Randomness and Computation
or, “Randomized Algorithms”
Mary Cryan
School of Informatics
University of Edinburgh

Matrix multiplication verification
Assume that the values in the $n \times n$ matrices $A$, $B$, $C$ are integers over some field like $\mathbb{F}_2$ (also known as $GF(2)$), or indeed any $\mathbb{F}_p$ for prime $p > 2$, or even the standard field over $\mathbb{Z}$.

The algorithm is parametrized by some natural number $k > 1$.

Algorithm MMVERIFY($n, A, B, C$)
1. for $j = 1, \ldots, k$ do
2. Generate $\bar{x}$ uniformly at random from $\{0, 1\}^n$
3. Calculate $\bar{y}_B = B \cdot \bar{x}$ in $\Theta(n^2)$ time.
4. Calculate $\bar{y}_{AB} = A \cdot \bar{y}_B$ in $\Theta(n^2)$ time.
5. Calculate $\bar{y}_C = C \cdot \bar{x}$ in $\Theta(n^2)$ time.
6. if $\bar{y}_{AB} \neq \bar{y}_C$
7. return “no”
8. return “yes”

Analysing MMVerify
We have already (Friday 19th) shown that each of steps 3., 4., 5. can be done in $\Theta(n^2)$ for a specific vector $\bar{x}$ of length $n$. Now for the analysis . . .

We will show “One-sided error” (like polynomial identity testing)

- $AB \equiv C$: In this case, we know that $AB \cdot \bar{x} = C \bar{x}$ for every $\bar{x} \in \{0, 1\}^n$. Hence $\text{MMVERIFY}$ is guaranteed to return the value “yes”.
- $AB \not\equiv C$: We will now show that in this case, that when a vector $\bar{x}$ is drawn uniformly at random (uar) from $\{0, 1\}^n$, the probability that $AB \cdot \bar{x} = C \cdot \bar{x}$ is at most $1/2$.

Then we will consider the effect of doing $k$ trials.

Analysing MMVerify: $AB \not\equiv C$
Consider the two $n \times n$ matrices $AB$ and $C$. They are non-identical, so there must be at least one cell $(i^*, j^*)$ such that the values $(AB)_{i^*, j^*}$ and $C_{i^*, j^*}$ are different.

Let $D = (AB - C)$. Then equivalently, we have $D_{i^*, j^*} \neq 0$.

Consider row $i^*$ of $D$, and consider its product with a vector $\bar{x} \in \{0, 1\}^n$:

$$\sum_{j=1}^{n} D_{i^*, j} \cdot x_j.$$ 

This gives the value for position $i^*$ in the length-$n$ vector computed by $D \cdot \bar{x}$. We will show that this will be 0 with probability at most $1/2$. 

RC (2017/18) – Lecture 3 – slide 1

RC (2017/18) – Lecture 3 – slide 2

RC (2017/18) – Lecture 3 – slide 3

RC (2017/18) – Lecture 3 – slide 4
Analysing MMVerify: $AB \not\equiv C$

When drawing a random $\bar{x} \in \{0,1\}^n$ uniformly at random (u.a.r.), each $\bar{x}$ has equal probability $(1/2^n)$.

This is equivalent to choosing the value $x_i \in \{0,1\}$ independently with probability $1/2$, for each $i \in [n] = \{1, \ldots, n\}$.

Use this in the analysis (principle of deferred decisions).

Write $\sum_{j=1}^n D_{i,j} \cdot x_j$ as

$$\left( \sum_{j \in [n] \setminus \\{j^*\}} D_{i,j} \cdot x_j \right) + D_{i,j^*} \cdot x_{j^*}$$

Think about sampling $\bar{x}$ (deferred decisions) as $\{0,1\}^{n-1}$ vector first, followed by the value for $x_{j^*}$ last.

RC (2017/18) – Lecture 3 – slide 5

Analysing MMVerify: $AB \not\equiv C$

Notice that the $k$ repeated trials fit into the paradigm of “sampling with replacement”.

Let $E_j$ be the event that the $j$-th sampled $\bar{x}$ satisfies $D \cdot \bar{x} = 0$ (ie $AB \cdot \bar{x} = C \cdot \bar{x}$).

$E_1, \ldots, E_k$ are all pairwise independent. Thus, applying Defn 1.3 repeatedly,

$$\Pr[\bigcap_{j=1}^k E_j] = \prod_{j=1}^k \Pr[E_j].$$

We have already shown that $\Pr[E_j] \leq 1/2$.

Hence $\Pr[\bigcap_{j=1}^k E_j]$, the probability that the algorithm returns “yes” is at most $1/2^k$ (in the case of $AB \not\equiv C$).

(note: I don’t like/approve-of the stuff with Bayes in the book)

RC (2017/18) – Lecture 3 – slide 7

Karger’s Min-Cut Algorithm

We are given an undirected graph $G = (V, E)$ with $|V| = n$, and are interested in computing a “min cut”; that is, a partition of $E$ into two non-empty sets $S, V \setminus S$, such that the following quantity is minimized:

$$\{e = (u, v) : u \in S, v \in V \setminus S\}$$

There are many deterministic algorithms which can solve this problem in polynomial-time (discuss). Karger’s algorithm (faster) uses random sampling:

Repeatedly, choose an edge uniformly at random (from the not-yet contracted edges) and conjoin its endpoints.

When there are just two “vertices” left, return that cut.

RC (2017/18) – Lecture 3 – slide 8
Karger's Min-Cut Algorithm - Example

Min degree of any vertex is 3. This means the min cut of the graph is no larger than 3.

RC (2017/18) – Lecture 3 – slide 9

Karger's Min-Cut Algorithm - Example

Let's focus on the larger "S" as our target to achieve the min cut value of 3, for the analysis.
Our hope is that the random process will avoid the magenta edges with some decent probability.

RC (2017/18) – Lecture 3 – slide 11

Karger's Min-Cut Algorithm - Example

The green lines are marking groups of vertices which form "minimum-sized" cuts in the graph.
The analysis of Karger's algorithm involves showing there is good probability a randomly-chosen edge avoids the cut boundary.
Let's focus on the larger "S" as our target to achieve the min cut value of 3, for the analysis.

RC (2017/18) – Lecture 3 – slide 10

Karger's Min-Cut Algorithm - Example

Randomly choosing our first edge to contract...
First choice of edge is uniformly at random from all 15 edges, as it happens we avoid the magenta ones (not surprising as there are only 3, probability of hitting one of them is only 1/5)

RC (2017/18) – Lecture 3 – slide 12
Karger's Min-Cut Algorithm - Example

Second choice of edge is uniformly at random from remaining 14 edges, again we avoid the magenta ones (not surprising as there are only 3, probability of hitting one of them is only 3/14)

Karger's Min-Cut Algorithm - Example

Went to avoid ever contracting one of the cut edges (as has happened in this step) as we end up with a composite vertex which has hidden one of the cut edges (and apart from hiding this particular cut, we probably have reduced the chance of finding a good one).

Karger's Min-Cut Algorithm - Example

Again we have lost another vertex (only 6 now) and another edge (only 13 now). Note that the probability we avoided magenta for two steps was \( (12/15) \times (11/14) \) which is 22/35.

Notice that when we contract an edge, we are putting those vertices into the same "side" of our eventual cut. So we never want to contract a magenta edge. But our algorithm does not know which are the cut edges.

Analysis of Karger's Min Cut coming Friday.