Lecture #19: Reintroduction to Discrete Probability Theory

What To Do Before the Lecture

1. Complete the preparatory viewing or reading for Lecture #19 — noting that each video will probably be understandable if you play it at double speed.
2. ***Print*** and read through the rest of this document and — if you have time — try to solve the problems!

Problems To Be Solved

1. Consider the problem of searching for a copy of `true` in a Boolean array A of length n, for a positive integer n. An algorithm that solves this problem — returning an integer i such that $0 \leq i \leq n-1$ and $A[i] = \texttt{true}$ if the array contains a copy of `true` and throwing a `NoSuchElementException` otherwise — is shown in Figure 1 on page 2.

   The ***experiment***, to be considered, will concern the use of this algorithm to search for a copy of `true` in an input array. The ***sample space*** will include all possible input arrays with length n, so that

   $$\Omega = \{(\alpha_1, \alpha_2, \ldots, \alpha_n) \mid \alpha_i \in \{\texttt{true}, \texttt{false}\} \text{ for every integer } i \text{ such that } 1 \leq i \leq n\} \quad (1)$$

   — where an element

   $$\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n) \quad (2)$$

   represents an input array A such that $A[i] = \alpha_i$ for $1 \leq i \leq n$ — and so that $|\Omega| = 2^n$.

```
integer search ( boolean[] A ) {
1.  integer n := A.length
2.  integer i := 0
3.  while (i < n) {
4.    if (A[i] == true) {
5.      return i
      }
6.    i := i + 1
    }
7.  throw a NoSuchElementException
}
```

Figure 1: Algorithm to Search in a Boolean Array

Suppose we wish to analyze the number of elements of the array that are checked when the algorithm is executed — which is the same as the number of times that the test at line 4 is checked before the execution of the algorithm ends.

(a) Suppose that $\vec{\alpha}$ is as shown at line (2), above. How is the number of times that the test at line 4, on outcome $\vec{\alpha}$, related to $\alpha_1, \alpha_2, \ldots, \alpha_n$? How small can this number be? How large can it be?

(b) Suppose, now, that ***every input array with length n is equally likely*** — so that the ***uniform probability distribution*** $\mathsf{P} : \Omega \to \mathbb{R}$ will be used for an analysis.

Let k be an integer such that $1 \leq k \leq n$. What is probability that the step at line 4 is executed exactly k times (during an execution on a "randomly chosen" input array, using this probability distribution)?

2. Consider the experiment, involving the algorithm in Figure 1, once again — so that the same space Ω and the random variable C are as above — but consider a different probability distribution.

   In particular, let p be a real number such that $p < 1$, and let $\mathsf{P} : \Omega \to \mathbb{R}$ such that the following properties are satisfied:

   - For every integer i such that $0 \leq i \leq n - 1$, α_i is `true` with probability p.
   - The events

     $$\alpha_0 = \texttt{true}, \alpha_1 = \texttt{true}, \alpha_2 = \texttt{true}, \ldots, \alpha_{n-1} = \texttt{true}$$

     are ***mutually independent***.

   Once again, let k be an integer such that $1 \leq k \leq n$. What is the probability that the step at line 4 is executed exactly k times (during an execution on a "randomly chosen" input array, using this probability distribution)?

```
integer rSearch ( boolean[] A ) {
1. integer n := A.length
2. integer i := 0
3. while (i < n) {
4.    Choose an integer j uniformly and randomly from the set
                        {0, 1, 2, ..., n − 1}.
5.    if (A[j] == true) {
6.      return j
      }
7.    i := i + 1
    }
8. throw a NoSuchElementException
}
```

Figure 2: Randomized Algorithm to Search in a Boolean Array

3. Consider the same problem, but consider a *different* algorithm, namely the ***randomized algorithm*** shown in Figure 2.

   (a) Explain, as precisely as you can, why this algorithm ***does not*** solve the search problem that is being considered — but "comes close" in some significant sense. Describe, as precisely as you can, how this algorithm can produce *incorrect* output, or throw an exception when it should not.

Consider the execution of this algorithm on some ***fixed*** input array A. Once again, let us consider the number of times that array entries are checked, that is, the number of times that the step at line 5 of this *new* algorithm is executed.

(b) Suppose, first, that $A[i] = \texttt{true}$ for every integer i such that $0 \leq i \leq n-1$. Describe the ***sample space*** and ***probability distribution*** that would be used in this first (simple) case. What can be said about the number of times that the step at line 5 is executed?

(c) Suppose, next, that $A[i] = \texttt{false}$ for every integer i such that $0 \leq i \leq n-1$, instead. Describe the ***sample space*** and ***probability distribution*** that would be used in *this* (simple) case. What can be said about the number of the times that the step at line 5 is executed?

The more interesting situation is one where some of the entries of the input array A are `true` and others are false. Let

$$S_T = \{i \in \mathbb{N} \mid 0 \leq i \leq n-1 \text{ and } A[i] = \texttt{true}\} \tag{3}$$

— so that $1 \leq |S_T| \leq n-1$ in this case. Let

$$p = \frac{|S_T|}{n} \tag{4}$$

— so that

$$\tfrac{1}{n} \leq p \leq 1 - \tfrac{1}{n}.$$

(d) Describe the ***sample space*** and ***probability distribution*** that would be used in this (more interesting) case.

(e) Let k be an integer such that $1 \leq k \leq n$. What is the probability that the step at line 5 is executed exactly k times?

More Problems for Later

The following problems might also be of interest. You are welcome to discuss these with the instructor during office hours or on Piazza.

4. Consider the deterministic algorithm in Figure 1 once again. Let k be an integer such that $1 \leq k \leq n$ and consider the case that the input array includes exactly k copies of "`true`" and exactly $n - k$ copies of "`false`" — with each of the input arrays, that satisfy this condition, occurring with the same probability.

   Describe the probability distribution that models *this* situation. Then state the probability that the probability that the step at line 4 is executed exactly ℓ times, for each integer ℓ such that $1 \leq \ell \leq n$.

5. Unfortunately the randomized algorithm, considered in Question #3, was not guaranteed to return correct output in all cases.

   (a) Describe a very small change to this algorithm, that results in an algorithm that does almost the same thing but whose output is always correct.

   Hint: You will only need to change one line in the pseudocode. The deterministic algorithm, considered in Questions #1 and #2, will be useful.

   (b) You have probability described an algorithm such that, if the input array has length n, then array entries can be accessed up to $2n$ times (for at least one input). Say as much as you can about the probability that that array entries are accessed exactly ℓ times, for every integer ℓ such that $1 \leq \ell \leq 2n$.