# Lecture #20: Random Variables and Expectation

## What To Do Before the Lecture

1. Complete the preparatory viewing or reading for Lecture #20 — nothing 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. Let $\Omega$ be a finite sample space with a probability distribution $\mathsf{P} : \Omega \to \mathbb{R}$.

   Let $X : \Omega \to \mathbb{R}$ such that $X$ only has a finite set of values, $\{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, for some positive integer $k$. Prove that (with respect to the probability distribution P)

   $$\mathsf{E}[X] = \sum_{i=1}^{k} \alpha_i \times \mathsf{P}(X = \alpha_i).$$

   ***Note:*** This property is quite useful, when one wishes to compute the expected value of a random variable — and you may use it without proving it when solving problems in this course.

```
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

2. 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, above. Note that this is the algorithm that was considered in the first problem in the previous lecture presentation.

   Consider the same ***experiment*** as in the problem in the previous lecture presentation — so that the ***sample space*** will include all possible input arrays with length $n$:

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

   — where an element

$$\vec{\alpha} = (\alpha_1, \alpha_2, \dots, \alpha_n) \tag{2}$$

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

   Suppose 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.

   Suppose, as well, that 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. This can be modelled using a random variable

$$C : \Omega \to \mathbb{N} \tag{3}$$

where, for an element $\vec{\alpha}$ of the same space, $C(\vec{\alpha})$ is the number of times that this step is executed when the input is the array represented by $\vec{\alpha}$.

What is the ***expected value*** of the random variable, $C$, with respect to this probability distribution?

***Note:*** Some of the results that were proved when answering the first question on the previous tutorial exercise will be useful when solving this problem.

3. Consider the experiment, involving the algorithm in Figure 1, once again — so that the same space $\Omega$ 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 $0 \leq i \leq n-1$, $\alpha_i$ 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***.

   What is the expected value of the random variable $C$ with respect to *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

4. Consider the same problem, but consider a *different* algorithm, namely the ***randomized algorithm*** shown in Figure 2, above. Note that this algorithm was considered in the third problem in the previous lecture presentation.

   (a) Suppose, first, that $A[i] = \texttt{true}$ for every integer $i$ such that $0 \leq i \leq n-1$. A ***sample space*** and ***probability distribution***, that corresponds to this case, were identified during the previous lecture presentation.

   Use this to compute the expected number of times that the step at line $5$ is executed.

(b) Suppose, next, that $A[i] = \texttt{false}$ for every integer $i$ such that $0 \leq i \leq n-1$. A ***sample space*** and ***probability distribution***, corresponding to this case, were also identified during the previous lecture presentation.

Use this to compute the expected number of times that the step at line $5$ is executed in this case, as well.

The more interesting situation is one where some of the entries of the input array $A$ are $\texttt{true}$ and others are false. As in the previous lecture presentation, let

$$S_T = \{i \in \mathbb{N} \mid 0 \leq i \leq n-1 \text{ and } A[i] = \texttt{true}\} \tag{4}$$

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

$$p = \frac{|S_T|}{n} \tag{5}$$

— so that

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

Once again, a ***sample space*** and ***probability distribution*** corresponding to this case were identified during the previous lecture presentation.

(c) Use this to compute the expected number of times that the step at line $5$ is executed in *this* (more interesting) case.