CS 473 ✧ Fall 2024

Homework 4

Due Tuesday, October 8, 2024 at 9pm Central Time

Unless a problem specifically states otherwise, you may assume a function RANDOM that takes a positive integer k as input and returns an integer chosen uniformly and independently at random from $\{1, 2, \ldots, k\}$ in $O(1)$ time. For example, to model a fair coin flip, you could call RANDOM(2).

---

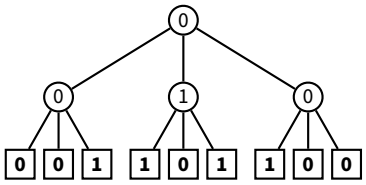
0. **[Warmup only. Do not submit solutions!]**

   After sending his loyal friends Rosencrantz and Guildenstern off to Norway, Hamlet decides to amuse himself by repeatedly flipping a fair coin until the sequence of flips satisfies some condition. For each of the following conditions, compute the *exact* expected number of flips until that condition is met.

   (a) Hamlet flips heads.
   (b) Hamlet flips both heads and tails (in different flips, of course).
   (c) Hamlet flips heads twice.
   (d) Hamlet flips heads twice in a row.
   (e) Hamlet flips heads followed immediately by tails.
   (f) Hamlet flips more heads than tails.
   (g) Hamlet flips the same number of heads and tails.
   (h) Hamlet flips the same positive number of heads and tails.
   (i) Hamlet flips more than twice as many heads as tails.

   [Hint: Be careful! If you're relying on intuition instead of a proof, you're probably wrong.]

1. A ***majority tree*** is a complete ternary tree in which every leaf is labeled either 0 or 1. The *value* of a leaf is its label; the *value* of any internal node is the majority of the values of its three children. For example, if the tree has depth 2 and its leaves are labeled $1, 0, 0, 0, 1, 0, 1, 1, 1$, the root has value 0.



A majority tree with depth 2.

   It is easy to compute the value of the root of a majority tree with depth n in $O(3^n)$ time, given the sequence of 3^n leaf labels as input, using a simple postorder traversal of the tree. Prove that this simple algorithm is optimal, and then describe a better algorithm. More formally:

(a) Prove that *any* deterministic algorithm that computes the value of the root of a majority tree *must* examine every leaf. *[Hint: Consider the special case $n = 1$. Recurse.]*

(b) Describe and analyze a randomized algorithm that computes the value of the root in worst-case expected time $O(c^n)$ for some explicit constant $c < 3$. *[Hint: Consider the special case $n = 1$. Recurse.]*

2. Let A and B be two arrays of integers of length n and m respectively. Assume that all integers in the union of two arrays are distinct. Suppose also that for any indices $i \in [n]$ and $j \in [m]$, we can compare whether $A[i] > B[j]$ or $A[i] < B[j]$ in $O(1)$ time, but we ***cannot*** directly compare two elements in A or two elements in B.

(a) Show that with the allowed comparisons, it is not possible to perfectly sort the union of two arrays $A \cup B$, unless the two arrays perfectly interleave, i.e. if we sort the union $A \cup B$ then every two elements of A is separated by at least one element of B and vice-versa.

For example, arrays $A = [2, 5, 8, 12]$ and $\boldsymbol{B = [1, 3, 6, 10, 14]}$ perfectly interleave, because $A \cup \boldsymbol{B} = [\mathbf{1}, 2, \mathbf{3}, 5, \mathbf{6}, 8, \mathbf{10}, 12, \mathbf{14}]$, but arrays $A' = [2, 5, 8, 12]$ and $\boldsymbol{B' = [1, 3, 4, 6, 10, 14]}$ do not perfectly interleave, because their union $A' \cup \boldsymbol{B'} = [\mathbf{1}, 2, \underline{\mathbf{3}, \mathbf{4}}, 5, \mathbf{6}, 8, \mathbf{10}, 12, \mathbf{14}]$ contains two consecutive elements from A'.

(b) Give a randomized algorithm that only uses the allowed comparisons and sorts the union of the two arrays $A \cup B$ when they perfectly interleave. Analyze the expected running time of your algorithm. *[Hint: Recall the algorithm to sort nuts and bolts.]*

3. An ***interval*** is a set of contiguous integers $\{a, a+1, \ldots, b\}$ where $a \le b$. We use the notation $[a, b]$ to denote the corresponding interval.

Consider the following process for sampling a set of two distinct integers $\{i, j\}$ that both lie in the base interval $[1, n]$: first, we choose two non-overlapping sub-intervals $I \cup J$ of $[1, n]$. Then, we sample integers $i \in I$ and $j \in J$ uniformly at random from each sub-interval.

(a) Suppose we first **deterministically** choose two disjoint intervals I and J of $[1, n]$, and then choose elements $i \in I$ and $j \in J$ **uniformly at random** from each sub-interval. Prove that the set $\{i, j\}$ is is **not** uniformly distributed among all sets of two integers in $[1, n]$.

Since deterministic methods don't work, let's consider *randomized* ways of sampling the two sub-intervals. We would like to make the sub-intervals as large as possible. The ***score*** of two non-overlapping sub-intervals $I \cup J$ of $[1, n]$ is given by

$$\text{score}(I \cup J) = \frac{n}{|I|} + \frac{n}{|J|}.$$

A small score implies that both sub-intervals are large. For instance, if both sub-intervals have size $n/2$, their score is 4, which is best possible up to constant factors. However, if one or both sub-intervals have size $O(1)$, their score is $\Theta(n)$.

To simplify our sampling problem, suppose n is a power of 2. Consider the following randomized algorithm for choosing a pair of disjoint sub-intervals. The input to the algorithm is a *base interval* $[lo, hi]$ whose length $hi - lo + 1$ is a power of 2, and another integer $0 \le m \le 2$. The algorithm returns a set of m disjoint sub-intervals inside the base interval $[lo, hi]$. The top-level call is SAMPLESUBINTERVAL$(1, n, 2)$

```
SAMPLESUBINTERVAL(lo, hi, m):
  if m = 0
      return {}
  if m = 1
      return {[lo, hi]}
  ℓ ← hi − lo + 1          ⟨⟨length of base interval⟩⟩
  mid ← ⌊(lo + hi)/2⌋      ⟨⟨end of left half⟩⟩
  roll ← RANDOM(4ℓ − 4)
  if roll ≤ ℓ − 2
      return SAMPLESUBINTERVAL(lo, mid, 2)
  else if roll ≤ 3ℓ − 2
      I ← SAMPLESUBINTERVAL(lo, mid, 1)
      J ← SAMPLESUBINTERVAL(mid + 1, hi, 1)
      return I ∪ J
  else
      return SAMPLESUBINTERVAL(mid + 1, hi, 2)
```

In words, depending on the roll of an unbalanced three-sided die, the algorithm either recursively samples the left half of the base interval, recursively samples the right half of the base interval, or simply returns the left and right halves of the base interval.

(b) Prove that if we choose integers $i \in I$ and $j \in J$ uniformly at random from the sub-intervals $I \cup J$ returned by SAMPLESUBINTERVAL$(1, n, 2)$, the resulting set $\{i, j\}$ is uniformly distributed among all sets of two distinct integers in $[1, n]$.

(c) Show that the *expected* score of the pair of sub-intervals returned by returned by SAMPLESUBINTERVAL$(1, n, 2)$ is $O(\log n)$, which is close to optimal.

*(d) **Extra Credit:** Consider a similar process for sampling a set of k distinct integers in $[1, n]$: first select non-overlapping intervals $I_1 \cup I_2 \cdots \cup I_k$ of $[1, n]$ and then for each $\ell \in [k]$ sample one point $x_\ell \in I_\ell$ uniformly. The score of such a tuple of k intervals is defined as

$$\text{score}(I_1 \cup I_2 \cdots \cup I_k) = \frac{n}{|I_1|} + \frac{n}{|I_2|} + \cdots + \frac{n}{|I_k|}.$$

Give a randomized algorithm to sample k non-overlapping intervals of $[1, n]$ such that (i) sampling one point from each interval given by the algorithm results in a uniformly distributed set of k distinct integers in $[1, n]$, and (ii) the *expected* score of the intervals is within a $O(\log n)$ factor of the optimal score.

[Hint: What is the best possible score with k intervals? Generalize the algorithm given above to a set of size k. We recommend solving all the other homework problems, including problem o, before attempting this one.]