ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Mohsen Ghaffari, Angelika Steger, Emo Welzl, Peter Widmayer

Algorithms, Probability, and Computing **Solutions KW45** **HS16**

Solution 1: Randomized Algebraic Algorithms

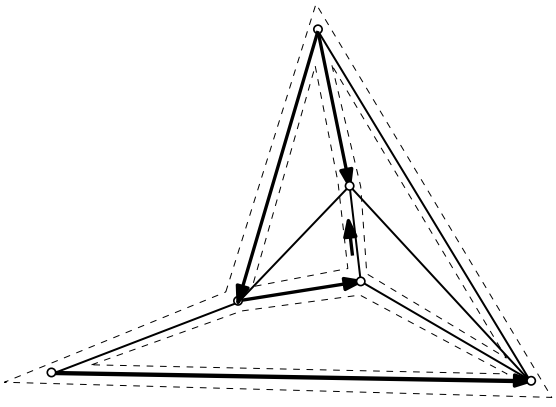
(a) This is a type of argument that we are by now very familiar with. Let $i, j \in \{1..n\}$ be indices such that A_{ij} is nonzero. Consider the i-th entry $(Ax)_i$ of the matrix-vector product. It calculates as

$$(Ax)_i = \sum_{k \in \{1..n\} \setminus \{j\}}^{n} A_{ik} x_k + A_{ij} x_j.$$

Since the x_i are being chosen independently of one another, we may prescribe any order in which they are evaluated; let us evaluate x_j last. Once all terms in the sum are being fixed, then it takes a fixed real value s. At most one out of the possible choices for x_j can yield $A_{ij} x_j = -s$ and so the probability that this happens is at most $1/3$, yielding the claim.

(b) Take the union of two supposed perfect matchings. If they differ, then their union must contain an even cycle.

(c) There are six nice cycles (four of length 4 and two of length 6). The orientation of one edge (the one which is oriented in the following figure, but not on the exercise sheet) is determined by this. Two more edges can be oriented arbitrarily (not part of any nice cycle), and there are two possibilities to orient the remaining three edges (fixing the orientation of any one of those edges determines the orientations of the other two).



Solution 2: Checking Matrix Multiplication

(a) This is discussed in the lecture notes (but we did not cover it in the lecture, so we do it here), section 4.1.

(b) Assume that the matrix C is wrong in exactly the i-th row compared to the correct product AB. We define $D = AB - C$. This is a zero-matrix except in the i-th row, $D_{i,-} \neq (0, \ldots, 0)$. The probability of detecting an error equals $\Pr\left[(D_{i,-})^T x \equiv_2 1\right]$, where $x \in_{\text{u.a.r.}} \{0,1\}^n$. We have seen in the lecture (not in the lecture notes) that this probability equals $\frac{1}{2}$. Here is a formal proof of the argument: Let j be such that $D_{ij} = 1$. We have

$$(Dx)_i = (D_{i,-})^T x = \sum_{k=1}^{n} D_{ik} x_k = \underbrace{\sum_{k=1, k\neq j}^{n} D_{ik} x_k}_{=: S} + \underbrace{D_{ij}}_{=1} x_j,$$

hence

$$\begin{aligned}
\Pr[(Dx)_i = 1] &= \Pr\left[S + x_j = 1\right] \\
&= \Pr\left[S + x_j = 1 \,|\, S = 0\right] \cdot \Pr[S = 0] + \Pr\left[S + x_j = 1 \,|\, S = 1\right] \cdot \Pr[S = 1] \\
&= \Pr\left[x_j = 1 \,|\, S = 0\right] \cdot \Pr[S = 0] + \Pr\left[x_j = 0 \,|\, S = 1\right] \cdot \Pr[S = 1] \\
&= \Pr\left[x_j = 1\right] \cdot \Pr[S = 0] + \Pr\left[x_j = 0\right] \cdot \Pr[S = 1] \\
&\qquad\qquad \text{because } S, x_j \text{ are independent} \\
&= \frac{1}{2} \cdot \Pr[S = 0] + \frac{1}{2} \cdot \Pr[S = 1] \\
&= \frac{1}{2}.
\end{aligned}$$

Solution 3: The Schwartz-Zippel Theorem is Tight

Let $\{a_1, \ldots, a_d\} \subseteq S$ be a set of d elements. We define the polynomial $p(x_1, \ldots, x_n)$ as

$$p(x_1, \ldots, x_n) := (x_1 - a_1)(x_1 - a_2) \cdots (x_1 - a_d).$$

Note that the only variable occurring in this polynomial is x_1, and the degree of the polynomial is d.

This polynomial evaluates to zero if and only if $x_1 \in \{a_1, \ldots, a_d\}$. The other variables $x_2, \ldots, x_n$ can be set to arbitrary values in S. Therefore, the number of n-tuples $(r_1, \ldots, r_n) \in S^n$ with $p(r_1, \ldots, r_n) = 0$ is exactly

$$\underbrace{d}_{\text{choices for } r_1} \times \underbrace{|S|}_{\text{choices for } r_2} \times \cdots \times \underbrace{|S|}_{\text{choices for } r_n},$$

which is $d \cdot |S|^{n-1}$.

Solution 4: The Permanent and the Determinant

(a) By the definition of the determinant and by linearity of expectation, we have

$$\mathbf{E}[\det(B)] = \sum_{\pi \in \mathcal{S}_n} \operatorname{sign}(\pi)\, \mathbf{E}\Big[b_{1,\pi(1)} b_{2,\pi(2)} \dots b_{n,\pi(n)}\Big].$$

Now let $Z \subseteq \mathcal{S}_n$ be defined as

$$Z := \{\pi \in \mathcal{S}_n | a_{1,\pi(1)} a_{2,\pi(2)} \dots a_{n,\pi(n)} = 1\},$$

that is the set of transversals that do not contain a zero element of A. We then have that

$$\mathbf{E}[\det(B)] = \sum_{\pi \in Z} \operatorname{sign}(\pi)\, \mathbf{E}\Big[\epsilon_{1,\pi(1)} \epsilon_{2,\pi(2)} \dots \epsilon_{n,\pi(n)}\Big],$$

and by independence of the $\epsilon_{i,j}$,

$$\mathbf{E}[\det(B)] = \sum_{\pi \in Z} \operatorname{sign}(\pi)\, \mathbf{E}\Big[\epsilon_{1,\pi(1)}\Big]\, \mathbf{E}\Big[\epsilon_{2,\pi(2)}\Big] \dots \mathbf{E}\Big[\epsilon_{n,\pi(n)}\Big] = 0,$$

as each expectation is zero.

(b) This calculation is more involved. We first note that by definition (and reusing the set Z from (a)),

$$\mathbf{E}\Big[(\det(B))^2\Big] = \mathbf{E}\left[\left(\sum_{\pi \in Z} \operatorname{sign}(\pi) \epsilon_{1,\pi(1)} \epsilon_{2,\pi(2)} \dots \epsilon_{n,\pi(n)}\right)^2\right].$$

Expanding the multiplication and applying linearity of expectation yields

$$\mathbf{E}\Big[(\det(B))^2\Big] = \sum_{\pi_1, \pi_2 \in Z} \operatorname{sign}(\pi_1) \cdot \operatorname{sign}(\pi_2) \cdot \mathbf{E}\Big[\epsilon_{1,\pi_1(1)} \epsilon_{1,\pi_2(1)} \epsilon_{2,\pi_1(2)} \epsilon_{2,\pi_2(1)} \dots \epsilon_{n,\pi_1(n)} \epsilon_{n,\pi_2(n)}\Big].$$

Now we start disentangling dependencies. First of all, since the $\epsilon_{i,j}$ are independent from one another, we can separate the expectation as

$$\mathbf{E}\Big[(\det(B))^2\Big] = \sum_{\pi_1, \pi_2 \in Z} \operatorname{sign}(\pi_1) \cdot \operatorname{sign}(\pi_2) \cdot \mathbf{E}\Big[\epsilon_{1,\pi_1(1)} \epsilon_{1,\pi_2(1)}\Big]\, \mathbf{E}\Big[\epsilon_{2,\pi_1(2)} \epsilon_{2,\pi_2(1)}\Big] \dots \mathbf{E}\Big[\epsilon_{n,\pi_1(n)} \epsilon_{n,\pi_2(n)}\Big].$$

Now we observe that

$$\mathbf{E}\Big[\epsilon_{i,j} \epsilon_{i,k}\Big] = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise.} \end{cases}$$

For that reason, all the summands with $\pi_1 \neq \pi_2$ have at least one zero factor in the product and thus vanish. Remaining are the summands where the permutations are equal and thus

$$\mathbf{E}\Big[(\det(B))^2\Big] = \sum_{\pi \in Z} \operatorname{sign}^2(\pi) \cdot\, \mathbf{E}\Big[\epsilon_{1,\pi(1)}^2\Big]\, \mathbf{E}\Big[\epsilon_{2,\pi(2)}^2\Big] \dots \mathbf{E}\Big[\epsilon_{n,\pi(n)}^2\Big] = |Z|.$$

On the other hand, obviously

$$\operatorname{per}(A) = \sum_{\pi \in Z} 1 = |Z|,$$

which establishes the claim.

Solution 5: Existence vs. Explicit Construction of Matchings

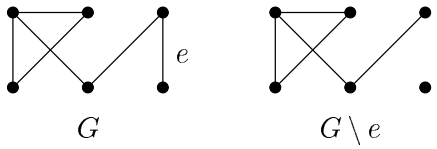
We are given an algorithm A for testing the existence of a perfect matching in a given graph, with running time at most $T(n)$ for any n-vertex graph.

(a) We want to find a perfect matching of a graph G by using repeated calls to algorithm A (supposed that G has a perfect matching).

First we call $A(G)$. If it says "No", G has no perfect matching. Done. If the algorithm says "Yes" G has a perfect matching. We have to find one.

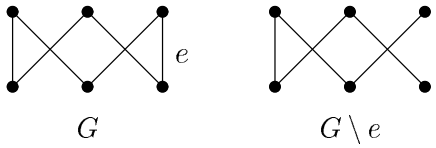
Choose an arbitrary edge e of the graph G. Now we are going to check whether e is part of *every* perfect matching of G. To do that, consider deleting e from G. Denote by $G \setminus e$ the result of the deletion. Then we call $A(G \setminus e)$. We have two cases.

Case 1. If the algorithm says "No", then e is a part of every perfect matching of G (since G contains a perfect matching but $G \setminus e$ does not).



In this case we keep e as an edge of the perfect matching that we will output later, and continue with the remaining graph, i.e., the graph obtained by removing the vertices incident to e (because they are already matched by e).

Case 2. In case the algorithm says "Yes", $G \setminus e$ contains a perfect matching, which is also a perfect matching of G.



Therefore we continue with $G \setminus e$ to find a perfect matching in $G \setminus e$.

This is the idea of our procedure, as given by the following Algorithm 1 in pseudocode.

```
Algorithm 1:  Finding a Perfect Matching in a Graph
    Input:   a graph G = (V, E)
    Output:  a perfect matching M of G if exists and 'No' if not
    IF A(G) = 'No' THEN
       RETURN 'No'
    ELSE
       M ← ∅
       WHILE M is not a perfect matching of G DO
          e ← an arbitrary edge in E
```

```
        IF A(G \ e) = 'No' THEN
            M ← M ∪ {e}
            G ← the graph obtained by removing the vertices incident to e
        ELSE
            G ← G \ e
        END
    END
    RETURN M
END
```

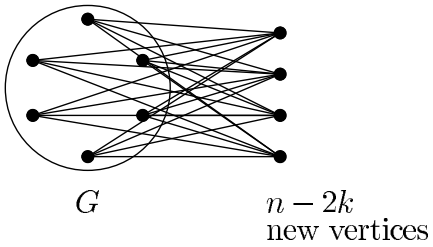
The correctness of the algorithm follows from the discussion above. What is the running time? One call to $A(G)$ takes $T(n)$ time. Then, we will potentially enter the while-loop. In each iteration of the loop at least one edge is removed from the graph and the number of vertices in the graph is always at most n. The time we need for the deletion of an edge or a vertex depends on a data structure used in the test A, so let us denote it by $t(n)$, when we are dealing with a graph with n vertices. Then the worst-case total running time is at most $T(n) + O(m \cdot (t(n) + T(n))) = O(mT(n))$. Here, $m := |E|$ as usual and we safely assume $t(n) < T(n)$.

If we use the algorithm from the lecture as A, we get a running time $O(n^{4.376})$.

(b) How can the above algorithm be used for finding a maximum matching in a given graph?

Let G be a graph with n vertices. The basic step is to decide whether G has a matching of size k (i.e., consisting of k edges). With this subroutine, we search for the maximum k by performing the binary search on $\{0..\lfloor n/2 \rfloor\}$ (note that if G contains a matching of size k then it also contains a matching of smaller size). Therefore the overall running time will be $O(\log n)$ multiplied by the time needed to decide whether G contains a matching of a given size.

Let us fix $k \in \{0..\lfloor n/2 \rfloor\}$. To decide whether G has a matching of size k, we construct an auxiliary graph G^* from G as follows. The vertex set of G^* is the vertex set of G plus additional $n - 2k$ vertices. The edge set of G^* is the edge set of G plus the following edges: we connect every vertex of G to each of the new vertices by an edge. This is our construction of G^*:



Lemma 1. *G has a matching of size k if and only if G^* has a perfect matching.*

Proof. Assume that G has a matching of size k. Take such a matching. There are $n - 2k$ vertices in G which are not incident to any edge of the matching. Then, in G^* these vertices can be matched with the additional vertices, which gives a perfect

matching of G^*. Conversely, if we have a perfect matching of G^*, by removing the additional $n-2k$ vertices and the edges incident to them we obtain a matching in G of size k.

In this way, deciding if G has a matching of size k reduces to deciding whether G^* has a perfect matching. We observe that G^* has $2n-2k$ vertices and $m+n(n-2k)$ edges, which are at most $2n$ and $m+n^2 = O(n^2)$ respectively. Therefore, running the above binary search to find the appropriate maximum k and applying the result of (a) to finally find a matching of size k we can find a maximum matching of G in $O(\log(n)T(2n) + n^2T(2n)) = O(n^2T(2n))$ time. □