

Overview

- Morris' counter
 - partially sorted array
 - average-case vs expected (hiring problem)
 - sorting lower bound
 - additional problems
-

Problems

Q1. Morris' probabilistic counting. With a deterministic b -bit counter, we can only count up to $2^b - 1$. With *probabilistic counting* we can count to *larger values* at the expense of *loss of precision*.

We let a counter reading of i represent a count of v_i , for $0 \leq i \leq 2^b - 1$. Initially the counter reads 0, indicating the count of $v_0 = 0$.

The operation *increment* works on a counter with reading i in a probabilistic manner:

- if $i < 2^b - 1$, increase counter reading with probability

$$\frac{1}{v_{i+1} - v_i},$$

and leave the counter unchanged otherwise.

- if $i = 2^b - 1$, report overflow.

Note that if we select $v_i = 1$, then the counter is an ordinary deterministic counter. More interesting situations arise if $v_i = 100i$, $v_i = 2^i$, or $v_i = i$ -th Fibonacci number.

Assume that the probability of an overflow is negligible. Show that the value represented by the counter after n *increment* operations is n .

Q2. Average-case vs expected (hiring problem). Suppose we must hire a new employee. There are n candidates arriving sequentially, one each day.

It takes I time units to interview a candidate, and it takes H units to hire them.

We want to have at all times the best possible person for the job. After interviewing each applicant, if they are better than our current employee, we hire them immediately (and fire our current employee).

We can compare two candidates in constant time.

```

hire(cand[1..n]):
    curr = dummy candidate // compares worse than anyone
    for i = 1..n:
        interview cand[i]
        if cand[i] is better than curr:
            hire cand[i]
            curr = cand[i]

```

Suppose m candidates are hired. Then the worst-case runtime is in $\Theta(In + Hm)$.

We can rank each candidate with a unique number between 1 and n and use $rank[i]$ to denote the rank of candidate i . We adopt the convention that a higher ranked applicant corresponds to a better qualified applicant.

Note that the ordered list

$$\langle rank[1], \dots, rank[n] \rangle$$

is a permutation of the list $\langle 1, \dots, n \rangle$.

1. Describe an instance that achieves the runtime $\Omega(Hn)$.
2. Show that in the **average-case** we hire a new candidate $O(\log n)$ times.

Q3. Partially Sorted. Let $0 < \epsilon < 1$. Suppose that we have an array A of n items such that the first $n - n^\epsilon$ items are sorted. Describe an $O(n)$ time algorithm to sort A .

Q4. Searching lower bound. Show that any comparison-based searching algorithm uses $\Omega(\log n)$ comparisons.

1. in the worst case; and
2. in the average case.

Additional problems

Q7. Indicator random variables. Let $A[1..n]$ be an array of n distinct integers. We say that a pair (i, j) is an *inversion* of A if $i < j$ but $A[i] > A[j]$. Suppose that the elements of A form a uniform random permutation of $\langle 1, \dots, n \rangle$. Find the expected number of inversions of A .

Q8. Recursion tree. Use a recursion tree to give an asymptotically tight solution to the recurrence

$$T(n) = T(n - a) + T(a) + cn$$

for constants $a \geq 1, c > 0$.

Q9. Perfect square.

- (a) Give an algorithm to test whether a given $n \geq 1$ is a perfect square in $O(\log n)$ time.
- (b) If n is a perfect square, compute \sqrt{n} in $O(\log n)$ time.

Q10. Shifted array. We are given an array a of n numbers that was initially in sorted order, and then was shifted by some unknown amount. For example,

$$[1, 2, 3, 4, 5, 6] \mapsto [3, 4, 5, 6, 1, 2].$$

Find the minimum element of a in $O(\log a)$ time.