Recitation 5 – *Probability and Collect*

Parallel and Sequential Data Structures and Algorithms, 15-210 (Spring 2014)

February 11th, 2014

1 Announcements

- How did *Bignum* go?
- *BabbleLab* is out! We'll go over it a bit at the end of lecture.
- Questions about homework or lecture?

2 Probability Basics

Many of you have seen probability before. We'll quickly go through the basics and move on to more interesting things.

2.1 Conditional Probability

$P(A|B) = P(A \cap B)/P(B)$. This identity can be intuitively demonstrated on a Venn diagram.

2.2 Independence

Say I throw a fair six-sided dice three times. What is the probability of rolling an even number, then a four, then an even number again?

Solution 2.0 $\frac{1}{24}$. This is because rolling an even number on a six-sided dice occurs with $\frac{1}{2}$ probability, and rolling a four occurs with $\frac{1}{6}$ probability. Because these events are independent, we can simply multiply the probabilities together. $(\frac{1}{2})^2 \cdot \frac{1}{6} = \frac{1}{24}$.

Formally, event A is independent from event B iff $P(A|B) = P(A)$; i.e. the probability of A remains the same whether or not B has occurred.

Prove that if an event A is independent from the event B, then event B is independent from A.

Solution 2.0 From the identity $P(A|B) = P(A \cap B)/P(B)$, it follows that

$$P(A|B) = P(A) \iff P(A \cap B)/P(B) = P(A) \iff P(A \cap B)/P(A) = P(B) \iff P(B|A) = P(B)$$

so we see that event A being independent from event B also implies that B is independent from A.

2.3 Inclusion-Exclusion

$P(A \cup B) = P(A) + P(B) - P(A \cap B)$. This principle can be intuitively demonstrated on a Venn diagram. Note that $P(A \cap B) = 0$ if and only if A and B are mutually exclusive, so be careful when directly adding probabilities.

2.4 Expected value

A random variable (usually a 1-dimensional number in the reals) on a sample space Ω is a real-valued function on Ω, thus having type: $\Omega \to \mathbb{R}$. While ordinary variables are considered to have a single unknown value, random variables are considered to have all possible values, each with a particular probability. This isn't quite true, but it's good enough for 210, where we only deal with discrete probability.

The expected value of a random variable is the weighted mean of its possible values, where each value x is weighted by the probability that x is the outcome. This can be written as the following: $\mathbf{E}[\mathbf{X}] = \sum_{w \in \Omega} w * P(w)$

What is the expected value of rolling a fair six-sided dice once?

> **Solution 2.0** Let X be a random variable representing the outcomes of a fair six-sided dice. $\mathbf{E}[X] = \frac{1}{6}(1 + 2 + 3 + 4 + 5 + 6) = 3.5$

What is the expected value of rolling an unfair dice, where even numbers are rolled with probability $\frac{2}{9}$ and odd numbers are rolled with probability $\frac{1}{9}$?

> **Solution 2.0** Let X be a random variable representing the outcomes of the unfair six-sided dice. $\mathbf{E}[X] = \frac{2}{9}(2 + 4 + 6) + \frac{1}{9}(1 + 3 + 5) = \frac{33}{9}$

The expected value function $\mathbf{E}$ (also known as "expectation") is a linear function, meaning that for any value c, $\mathbf{E}[c \cdot X] = c \cdot \mathbf{E}[X]$ and $\mathbf{E}[X] + \mathbf{E}[Y] = \mathbf{E}[X + Y]$. Furthermore, $\mathbf{E}[X \cdot Y] = \mathbf{E}[X]\mathbf{E}[Y]$ holds when X and Y are independent. For example, in the random process where we have n red dice and m blue dice, and will roll them and multiply the sum of the red values with the sum of the blue values, the expected value is simply $(3.5 \cdot n) \cdot (3.5 \cdot m) = 12.25 \cdot nm$. For more detail about the linearity of expectation, see the lecture slides.

2.5 Probability Bounds

Theorem 2.1. *(Markov's Inequality)* $P(|X| \geq a) \leq \dfrac{\mathbf{E}[|X|]}{a}$

Proof:

Solution 2.0

Proof. Let Y be a random indicator variable that is 1 when $|X| \geq a$ and 0 otherwise. Note that $\mathbf{E}[Y] = P(|X| \geq a)$.

Trivially, $aY \leq |X|$, because when $|X| < a$ then $Y = 0$ and the left hand side is zero; and when $|X| \geq a$, then $Y = 1$ and the left hand side is a.

Thus, $\mathbf{E}[aY] \leq \mathbf{E}[|X|] \implies a\mathbf{E}[Y] \leq \mathbf{E}[|X|] \implies P(|X| \geq a) \leq \dfrac{\mathbf{E}[|X|]}{a}$ □

2.6 Cost of Table.collect

The following code was shown in class and implements `Table.collect`(S).

`Seq.reduce (Table.merge Seq.append)` $\langle\,\rangle$ $\{k \mapsto \langle v \rangle : (k, v) \in S\}$

What does this code actually do?

Derive (tight) asymptotic upper bounds on the work and span for the code in terms of $n = |S|$. You can assume the comparison function used by the table takes constant work, and that the Sequence is a array sequence.

Solution 2.0 Work = O($nlogn$) Span = O($log^2 n$)

Remember that a reduce tree does a pairwise reduce, which means that the depth of the tree is log n. Thus, the total work of this function will be O(log n * the amount of work done at each level)

The work done at each level consists of an append and a merge.
A worst case sequence is one in which there is only 1 unique key, and it points to different values, ie a sequence like <(1,1), (1,2), (1,3), (1,4)>.
If we look at the top level, we do an append (which costs O(2)) twice, so that's a total of O(n) work, if n is the length of the sequence.
Now if we look at the merge, the work of merge is O(m log (1 + n/m)), where m is the minimum length of the two sequences and n is the max. Now we can assume that they're the same length, which gives you log(2) - a constant, so it can be ignored.
In the first level, therefore, we're doing $\frac{n}{2}$ times log(2), which is just $\frac{n}{2}$.
Extrapolating that down the tree, you end up getting $n + \frac{n}{2}$ work at each level.

Therefore, the total work is $O(nlogn)$

Span of reduce = $O(lognS(f(x,y)))$
Span of Table.merge = $O(mlog(1+\frac{n}{m}) + maxS(f(x,y))$
Span of Seq.append = $O(1)$
Span of Table.merge = $O(logn)$
Span of reduce = $O(log^2 n)$

Therefore the total work is $O(log^2 n)$

2.7 identifyRepeats

Implement a function `identifyRepeats` that given a sequence of integer returns a table that maps each unique element to either `ONE` or `MULTI`. For example

`identifyRepeats`$(\langle 7,2,4,2,3,2,1,3\rangle)$

would return $\{1 \mapsto$ `ONE`$, 2 \mapsto$ `MULTI`$, 3 \mapsto$ `MULTI`$, 4 \mapsto$ `ONE`$, 7 \mapsto$ `ONE`$\}$.

```
structure T (* A table with integer keys *)

datatype repeat = ONE | MULTI

fun identifyRepeats (S : int Seq) :  repeat T.table =
```

Solution 2.0

```
fun identifyRepeats (S : int seq) : repeat Table.table =
    let
      val S' = map (fn x => (x, ())) S
      val grouped = collect Int.compare S'
      fun multi (k, v) =
          (k, if length v = 1
              then ONE
              else MULTI)
    in
      Table.fromSeq (map multi grouped)
    end
```

Work = $O(nlogn)$ Span = $O(log^2 n)$

What is the (tight) assymptotic upper bound for work and span for your implementation in terms of $n = |S|$?

2.8 BabbleLab

BabbleLab is due a week from now, and so we're going to be going over some information that may be useful for the lab.

This lab deals with strings, so it will be useful to look at the standard ML basis library on strings to help convert the input text (the corpus) into something that will be easier to manipulate and use.

One of the types that will be used extensively is

```
type 'a hist = (a * int) seq
```

which associates each value with it's frequency count. Think about how you could implement this, given what we've gone over in recitation today.