Assignment 1   CSCI 429   Solutions

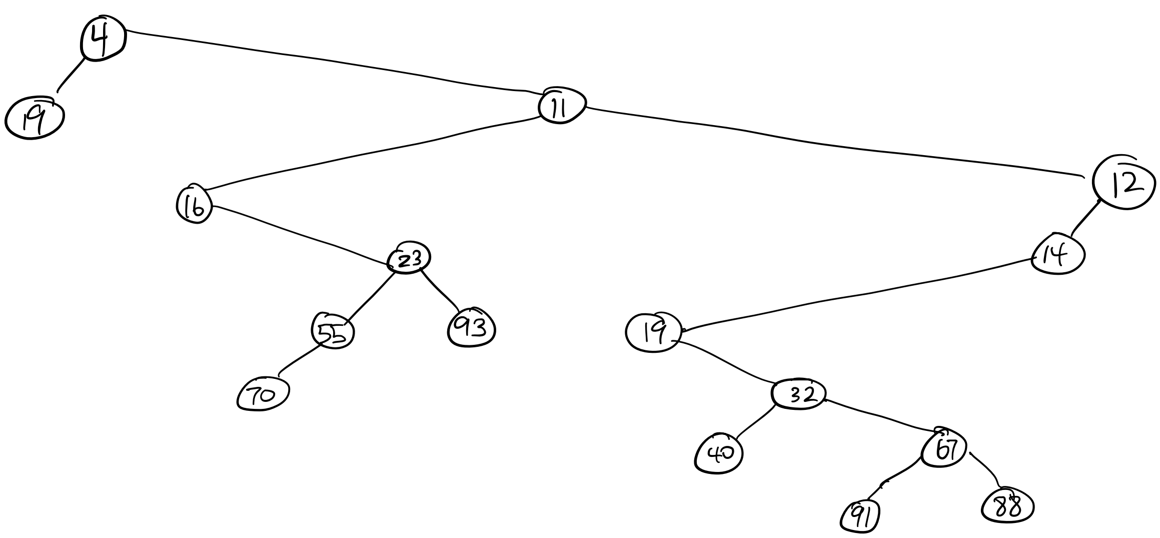
1. A = [0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 . . . . 6 7 8 9]

Sparse Table first 10 rows are

ℓ \ range size	2^0	2^1	4	8	16	32	64
0	0	0	0	0	0	0	0
1	1	1	1	1	10	10	10
2	2	2	2	2	10	10	10
3	3	3	3	10	10	10	10
4	4	4	4	10	10	10	10
5	5	5	5	10	10	10	10
6	6	6	6	10	10	10	10
7	7	7	10	10	10	10	10
8	8	8	10	10	10	10	10
9	9	10	10	10	10	10	10

2. Cartesian tree for

[19 4 16 70 55 23 93 11 19 40 32 91 67 88 14 12]



3. Write pseudocode for Cartesian tree construction, given array A

The question does not require the algorithm be in linear time, so the basic divide-and-conquer will do:

```
CartTree (array A[0..n-1] of int)
  T = CartTreeHelper (A, 0, n-1)
  return T

CartTreeHelper (A, l, r)   // creates a CartTree out of
                           A[l..r] and returns pointer
                           to that tree //
    if l > r return NULL
    if l == r return node (l, NULL, NULL)
    min = l
    for (int i = l+1; i ≤ r; i++)
        if A[i] < A[min]
            min = i

    left = CartTreeHelper (A, l, min-1)
    right = CartTreeHelper (A, min+1, r)
    return node (min, left, right)
```

4.1 CT to Bitstring (CartTree T)

```
// given a Cartesian Tree T, constructs an integer, which we
// interpret as a bitstring encoding the shape of a binary tree
b = 0        // b is a word of 0's.     Assumes our
if T == NULL   return b               // word size is sufficient
                                      // otherwise can string
queue BFS                             // words together or use
BFS.enqueue(T)                        // a vector of bool
while !BFS.empty()
    v = BFS.dequeue()
    if v == NULL     b = b << 1 + 1   // shifts a "1" onto b
    else
        BFS.enqueue(v -> left)
        BFS.enqueue(v -> right)
        b = b << 1          // shifts a "0" onto b

// finally, remove final 1, as it is not part of
// our encoding
b = b >> 1   // The bitstring encoding is the suffix of b that
             // has same number of 0's as 1's.
```

4.2. No, There is a bijection between the Cartesian Trees of size n and the bitstrings of length $2n$ that
- have #0's = #1's
- have the prefix property : no

prefix has #1's > #0's.

(as indicated in tutorial)

# Cart Trees of size n	$\leq$	# bitstrings with n 0's n 1's and prefix property	$<$	# bitstrings on $2n$ bits, if $n > 1$

Def^n: Bal_n = set of strings with n 0's, n 1's that have the prefix property.

Claim: $$|Bal_n| = \begin{cases} 1 & \text{if } n = 0 \\ \sum_{i=1}^{n} C_{i-1} \cdot C_{n-i} & \text{otherwise} \end{cases}$$ nth catalan number

Proof: By induction on n.

Basis: if $n=0$, $\exists$ just one string, ε, of ~~len~~ length 0 and $Bal_0 = \{\varepsilon\}$

Now let n be any integer > 0, and suppose that the claim holds for all $i < n$.

Consider any $s \in Bal_n$.

There must exist some smallest $i > 0$ such that the prefix $S[1..2i] \in Bal_i$

eg $\big((\,)(\,)(\,)\big)$ with $i=3$

$\big((\,)\big)\big((\,)(\,)(\,)\big)$ with $i=2$

Call i the "first cut" for the string.

We will sum up over all possible i, $1 \le i \le n$, the ways to build a string in B_n that has i as its first cut.

Fix i: $\Big(\underbrace{\sim\sim\sim\sim}_{\text{all possible } B_{i-1}}\Big)\underbrace{\Big(\qquad\qquad\Big)}_{\text{all possible } B_{n-i}}$

$$\therefore\ |B_n| = \sum_{i=1}^{n} C_{i-1} \cdot C_{n-i}. \qquad \blacksquare$$

4.3 BitsToCTree (b)

// b is a bit string which could be stored as a long
// unsigned integer

Reverse b so root is at least-sig-bit
of the representation // ∃ many ways to do this

```
new queue BFS = NULL

new treenode T = node(_, NULL, null)
t = &T
while b > 0

    if b is even   // least sig bit is 0
        t = node(_, NULL, NULL)
        BFS.enqueue(&t -> left)
        BFS.enqueue(&t -> right)

    else
        t = NULL

    b = b >> 1

    t = BFS.dequeue()

// need the extra 1 node to complete the tree

t = NULL.
```

5. MIT notes acknowledge that the values of the integers stored are in range $1 .. \lg n$, and such integers can be encoded in $\lg \lg n$ bits.

$\therefore$ the table size = # entries * size of entry encoding

$= \sqrt{n} \lg^2 n * \lg \lg n$

$= \sqrt{n} \lg^2 n \lg \lg n.$

6. We not that $1 \notin \Omega(n^{\varepsilon})$ for any $\varepsilon > 0$,
so $n^{\varepsilon} \in o(1) \ \forall \varepsilon > 0$.

Claim: $n^{1/4} \notin O(1)$ (ie, $1 \in o(n^{1/4})$)

Proof: BWOC. $\$ \ \exists \ n_0, c$ such that $n^{1/4} \leq c \cdot 1 \ \forall n \geq n_0$

$\Rightarrow n \leq c^4 \ \forall n \geq n_0$

But $n = 1 + \max(n_0, c^4)$ provides a contradiction. ▨

Claim: $\sqrt{n} \lg^2 n \lg \lg n \in o(n)$

Proof: ① $\sqrt{n} \in O(\sqrt{n})$ constant factors

② $\lg^2 n \in O(n^{\frac{1}{8}})$ Log Domination Rule

③ $\lg n \in O(n^{\frac{1}{8}})$ Log Domination

④ $\lg \lg n \in O(\frac{1}{8} \lg n)$ ③, taking logs

⑤ $\lg \lg n \in O(\lg n)$ ④, constant factors

⑥ $\lg \lg n \in O(n^{\frac{1}{8}})$ ⑤, ③, transitivity

⑦ $\sqrt{n} \lg^2 n \lg \lg n \in O(n^{6/8})$, ①, ②, ⑥ Product Rule

⑧ $1 \in o(n^{2/8})$ Separately Proved.

⑨ $\sqrt{n} \lg^2 n \lg \lg n \in o(n)$, ⑦, ⑨ product rule.

▨

7. MinContigSum (A [0 . . . . n-1])

Best Include = 0

Best Skip = 0

for i = 0 to n-1

BestInclude = min(0, BestInclude + A[i], A[i])

BestSkip = min (BestSkip, BestInclude)

return BestSkip

Why it works:

Induction. Before any iterations of the loop, i.e. i = -1, the empty subsequence is best contig sum so far, and its sum is 0.

If, after i iterations of the loop,

- BestSkip is overall min contig sum of A[0 .. i-1]
- Best Include is min contig sum that ends at A[i-1],

then after one more iteration the same is true - it is a "loop invariant"

∴ at end, BestSkip will be min contig sum.