Randomness and Computation

or, “Randomized Algorithms”

Mary Cryan

School of Informatics
University of Edinburgh

warm-up: Birthday Paradox

30 people in a room. What is the probability they share a birthday?

- Assume everyone is equally likely to be born any day (*uniform at random*). Exclude Feb 29 for neatness.
- Generate birthdays one-at-a-time from the pool of 365 (*principle of deferred decisions*).

Probability p_{30diff} that all birthdays are *different* is

$$p_{30diff} = \prod_{i=1}^{30} \frac{365-(i-1)}{365} = \prod_{i=1}^{30} \left(1 - \frac{(i-1)}{365}\right) = \prod_{j=1}^{29} \left(1 - \frac{j}{365}\right).$$

Recall that $1 + x < e^x$ for all $x \in \mathbb{R}$, hence $(1 - \frac{j}{365}) < e^{-j/365}$ for any j.

warm-up: Birthday Paradox

Hence

$$p_{30diff} < \prod_{j=1}^{29} e^{-j/365} = \left(\prod_{j=1}^{29} e^{-j}\right)^{\frac{1}{365}} = \left(e^{-\sum_{j=1}^{29} j}\right)^{\frac{1}{365}} = \left(e^{-435}\right)^{\frac{1}{365}},$$

last step using $\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$. $\left(e^{-435}\right)^{\frac{1}{365}} \sim e^{-1.19} \sim 0.3$. So with probability of at least 0.7, two people at the party share a birthday.

More general framework:

n birthday options, m party guests

warm-up: General Birthday Paradox

More general framework:

n birthday options, m party guests

Probability $p_{all-m-diff}$ that all are *different* is

$$p_{all-m-diff} = \prod_{j=1}^{m} \left(1 - \frac{(j-1)}{n}\right) = \prod_{j=1}^{m-1} \left(1 - \frac{j}{n}\right).$$

Continuing,

$$p_{all-m-diff} \leq \prod_{j=1}^{m-1} e^{-j/n} = \left(\prod_{j=1}^{m-1} e^{-j}\right)^{\frac{1}{n}} = \left(e^{-\sum_{j=1}^{m-1} j}\right)^{\frac{1}{n}} = e^{-\frac{(m-1)m}{2n}},$$

approximately $e^{-m^2/2n}$.
Suppose we set $m = \lfloor\sqrt{n}\rfloor$, then $e^{-m^2/2n}$ becomes $\sim e^{-0.5} \sim 0.6$.

Balls in Bins

- *m* balls, *n* bins, and balls thrown *uniformly at random* into bins (usually one at a time).
- Magic bins with no upper limit on capacity.
- Common model of random allocations and their affect on overall *load* and *load balance*, typical *distribution* in the system.
- (by the birthdays analysis) we know that for $m = \Omega(\sqrt{n})$, then there is some constant probability $c > 0$ of a birthday clash (BOARD).
- “Classic" question - what does the distribution look like for $m = n$? Max load? (*with high probability* results are what we want).

Balls in Bins maximum load

Lemma (5.1)

Let n balls be thrown independently and uniformly at random *into n bins. Then for sufficiently large n, the maximum load is bounded above by $\frac{3\ln(n)}{\ln\ln(n)}$ with probability at least* $1 - \frac{1}{n}$.

Proof The probability that bin i receives $\geq M$ balls is *at most*

$$\binom{n}{M}\frac{n^{m-M}}{n^m} = \binom{n}{M}\frac{1}{n^M}.$$

Expanding $\binom{n}{M}$, this is

$$\frac{n\ldots(n-M+1)}{M!}\frac{1}{n^M} \leq \frac{1}{M!}.$$

To bound $(M!)^{-1}$ note that for any k, we have $\frac{k^k}{k!} \leq \sum_{i=0}^{\infty}\frac{k^i}{i!} = e^k$, hence $\frac{1}{k!} \leq (\frac{e}{k})^k$. *Or use Stirling ...*

Balls in Bins maximum load

Proof of Lemma 5.1 cont'd.

So bin i gets $\geq M$ bins with probability at most

$$\left(\frac{e}{M}\right)^M.$$

Set $M =_{def} \frac{3\ln(n)}{\ln\ln(n)}$. Then the probability that *any* bin gets $\geq M$ balls is (using the Union bound) at most

$$n \cdot \left(\frac{e \cdot \ln\ln(n)}{3\ln(n)}\right)^{\frac{3\ln(n)}{\ln\ln(n)}} \leq n \cdot \left(\frac{\ln\ln(n)}{\ln(n)}\right)^{\frac{3\ln(n)}{\ln\ln(n)}} = e^{\ln(n)} \left(\frac{\ln\ln(n)}{\ln(n)}\right)^{\frac{3\ln(n)}{\ln\ln(n)}}.$$

Again using properties of ln, this expands as

$$e^{\ln(n)} \left(e^{\ln\ln\ln(n) - \ln\ln(n)}\right)^{\frac{3\ln(n)}{\ln\ln(n)}} = e^{\ln(n)} \left(e^{-3\ln(n) + 3\frac{\ln(n)\ln\ln\ln(n)}{\ln\ln(n)}}\right).$$

□

Balls in Bins maximum load

Proof of Lemma 5.1 cont'd.

Grouping the $\ln(n)$s in the exponents, and evaluating, we have

$$e^{-2\ln(n)} \cdot e^{3\frac{\ln(n)\ \ln\ln\ln(n)}{\ln\ln(n)}} \;=\; \frac{1}{n^2} n^{3\frac{\ln\ln\ln(n)}{\ln\ln(n)}}.$$

If we take n "sufficiently large" ($n \geq e^{e^{e^4}}$ will do it), then $\frac{\ln\ln\ln(n)}{\ln\ln(n)} \leq 1/3$, hence the probability of *some* bin having $\geq M$ balls is at most

$$\frac{1}{n}.$$

$\square$

Can derive a matching proof to show that "with high probability" there will be a bin with $\Omega(\frac{\ln(n)}{\ln\ln(n)})$ balls in it. We are going to skip over this (can read in Sections 5.3 and 5.4, won't be examined)

$\Omega(\cdot)$ bound on the maximum load (chat)

- We implicitly used the *Union Bound* in our proof Lemma 5.1, when we multiplied by n on slide 7. However, in reality, bin i has a lower chance of being “high" (say $\Omega(\frac{\ln(n)}{\ln\ln(n)})$) if other bins are already “high" (the “high-bin" events are *negatively correlated*).
- This means that we can't use the same approach as in Theorem 5.1 to prove a partner result of $\Omega(\frac{\ln(n)}{\ln\ln(n)})$.
- Solution is to use the fact that for the binomial distribution $B(m, \frac{1}{n})$ for an individual bin, that as $n \to \infty$,

$$\Pr[X = k] = \binom{m}{k}\left(\frac{1}{n}\right)^k\left(1 - \frac{1}{n}\right)^{m-k} \to \frac{e^{-m/n}(m/n)^k}{k!}$$

  (ie, close to the probabilities for the Poisson distribution with parameter $\mu = m/n$)
- The Poisson's aren't independent but the dependance can be limited to an extra factor of $e\sqrt{m}$ (Section 5.4).

Average-case analysis of Bucket Sort

- Items to be sorted are natural numbers from some bounded range $[0, 2^k)$, some large k.
- We have a collection of empty “buckets" (extendable arrays or lists).
- Each bucket has an “index" used to access it.
- We have some value m, the “number of prefix bits" (substantially smaller than k). We will have a bucket for each individual $\{0, 1\}^m$.

Algorithm BUCKETSORT$(a_1, \ldots, a_n)$

1. Do a linear scan of the inputs, adding a_i to the bucket matching its first m bits.
2. **for** every $b \in \{0, 1\}^m$ **do**
3. Sort bucket b with any $O(n^2)$ sorting algorithm.

Average-case analysis of Bucket Sort

Imagine that we draw the n inputs to BUCKETSORT independently and uniformly at random from $\{0,1\}^k$. Hence ...

The first-m-bits of the inputs are independently uniform from $\{0,1\}^m$.

Each a_i has probability $\frac{1}{2^m}$ of entering any bucket.

Bucket Sort can be seen as a “balls-in-bins" experiment.

Running time is $\Theta(n)$ for the linear scan of 1. The *expected* running time for 2.-3. will be $\mathrm{E}[\sum_{b \in \{0,1\}^m} c \cdot (X_b^2)]$, where X_b is the number of inputs landing in bucket b, and $c > 0$ is the fixed constant of the $O(n^2)$ algorithm.

We want to evaluate $\mathrm{E}[\sum_{b \in \{0,1\}^m} c \cdot (X_b^2)] \ = \ \sum_{b \in \{0,1\}^m} c \cdot \mathrm{E}[X_b^2]$.

We are now going to use an unexpected “trick" where we exploit the “second moment" of Binomial random variables to bound the $\mathrm{E}[X_b^2]$.

Average-case analysis of Bucket Sort

Realise each X_b is a binomial random variable $B[n, \frac{1}{2^m}]$ with

$$\mathrm{E}[X_b^2] = n(n-1)2^{-2m} + n2^{-m}.$$

Multiplying by 2^m (for each $b \in \{0,1\}^m$), and by c, this gives expected time for 2.-3. at most

$$c \cdot \left(n^2 2^{-m} + n\right).$$

Choose m carefully to satisfy $m \geq \lg(n)$ and we see that this ensures the expected number of steps for 2.-3. is at most $2 \cdot c \cdot n$.

The rest of the course

Lect 11	Random Graphs and Hamilton cycles (Section 5.6)
Lects 12-13	The Probabilistic method, derandomization via Conditional expectation (bit more than half Chapter 6)
	Will hold a “tutorial" in the lecture slot for Friday 10th March (we will cover questions about Coursework 1, and “end of printout" questions between now and then)
Lects 14-15	Markov chain basics (first half Chapter 7)
Lects 16-17	The Monte Carlo method (some of Chapter 9)
Lects 18-20	Mixing time bounds for Markov chains (Chapter 11)
	I will hold the second “tutorial" in the Lecture slot of Friday 7th April (our final meeting).

References and Exercises

- Sections 5.1, 5.2 of “Probability and Computing". And if you are interested in the Ω bound for the $\Theta(\frac{\ln(n)}{\ln\ln(n)})$ result, read Sections 5.3 and 5.4 also.
- Section 5.5 on Hashing is worth a read and has none of the Poisson stuff (I'm skipping it because of time limitations).

Exercises

- Exercise 5.3 (balls in bins when $m = c \cdot \sqrt{n}$).
- Exercise 5.10 (sequences of empty bins; this is a bit more tricky)