree on n nodes will either be in the left subtree or in the right subtree; for the left subtree we inherit the structure from $\lceil n/2 \rceil$ since we put the middle element, if any, on the left; for the right subtree we inherit the structure from $\lfloor n/2 \rfloor$, and in both cases we add the power of 2 to account for the fact that the left and right children of the root are not actually at the root (and all their children are one level deeper).

We can implement this formula in Mathematica and plot $S(n)/n$:

```
S[1] := 1;
S[n_] := Max[S[Ceiling[n/2]] + 2^Ceiling[Log2[Ceiling[n/2]]],
             S[Floor[n/2]] + 2*2^Ceiling[Log2[Floor[n/2]]]];
```

and plot it:

136 4

Related questions

Why does a segment tree need a 4n-size array? Why not 2n-1?

Generally, we know 4*N space should be declared in segment tree but if N is 10^9, it...

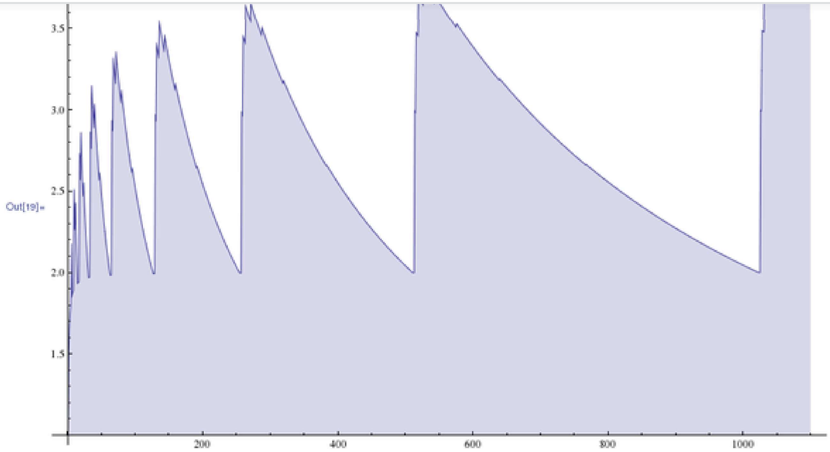
What should be the maximum size of segment tree built on an array on 100,000 elements?

Why is the limit $(1 - \frac{1}{n})^n$ equal to $\frac{1}{e}$?

How do we prove that the time complexity of the update and query operations in a segmen...

In a segment tree, if the length of an array is 5, then the number of nodes in the tree should...

Add question



From this plot it is natural to make the following conjecture:

Conjecture: $\limsup_{n\to\infty} \frac{S(n)}{n} = 4$

so 4 is, in fact, the smallest k such that it's safe to assume $S(n) \leq kn$!

If you're a competitive programmer, you probably don't care about the formal proof, because this plot probably convinces you of the conjecture (so you should just use $4n$). If you want the real proof, read on.

For the proof we will need the following Lemma:

Lemma: Let m be a positive integer. Then

$$S(2^m) = 2^{m+1} - 1$$

Furthermore, for all nonnegative integers $k < n$,

$$S(2^m + 2^k) = \frac{2^{k+1}-1}{2^k} 2^{m+1} + 1$$

Proof: By induction on m.

For the base case, take $m = 1$. Then using $S(2) = 3, S(3) = 5$, we can explicitly the claim for both $S(2^m)$ and $S(2^m + 2^k)$ for $k = 0$ (the only possible value).

For the inductive case, let $m \geq 2$ be given, and assume that for all smaller m, the claim is true for all possible k. Now there are two cases.

Case 1: $k = 0$. Then, using the recursive formula for $S(2^m + 1)$ and the inductive hypothesis for $S(2^{m-1} + 1)$ and $S(2^{m-1})$:

$$\begin{aligned}
S(2^m + 2^k) &= S(2^m + 1) \\
&= \max(S(2^{m-1} + 1) + 2^m, S(2^{m-1}) + 2^m) \\
&= \max(2^m + 1 + 2^m, 2^m - 1 + 2^m) \\
&= 2^{m+1} + 1 \\
&= \frac{2^1 - 1}{2^0} 2^{m+1} + 1
\end{aligned}$$

Case 2: $k > 0$. This is pretty similar:

$$\begin{aligned}
S(2^m + 2^k) &= \max(S(2^{m-1} + 2^{k-1}) + 2^m, S(2^{m-1} + 2^{k-1}) + 2^{m+1}) \\
&= S(2^{m-1} + 2^{k-1}) + 2^{m+1} \\
&= \frac{2^k - 1}{2^{k-1}} 2^m + 2^{m+1} + 1 \\
&= \left(1 + \frac{2^k - 1}{2^k}\right) 2^{m+1} + 1 \\
&= \frac{2^{k+1} - 1}{2^k} 2^{m+1} + 1
\end{aligned}$$

So we've already proven the upper bound, all that remains to prove is that $S(n)/n$ attains values arbitrarily close to 4, for arbitrarily large values of n. To do so, we can try to find the value of k that maximizes $S(2^m+2^k)/(2^m+2^k)$ for given m, which is a simple exercise in calculus and should yield $k \approx m/2$. So we can construct our examples by putting $n = 2^{2j}+2^j$ for $j = 1, 2, 3, \ldots$. Then

$$\begin{aligned}
\frac{S(n)}{n} &= \frac{\left(2^{j+1}-1\right)2^{-j}2^{2j+1}+1}{2^{2j}+2^j} \\
&= 4\frac{\left(2^{j+1}-1\right)2^{-(j+1)}2^{2j}+1/4}{2^{2j}+2^j} \\
&= 4\left[1-\frac{2^{2j}+2^j-\left(2^{j+1}-1\right)2^{-(j+1)}2^{2j}-1/4}{2^{2j}+2^j}\right] \\
&= 4\left[1-\frac{2^{-(j+1)}2^{2j}+2^j-1/4}{2^{2j}+2^j}\right] \\
&= 4\left[1-\frac{3\cdot 2^{j-1}-1/4}{2^{2j}+2^j}\right] \\
&> 4\left[1-\frac{3\cdot 2^{j-1}}{2^{2j}}\right] \\
&= 4(1-3\cdot 2^{-(j+1)})
\end{aligned}$$

It's easy to see that the last line approaches 4 as j goes to infinity. This completes the proof.

For example, if we take $j = 5$, we get $n = 2^{10}+2^5 = 1056$, for which $S(n)/n \approx 3.819$.

15.3K views · View 136 upvotes

4 comments from Manashi Sarkar and more

44 9

Updated Jul 28, 2020

2020 update:

NEW - Short answer - I have deleted a long :) old answer.

NOTE: You will have to re-read multiple times to understand all parts of this short answer.

—

Let's see visually:

Array vs it's Segment tree (a is array, st is segment tree)

Here segment tree does SUM of elements, so it is useful to use in RSQ (Range sum query)

—

a = [1];

—

a = [1, 2];

st [3,1,2]

.........3.........

.....1...2........ <——————————— n (2) is power of 2, 2*n-1 is st size

—

a = [1, 2, 3];

st [6,3,3,1,2,".",".","."]

.... 6..........

......3........**3.....** <——— n (3) is Not power of 2

..1...2..........<——— st size got doubled (one more line means double eles)

—

a = [1, 2, 3, 4];

st [10,3,7,1,2,3,4]

.............10...................

......3..............7...........

...1...2..... 3...4...... <——————————— n (4) is power of 2, 2*n-1 is st size

—

a = [1, 2, 3, 4, 5];

st [15,6,9,3,3,4,5,1,2,".",".",".",".",".",".",".",".","."]

.............15.................

.....6...............9..........

...3...**3.......4...5.....** <——— n (5) is Not power of 2

1..2..................... <——— st size got doubled (one more line means double eles)

—

From above, below can derived:

1. if N is power of 2, then segment tree needs 2*N -1
   a. so if array a, is 4
   b. then st will have 2*4–1 = 7 elements
      i. 4 (N) are leaf nodes
      ii. 3 (N-1) are internal nodes
2. if we increase only 1 element in a from 4 to 5 then what happens?
   a. st gets doubled. see point 3 below
3. if N is Not power of 2, then we need to go to next power of 2 (if 5, then 8), say 8s is 2 pow 3, say 3 x, then we need 2*x -1
   a. Going to next power of 2 means we doubled the size (multiplied by 2)
   b. But we only increased 1 element in array from 4 to 5, so it is almost 4 times (because it st was already 2 times when array was 4 elements)

iii. 2(N-1) - 2 elements are unused

iv. Approx 4N elements when array size is not 2 power

That’s all.

Task: Now draw segment tree (st) for below 3 ‘a’s:

- [1, 2, 3, 4, 5, 6]
- [1, 2, 3, 4, 5, 6, 7]
- [1, 2, 3, 4, 5, 6, 7, 8]

Cross check with above diagrams.

Hope that helped. Best of luck.

The case with segment trees is, we usually write then as full no matter if N is power of 2 or not. Consider a tree with 5 leafs: leaf 5 needs a parent - this is a node (A) which children would naturally be 5 and 6 and that node needs a parent (B) which children would naturally be [5, 6] and [7,8], and so on. Now, you could decide to implement the tree in a way that A and B have one child each, however this poses two problems:
- the code gets more complicated because every time you visit each node you have to check its number to determine if there are two children to consider or one.
- there is no node right of A ([5,6]) - but all the nodes from this level expect their children having indexes 2*parent and 2*parent + 1. It's no longer true if node [7, 8] doesn't appear (we could've left an empty space instead of constructing the node, but then the array must be as large as if the node was there).

Thus, the most rational solution is to decide that each segment tree is full - if we have 5 nodes we allocate the tree for 8 nodes and make the nodes 6, 7, 8 "empty", i.e. behaving so that they don't change the behaviour of the queries (e.g. if the tree searches minimum on the segment, those nodes will have the "INF" value).

Related questions

More answers below

Why does a segment tree need a 4n-size array? Why not 2n-1?

Generally, we know 4*N space should be declared in segment tree but if N is 10^9, it gives a compile error. What to do?

What should be the maximum size of segment tree built on an array on 100,000

Try to put all the array elements in leaf nodes of a complete binary tree and add dummy elements in remaining leaf nodes. Now calculate the full size of the tree like 1 + 2 + 4 + 8 + 16 ... for incremental levels. This should give the most optimal space allocation for a segment tree. 4n allocates more space according to me.

```
private int getSegmentTreeSize(int inputCount) {
    int index = 1, segmentTreeSize = 0, powerOf2 = 0;
    while(powerOf2 < inputCount) {
        powerOf2 = (int)Math.pow(2, index++);
        segmentTreeSize += powerOf2;
    }
    return segmentTreeSize;
}
```


Related questions

Why does a segment tree need a 4n-size array? Why not 2n-1?

Generally, we know 4*N space should be declared in segment tree but if N is 10^9, it gives a compile error. What to do?

What should be the maximum size of segment tree built on an array on 100,000 elements?

Why is the limit $(1-\frac{1}{n})^n$ equal to $\frac{1}{e}$?

How do we prove that the time complexity of the update and query operations in a segment tree are $\Theta(\lg n)$?

In a segment tree, if the length of an array is 5, then the number of nodes in the tree should be 8*2-1=16. Why is this and what does the tree look like?

Sphere Online Judge (SPOJ): How do I use a persistent segment tree to solve a problem?

∑ (n=1 to ∞) [1/n*(4^n)] what is the sum of the series?

What should be the size of segment tree for an array of size n?

What is the simplest implementation of segment trees that you have seen?

What is $\sqrt{n+\sqrt{n+\sqrt{n+\dots}}}$?

What's the difference between an array version of segment tree and a structure version of a segment tree?

How do I find the sum of the series k * 2^k from k=1 to 100?

Why do people use mid=low+ (high-low) /2 instead of (low+high) /2?

How do I find the sum of a series k*x^(2k)?