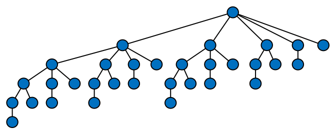
Lecture 13. Random Variables and Expected Value

CpSc 8400: Algorithms and Data Structures

Brian C. Dean



School of Computing
Clemson University
Spring, 2016

Analyzing Randomized Algorithms and Data Structures

- So far, our analyses of randomized algorithms have focused on proving "with high probability" results.
  - For example, randomized quicksort runs in O(n log n) time w.h.p.
- Our primary tools so far: the **randomized reduction lemma** and the **union bound.**
- Today, we'll see another type of randomized analysis where we look at the "average", or "expected" running time of an algorithm.

2

Random Variables : Introduction

- In elementary school, we all learn that a variable is a name, or "placeholder" for some specific value.
- By contrast, a **random variable** stands for a numeric value that is determined by the outcome of some random experiment. Examples:
  - Let X be the number we see when we roll a 6-sided die.
  - Let Y be the number of heads in 100 fair coin flips.
  - Let T be the number of comparisons made by applying randomized quicksort to a length-n array.
- Every r.v. has an associated probability distribution.
- Think of the r.v. as a placeholder for a value that will be "instantiated", or drawn from this distribution once our random experiment actually happens.
  - Example: X takes values 1..6 each with probability 1/6.
  - The distributions of Y and T are somewhat more complicated.

3

Equations Involving Random Variables

- Just like we can write equations involving standard variables, we can also write equations involving random variables.
- Simple example: Z = X + Y, where
  - X is the number we roll on our first roll of a die,
  - Y is the number we roll on the second roll, and
  - Z is the sum of the two numbers on both dice.
- More interesting example: $T = \Sigma_j X_j$, where
  - T is the total amount of time spent by an algorithm.
  - X_j is the amount of time spend only on element j.
- **Key property**: any equation or inequality involving random variables must hold for every possible random instantiation of these variables.

4

Events Derived from Random Variables

- Let D be the largest face value we see when we roll two 6-sided dice.
- The probability distribution for D is:

  1: 1/36   2: 3/36   3: 5/36   4: 7/36   5: 9/36   6: 11/36
- “D is even” and “D > 3” are events, so we can consider computing **Pr**[D is even] or **Pr**[D > 3].
- Another example: If T denotes the running time of randomized quicksort applied to an array of length n, then

  Pr[T = O(n log n)] ≥ 1 – 1/n^c, for any constant c > 0.
- Just be careful never to write **Pr**[D].

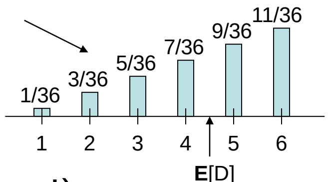
  This is a “syntax error”, since D is a random variable and not and event (a set of outcomes).

5

Expected Value

- The expected value of a discrete random variable X, denoted **E**[X], is defined as

  E[X] = $\Sigma_{\text{values } v}$ v **Pr**[X = v].
- Think of **E**[X] informally as the “center of mass” of X’s probability distribution.
- Example: Let D be the max of two dice rolls. Recall that D has this probability distribution.
  - Thus, **E**[D] = 1(1/36) + 2(3/36) + 3(5/36) + 4(7/36) + 5(9/36) + 6(11/36) = 161/36 = 4 $^{17}/_{36}$



- Careful: Don’t write **E**[A] if A is an event (another syntax error!)

6

Computing Expected Values

There are generally 4 different ways we will compute expected values in this class:

1. Directly using the definition $\mathbf{E}[X] = \Sigma_v \, v \, \mathbf{Pr}[X = v]$.
2. The special case of an **indicator** random variable.
3. The special case of a **geometric** random variable.
4. Expressing a complicated random variable in terms of a sum of simpler r.v.'s and applying **linearity of expectation**.

7

Indicator Random Variables

- Suppose E is some event.
  (e.g., "roll a 3 on a 6-sided die").
- Let X be a random variable taking the value 1 when E occurs, and 0 otherwise.
- We say X is an **indicator** random variable for E.
  (also called a **Bernoulli** r.v.)
- Easy to compute **E**[X]:

$$\begin{aligned}\mathbf{E}[X] &= 1\ \mathbf{Pr}[X = 1] + 0\ \mathbf{Pr}[X = 0] \\ &= \mathbf{Pr}[X = 1] \\ &= \mathbf{Pr}[E] \quad \text{(= 1/6 in our example).}\end{aligned}$$

- The expected value of any indicator r.v. is just the probability of its associated event.

8

Geometric Random Variables (Expected Trials Until Success)

- Suppose we perform a series of independent random trials, where each trial “succeeds” with probability p.
- Let X denote the number of trials until the first success.
- X has a **geometric** probability distribution.
- **E**[X] = 1 / p (easy to prove via definition).
- Example: if X denotes the number of dice rolls until we first see a ‘3’, then **E**[X] = 6.

9

Linearity of Expectation

- **E**[] is a **linear** operator:
  - **E**[cX] = c**E**[X] if c is a consant
  - **E**[X + Y] = **E**[X] + **E**[Y]
- The above holds for **any** random variables X and Y, regardless of whether or not they are independent!
- This gives us a very powerful tool for computing expectations of complicated random variables.
- Example:
  - Let H be the total number of heads in 100 coin flips.
  - Computing **E**[H] by definition of **E**[] looks messy!
  - Instead, write $H = H_1 + H_2 + \ldots + H_{100}$, where $H_j = 1$ if the j^{th} coin toss comes up heads.
  - Now $\mathbf{E}[H] = \mathbf{E}[H_1] + \ldots + \mathbf{E}[H_{100}] = 100(1/2) = 50$.

10

Linearity of Expectation : Examples

- What is the expected number of inversions in a randomly-permuted array?
  - How about the expected running time of insertion sort on a random permutation?
- If everyone in this room is wearing a hat and we randomly permute the hats, what is the expected number of people ending up with their original hat?
- If we randomly throw n balls into m bins, what is the the expected number of balls landing in a specific bin?
- If we put n people in a room, what is the expected number of pairs of people sharing the same birthday? (assuming all birthdays equally likely)

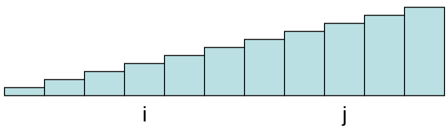
Example : The Coupon Collector Problem

- Suppose each time we open a box of breakfast cereal, we are equally likely to obtain one of n coupon types.
- What's the expected number of boxes we need to open before obtaining at least one of each type?
  - T: total # of boxes we open
  - T_j : # of boxes we open in the j^{th} "phase" where we have discovered j – 1 coupon types and are waiting for the j^{th}.
  - $\mathbf{E}[T_j] = n / (n - j + 1)$.
  - Now $T = T_1 + \ldots + T_n$
  - So $\mathbf{E}[T] = \mathbf{E}[T_1 + \ldots + T_n] = \mathbf{E}[T_1] + \ldots + \mathbf{E}[T_n]$
    $= n/n + n/(n-1) + \ldots + n/1$
    $= n(1/n + 1/(n-1) + \ldots + 1/2 + 1) = \Theta(n \log n)$.

Randomized Quicksort

- Let T be a random variable giving the total number of comparisons performed by randomized quicksort (note that T also tells us the total running time).

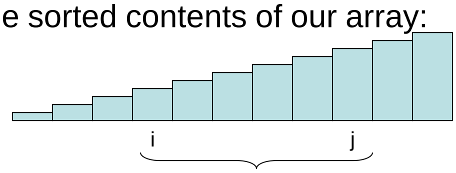
- Write $T = \Sigma_{i<j} T_{ij}$, where $T_{ij} = 1$ if the elements of ranks i and j are compared during the execution of randomized quicksort.
- Since the T_{ij}'s are indicator random variables, we have
  $\mathbf{E}[T_{ij}] = \mathbf{Pr}$[rank i element compared to rank j element]
  so $\mathbf{E}[T] = \Sigma_{i<j}\ \mathbf{Pr}$[rank i element compared to rank j element]



13

Randomized Quicksort

- $\mathbf{E}[T] = \Sigma_{i<j}\ \mathbf{Pr}$[rank i element compared to rank j element]
- Picture the sorted contents of our array:



Among this set of j – i + 1 elements, each is equally likely to be chosen earliest as a pivot during the execution of the algorithm.

- If an interior element (not rank i or j) is chosen earliest, this splits the rank i and j elements into separate sub-problems and prevents them from ever being compared.
  (alternatively, the rank i and j elements will only be compared if one of them is chosen first as a pivot among all the elements in this set.)
- So $\mathbf{Pr}$[rank i and j compared] $\leq 2 / (j - i + 1)$, and therefore
  $\mathbf{E}[T] = \Sigma_{i<j}\ 2/(j - i + 1)$.

14

Randomized Quicksort

- $\mathbf{E}[T] = \Sigma_{i<j}\, 2/(j - i + 1)$

$$= 2\, \Sigma_{i=1..n}\, \Sigma_{j=i+1..n}\, 1/(j - i + 1)$$

$$= 2\, \Sigma_{i=1..n}\, \Sigma_{k=i..n-i+1}\, 1/k \qquad (k = j - i + 1)$$

$$\leq 2\, \Sigma_{i=1..n}\, \Sigma_{k=i..n}\, 1/k$$

$$= 2n\, \Sigma_{k=i..n}\, 1/k$$

$$= 2n\, \underbrace{(1 + 1/2 + 1/3 + \ldots + 1/n)}_{\text{“Harmonic” Series}}$$

$$= 2n\, \Theta(\log n)$$

$$= \Theta(n \log n)$$

“Expected Value” Analogs of the Randomized Reduction Lemma

- Take an algorithm that in each iteration has probability $\geq p$ of reducing our problem size to $\leq q$ times its original size, where p and q are constants.
- Randomized reduction lemma: O(log n) iterations w.h.p.
- Now think of the operation of our algorithm as consisting of $\log_{1/q} n$ “phases”, where a phases consists of all iterations up to and including a “good” iteration where we reduce problem size:
  - Pr[specific iteration is good] $\geq p$
  - **E**[iterations up until and including the first good iteration] $\leq 1/p = O(1)$.
  - **E**[total iterations] $\leq (1/p)\log_{1/q} n = \Theta(\log n)$ by linearity of expectation. (this isn't surprising, since the stronger randomized reduction lemma gives us the same bound only w.h.p.)

Randomized Quickselect

- Randomized quickselect fits the model on the previous slide perfectly!
- Each iteration has probability ≥ 1/3 of being "good", and reducing problem size to ≤ 2/3 original size.
- So it operates in $\log_{(3/2)} n$ "phases":
  - 1^{st} phase: each iteration takes ≤ n units of work.
  - 2^{nd} phase: each iteration takes ≤ (2/3)n units of work.
  - 3^{rd} phase: each iteration takes ≤ $(2/3)^2$n units of work…
  - **E**[iterations per phase] ≤ 3.
  - So according to linearity of expectation,
    E[total work] ≤ $3[n + 2/3n + (2/3)^2 n + \ldots] = O(n)$.

17

"Per Element" Expected Analysis

- Recall (due to the union bound):

  "If an algorithm spends O(T) time on a generic input element w.h.p., then it spends O(nT) time on all input elements w.h.p."
- Linearity of expectation gives us a similar result:

  "If an algorithm spends O(T) expected time on a generic input element, then it spends O(nT) expected time on all input elements."
- So when trying to find the expected running time of an algorithm, we can often simplify this problem to the computation of expected time on one element.

18

Randomized Quicksort Revisited

- Let's think again about randomized quicksort.
- We've already shown an $O(n \log n)$ running time both w.h.p and also in expectation.
- There is another alternate expected running time proof that corresponds exactly to the w.h.p. proof:
- **W.h.p. proof:**

  Randomized reduction lemma $\rightarrow$ $O(\log n)$ / element w.h.p.

  Union bound $\rightarrow$ $O(n \log n)$ time for all elements w.h.p.
- **Expected running time proof:**

  Linearity of expectation $\rightarrow$ $O(\log n)$ expected time / element

  Linearity of expectation $\rightarrow$ $O(n \log n)$ total expected time.

19