Recitation 5 – *Probability and Collect*

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

September 23rd, 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?

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.

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

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

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 $\frac{1}{9}$?

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:

2.6 Cost of Table.collect

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

```
Seq.reduce (Table.merge Seq.append) ⟨ ⟩ ⟨{k ↦ ⟨v⟩} : (k, v) ∈ 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.

2.7 Making a Histogram

Implement a function `histogram` that, given a sequence of integers, returns a table that maps each unique element to the number of times it appears in the sequence. For example

```
histogram(⟨7, 2, 4, 2, 3, 2, 1, 3⟩)
```

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

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

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

2.8 Working for the Registrar

Suppose you are working on Academic Audit. An academic advisor comes to you asking for a table mapping each of her students to a set of the courses they've taken in their time at CMU, and you need to do this fast. What you have right now is a sequence of tuples, where each tuple is a string (the student's name), and a set of integers (the course numbers for their schedule in one semester). Write a function, `bulkAudit`, to do this for her.

```
fun bulkAudit (S : (string * (int Set)) Seq) :  (int Set) T.table =
```

Table/Set Operations	*Work*	*Span*
`size`(T) `singleton`(k,v)	$O(1)$	$O(1)$
`filter` f T	$O\left(\sum_{(k,v)\in T} W(f(v))\right)$	$O\left(\lg\|T\| + \max_{(k,v)\in T} S(f(v))\right)$
`map` f T	$O\left(\sum_{(k,v)\in T} W(f(v))\right)$	$O\left(\max_{(k,v)\in T} S(f(v))\right)$
`tabulate` f S	$O\left(\sum_{k\in S} W(f(k))\right)$	$O\left(\max_{k\in S} S(f(k))\right)$
`find` T k `insert` f (k,v) T `delete` k T	$O(\lg\|T\|)$	$O(\lg\|T\|)$
`extract` (T_1, T_2) `merge` f T_1 T_2 `erase` (T_1, T_2)	$O\left(m\lg(\frac{n+m}{m})\right)$	$O\left(\lg(n+m)\right)$
`domain` T `range` T `toSeq` T	$O(\|T\|)$	$O(\lg\|T\|)$
`collect` S `fromSeq` S	$O(\|S\|\lg\|S\|)$	$O(\lg^2\|S\|)$
`intersection` (S_1, S_2) `union` (S_1, S_2) `difference` (S_1, S_2)	$O\left(m\lg(\frac{n+m}{m})\right)$	$O\left(\lg(n+m)\right)$

where $n = \max(|T_1|, |T_2|)$ and $m = \min(|T_1|, |T_2|)$. For `reduce` you can assume the cost is the same as `Seq.reduce f init (range(T))`. In particular `Seq.reduce` defines a balanced tree over the sequence, and `Table.reduce` will also use a balanced tree. For `merge` and `insert` the bounds assume the merging function has constant work.