Randomized Algorithms
Lecture 1

Kousha Etessami & Raul Garcia-Patron Sanchez

School of Informatics
University of Edinburgh

Lectures and tutorials

Lectures:

- 14:10-15:00 Monday, Appleton Tower, room AT 2.05;
- 14:10-15:00 Wednesday, Appleton Tower, room AT 2.05.

Tutorials:

- Mondays 10:00-10:50, Old College, Teaching Room 6;

Lecturers & tutor

Lecturers:

- Kousha Etessami, IF 5.20 (Weeks 1-6)
  Email: `ketessami@inf.ed.ac.uk`
  Webpage: `http://homepages.inf.ed.ac.uk/kousha`
  Weekly Office hour: Wednesdays, 11:00am–12:00noon.

- Raul Garcia-Patron Sanchez, IF-3.06A (Weeks 7-10)
  Email: `rgarcia3@ed.ac.uk`
  Webpage: `https://www.inf.ed.ac.uk/people/staff/Raul_Garcia-Patron_Sanchez.html`

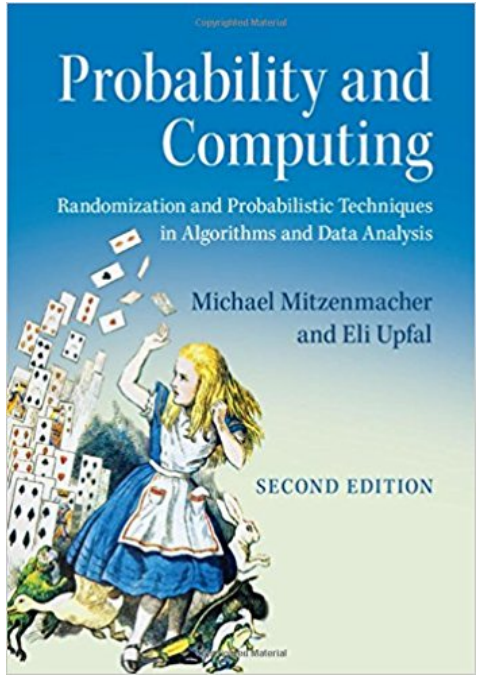
Tutor:

- Graham Freifeld
  Email: `g.freifeld@sms.ed.ac.uk`

Randomized Algorithms

- **Question:** What can we compute efficiently (exactly or approximately), when we have the ability to "toss coins" in our algorithms? In particular, can we do better than any "deterministic" algorithm?
- Of course, an algorithm that exploits randomness (random coin flips), and hence is no longer deterministic, can exhibit:
  a. variations in the answer/output it computes on the SAME input,
     or
  b. variations in its running time on the SAME input,
     or
  c. both!
- Although we can have variations in both running time and/or the answer returned by a randomized algorithm, we will aim to calculate the **expected** running-time, the **expected** value returned, and/or the **probability** of each possible answer. And we will aim to show that these are "good", by proving **bounds** on the expected running-time, the (expected) values returned, and their probabilities.

Textbook (essential for the course)



Probability and Computing - Randomized Algorithms and Probabilistic Analysis, by Michael Mitzenmacher and Eli Upfal; Cambridge University Press, 2017 (2nd ed).

- You are welcomed to work with Edition 1 if you find a cheaper copy.
- The library has a copy of Edition 1.

Syllabus

Chapter 1: Introduction, "Monte Carlo" and "Las Vegas" algorithms (Some Examples: checking polynomial identities and matrix products, a randomized min-cut algorithm.)

Chapter 2: Background in probability, random variables, expectation (Applications: Coupon collecting, ...)

Chapter 3: Moments & Deviations (Markov's, Chebyshev's, and Jensen's inequalities)

Chapter 4: Chernoff Bounds (Some applications: statistical sampling & parameter estimation; set balancing)

Chapter 5: Birthday paradox, and Balls in Bins (Applications: Load balancing, Hashing)

Chapter 6: The Probabilistic Method (random graphs, ramsey numbers, algorithms for Max-cut; (Max-)Satisfiability; Lovász Local Lemma and applications)

Chapters 7,11,12: Markov Chains and Random walks (hitting and cover times, Markov chain Monte Carlo, mixing times, coupling, applications to sampling and counting algorithms)

Course Webpage

Slides will be provided for each lecture (and notes sometimes if appropriate) on the course's DRUPAL webpage.

Recordings of lectures will be accessible from the LEARN page for RA.

The DRUPAL page will also provide access to tutorial sheets and solutions. Coursework assignments will be available on LEARN, and you will submit coursework via GradeScope, using the LEARN webpage for the course.

You will also need the book.

Pre-requisites

No official prerequisites.

However, strong maths background is necessary, especially *Discrete Maths*, Linear Algebra, Discrete Probability, and *confidence in proving things*.

We expect you to have covered an "Algorithms class" in the past, and to have done well in it (we can waive that, if your maths is very strong).

If you're not sure, come and speak to me.

Math you should know

You should know:

- what it means to *prove* a theorem (induction, proof by contradiction, etc ...) and to be *confident* in your ability to do this.
- The definitions of the main asymptotic operators $O(\cdot)$, $\Omega(\cdot)$, $\Theta(\cdot)$, and how to reason about them.
- How to multiply matrices or polynomials. Basic linear algebra.
- Some probability theory, definition of expectation (1st moment) and variance (related to 2nd moment), linearity of expectation, simple probability distributions and how they behave.
- Some graph theory.

Your work (formative and summative assessments)

- tutorial sheets

  8 tutorials, once a week, weeks: 3 – 10.
- Coursework 1 (due Thursday of week 6)
- Coursework 2 (due Wednesday of week 10)

Coursework (summative assessment)

We have 2 Courseworks (problem-solving and proofs), and both will be marked to give you feedback. Each coursework will count for 10% of the overall mark. Details are:

- Coursework 1. Worth 10%
  - OUT Thurs, 12th of October (Thurs week 4)
  - DUE **12:00 noon** Thurs, 26th October (Thurs week 6)
  - FEEDBACK by Thurs, 9th of November (Thurs week 8)
- Coursework 2. Worth 10%
  - OUT Wednesday, 8th of November (Wed, week 8)
  - DUE **12:00 noon** Wed, 22nd of November (Wed, week 10)
  - FEEDBACK by Friday, 1st of December (Friday, week 11)

Feedback given will include marks to individual sub-parts of questions, comments on scripts to explain why marks were lost. We will also distribute sample solutions for the coursework.

Let's get started:
a randomized algorithm for verifying polynomial identities

Suppose we are given two polynomials $F(x)$ and $G(x)$, where $F(x)$ is expressed as a product of d degree 1 polynomials, and $G(x)$ is given as usual, as a sum $\sum_{i=0}^{d} c_i x^i$ of monomial terms.

Question: *How fast can we decide whether $F(x) \equiv G(x)$?*

For example,

- $F(x) = (x-1)(x+2)(x-3)(x+4)(x-5)(x+6)$;
- $G(x) = x^6 - 7x^3 + 720$.

The simple "multiply out" algorithm on $F(x)$ (using no randomness) can give us the answer in $\Theta(d^2)$ time. (Assume for simplicity that each addition or multiplication requires one time step.)

(There exists a different deterministic algorithm, using FFT, for "multiply out" in $\Theta(d \cdot \lg^2(d))$ steps, but we will not discuss that algorithm.)

We instead use randomness to decide equility, *without multiplying out* $F(x)$.

Testing polynomial identities using random sampling

- Choose an integer x_0 *uniformly at random* from the set of integers $\{1, \ldots, 100d\}$.
- Calculate $F(x_0)$. For each degree 1 polynomial $(ax + b)$ **...** we do 1 addition and 2 multiplication.

  Overall this takes at most d additions and $2d$ multiplications.
- We also calculate $G(x_0)$. We first do d multiplications to get all of x_0, x_0^2, x_0^3, ..., x_0^d. Then we multiply each term with its coefficient and add everything up.

  Overall this takes at most d additions and $2d$ multiplications.
- Next, compare the two resulting integers, and answer "YES" if they are the same, "NO" otherwise.

Definition: *Uniformly at Random (u.a.r.)* - every possible outcome is chosen with equal probability. (So, if there are m possible outcomes, each is chosen with probability $\frac{1}{m}$.)

Monte Carlo algorithm

This is a *Monte Carlo* algorithm (coined by Stan Ulam), meaning that with some probability **it can give a wrong answer**.

The error in this case is **one-sided**.

- **Obvious Claim 1:** If $F(x)$ does equal $G(x)$, the algorithm always returns "YES".
- **Claim 2:** If $F(x) \not\equiv G(x)$, "NO" is returned with probability $\geq \frac{99}{100}$ (i.e., the failure probability is $\leq \frac{1}{100}$).

  Why?

Testing polynomial identities

The probability that the algorithm gives an incorrect answer ("YES" when it should be "NO") equals

$$\frac{|\{x : F(x) = G(x)\} \cap \{1, \ldots, 100d\}|}{100d} \leq \frac{|\{x : F(x) = G(x)\}|}{100d}$$

If $F(x) \not\equiv G(x)$, the set $\{x : F(x) = G(x)\}$ is equal to the set of roots of of the **non-zero** polynomial $(F - G)(x)$, namely, $\{x : (F - G)(x) = 0\}$.

Fact: The number of roots of any **non-zero** polynomial is at most its degree.
And the degree of $F - G$ is clearly at most d.

So the error probability is $\leq \frac{d}{100d} = \frac{1}{100}$.

Reducing the error probability

- One option to improve error rate is to increase the size of the sample set, e.g., by sampling a random integer from $\{1, \ldots, 1000d\}$. The error probability would drop to $\leq \frac{1}{1000}$. This improvement is not "free" though: it's more work to sample from larger sets.
  (Not officially costed by us yet, but roughly we need to flip $\lceil \log_2(m) \rceil$ fair coins in order to sample uniformly from the set $[m] = \{1, \ldots, m\}$. So, sampling from the set $[k \cdot d]$ costs $\lceil \log_2(k) + \log_2(d) \rceil$ coin flips. So sampling from from $[1000d]$ costs more than sampling from $[100d]$, but not much more: $\log_2(100) \approx 6.64$ whereas $\log_2(1000) \approx 10$.)

- Alternatively, suppose we run *two* random trials to test $F(x) \stackrel{?}{\equiv} G(x)$, first drawing x_1 u.a.r. from $\{1, \ldots, 100d\}$ and checking whether $F(x_1) \stackrel{?}{=} G(x_1)$, next drawing x_2 u.a.r. from $\{1, \ldots, 100d\}$ and checking whether $F(x_2) \stackrel{?}{=} G(x_2)$.

  We return "YES" if *both* calculations give matching values, otherwise we return "NO".

Repetition method to improve error probability of checking polynomial identities

Observation

This refined algorithm (based on repetition) again gives one-sided error*:*

- *If* $F(x) \equiv G(x)$*, certainly we will see that* $F(x_1) = G(x_1)$, and *that* $F(x_2) = G(x_2)$ *(answer: "YES").*
- *If* $F(x) \not\equiv G(x)$*, we will show the algorithm returns "NO" with probability at least* $(1 - \frac{1}{100^2})$*, i.e., with failure probability at most* $\left(\frac{1}{100^2}\right)$.

Refining the verification of polynomial identities (analysis)

Two options for "repeated sampling" from $\{1, \ldots, 100d\}$ (or any discrete set): *with replacement* or *without replacement*.

with replacement: We draw the random value x_2 uniformly at random from $\{1, \ldots, 100d\}$ (including x_1 as an option).
For this case, the two *events* of "generating x_1" and "generating x_2" are *(mutually) independent*.

Definition (1.3)

Events A_1, A_2, A_3, ..., A_k are said to be **mutually independent** if for any subset $I \subseteq \{1, \ldots, k\}$,

$$\Pr[\cap_{i \in I} A_i] = \prod_{i \in I} \Pr[A_i].$$

Refining the verification of polynomial identities (analysis)

with replacement (cont'd): Recall that if $F(x) \not\equiv G(x)$, then $(F-G)(x)$ has *at most* d roots; hence there are *at most* d values in $\{1,\ldots,100d\}$ that could give matching values for $F(x)$, $G(x)$.

If H_1 is the event that "a root of $(F-G)(x)$" is generated on this first trial, then $\Pr[H_1] \leq d/100d = (1/100)$.

But sampling with replacement, the outcome of the 2nd trial is *independent* of what happened before. So H_2 (the probability of generating a root of $(F-G)(x)$ on the 2nd trial) is *independent* of H_1. Furthermore, it happens to have identical probability.

The probability that *both* experiments would draw a root of $(F-G)(x)$ is, by independence (Definition 1.3), equal to

$$\Pr[H_1] \cdot \Pr[H_2] \leq \frac{1}{100} \cdot \frac{1}{100} = 1/10000.$$

Refining the verification of polynomial identities (analysis)

without replacement: Suppose we have already checked x_1 and found $F(x_1) = G(x_1)$ (else we'd finish, with "NO"). Next, we draw a value u.a.r. from the set $\{1, \ldots, 100d\} \setminus \{x_1\}$.

Events H_1' and H_2', corresponding to the event that the first and second trial, respectively, generate a root of $(F - G)(x)$, are now no longer independent.

Definition (1.4)

Let A and B be two events, with $\Pr[B] > 0$. The **conditional probability** of event A given event B is

$$\Pr(A \mid B) \doteq \frac{\Pr[A \cap B]}{\Pr[B]}.$$

Refining the verification of polynomial identities (analysis)

without replacement (cont'd): Let us apply Definition 1.4, where B is H_1' and A is H_2'. We want to calculate $\Pr[H_1' \cap H_2']$ (the two samples both giving a false match, i.e., both giving a root of (F-G)(x)). Using Definition 1.4, $\Pr[H_1' \cap H_2'] = \Pr[H_1'] \cdot \Pr[H_2' \mid H_1']$.

We know $\Pr[H_1'] \leq \frac{1}{100}$.

For $\Pr[H_2' \mid H_1']$, note that conditioned on H_1' (and hence conditioned on the integer removed being a root of $(F-G)(x)$), we have one less root of $(F-G)(x)$, $d'-1$ instead of d', say, remaining in the set $\{1,\ldots,100d\} \setminus \{x_1\}$. Hence $\Pr[H_2' \mid H_1'] = \frac{d'-1}{100d-1}$. Then

$$\Pr[H_1' \cap H_2'] \;=\; \Pr[H_1'] \cdot \Pr[H_2' \mid H_1'] \;\leq\; \frac{d'}{100d} \cdot \frac{d'-1}{100d-1} \;<\; \frac{1}{100^2},$$

where we have used the fact that $d' \leq d$ to deduce both that $\frac{d'}{100d} \leq \frac{1}{100}$ and that $\frac{d'-1}{100d-1} < \frac{1}{100}$.

Refining the verification of polynomial identities (wrapup)

We can generalize, to do any $k > 1$ different (mutually independent) trials of values sampled from $\{1, \ldots, 100d\}$.
The probability that all k trials result in a root of $(F - G)(x)$ (and hence the probability of failure) is at most $\frac{1}{100^k}$.

- Hence we have "one-sided error" at most $\frac{1}{100^k}$.
- The probability of failure (returning "YES" when $F(x), G(x)$ are non-identical) is always a little bit better in the "without replacement" case). However, not much better, and note that random sampling without replacement can be more costly/difficult to implement (so, not often used in practice).
- This iterated testing algorithm will take $\Theta(k \cdot d)$ time.
- Does it make any sense to do $k > d$ iterations??

Reading Assignment

Start reading Chapter 1 of "Probability and Computing".

You can also already start looking at Tutorial Sheet 1 (accessible on the DRUPAL page for RA). Please work on that tutorial sheet prior to Monday of Week 3 when it will be covered at the tutorial.