ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

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

Algorithms, Probability, and Computing **Solutions for SPA 2** **HS17**

Solution 1 — 15 points
Random matchings

(a) Let $k \in \mathbb{N}$ be such that 2^k is the smallest power of two that is at least N, i.e., $2^{k-1} < N \leq 2^k$. Take k random bits from the stream and interpret the sequence of k random bits as an integer i, written in its binary representation. Because the stream consisted of random bits, the number $i+1$ is uniformly distributed in the set $\{1, \ldots, 2^k\}$. If $i+1 \leq N$, then $i+1$ is uniformly distributed in $\{1, \ldots, N\}$ because for every $j \in \{1, \ldots, N\}$ it holds that

$$\Pr[j = i+1 \mid i+1 \leq N] = \frac{\Pr[j = i+1 \text{ and } i+1 \leq N]}{\Pr[i+1 \leq N]} = \frac{1/2^k}{N/2^k} = \frac{1}{N}.$$

To sample the required number we would repeat the above process, always sampling a new integer $i \in \{1, \ldots, 2^k\}$ by using k new random bits from the stream until $i+1 \in \{1, \ldots, N\}$. The success probability of one such sampling is $p := \frac{N}{2^k} > \frac{2^{k-1}}{2^k} = \frac{1}{2}$ and different repetitions are independent of each other. The number of repetitions needed until succeeding is geometrically distributed with parameter p. Therefore in expectation after $\frac{1}{p} < 2$ repetitions we will succeed. We conclude that the expected number of bits used is at most $2 \cdot k = O(\log N)$.

(b) We will first describe our algorithm SampleMatching for the problem and then prove its correctness and show that it satisfies the runtime requirement and that it uses only the required number of random bits. Given a graph $G = (V, E)$ and an edge $e \in E$ we let G_e denote be the graph attained from G by removing both endpoints of e and all their adjacent edges from G. For a vertex $v \in V$ we denote by $\delta(v) \subseteq E$ the set of all edges adjacent to v. We let Oracle(G) denote the oracle function that takes a graph and returns the number of perfect matchings in G. Our algorithm SampleMatching for the problem is defined below.

```
Input: A nonempty simple graph $G = (V, E)$.
Output: A uniformly random perfect matching $M \subseteq E$ in $G$ or $\emptyset$ if there is no perfect matching.
SampleMatching(G):
    1. Check with the oracle that G has at least one perfect matching and if not, return $\emptyset$.
    2. If $|V| = 2$, return the single edge in G.
    3. Let $v \in V$ be an arbitrary vertex and fix an ordering of $\delta(v) = \{e_1, \ldots, e_s\}$.
    4. For every $i = 1, \ldots, s$ let $N_i :=$ Oracle$(G_{e_i})$ and let $N := \sum_{i=1}^{s} N_i$.
    5. Using (a) choose a uniformly random integer $k \in \{1, \ldots, N\}$.
    6. Let j be the least index so that $\sum_{i=1}^{j} N_i \geq k$.
    7. Return $\{e_j\} \cup$ SampleMatching$(G_{e_j})$.
```

Correctness proof. We show that SAMPLEMATCHING really outputs a uniformly random perfect matching given that there is at least one such matching. If G has an odd number of vertices or no perfect matching is recognized in step 1 and is correctly handled so we need to prove correctness only for graphs with at least one perfect matching.

We proceed by induction on n. The base case $n = 2$ is handled correctly in step 2 since there is a unique perfect matching, the single edge. Assume now that the algorithm is correct for all graphs with at most $n - 2$ vertices, n even, and consider a graph G with n vertices. Fix also some perfect matching M in G. We observe first that the number N computed in step 4 is the number of perfect matchings in G. This is because every perfect matching of G contains exactly one of the edges $e \in \delta(v)$ and because every perfect matching M' that contains some edge $e \in \delta(v)$ has the property that $M' \setminus \{e\}$ is a perfect matching in G_e.

Let ℓ be such that M contains the edge $e_\ell \in \delta(v)$. For the algorithm to output M it has to be that $\ell = j$ and that the recursive call of step 7 returns the matching $M \setminus \{e_\ell\}$ when called on the graph G_{e_ℓ}. Notice that $\Pr[j = \ell] = \frac{N_\ell}{N}$ since there are N_ℓ values $k \in \{1, \dots, N\}$ for which ℓ is the least index satisfying the condition in step 6. By induction we have

$$\Pr[\textsc{SampleMatching}(G_{e_j}) = M \setminus \{e_\ell\} \mid j = \ell] = \frac{1}{N_\ell}.$$

Therefore the probability that the algorithm returns M is $\frac{N_\ell}{N} \cdot \frac{1}{N_\ell} = \frac{1}{N}$ which is uniform across all perfect matchings. This concludes the correctness proof.

Runtime analysis. In steps 1-3 we do one call to the oracle that uses time $T(n)$ and additionally we spend only $\mathrm{poly}(n)$ time. In step 4 we do at most $n-1$ calls to the oracle, each taking time $T(n)$. Because of part (a) step 5 takes time $O(\log N)$ in expectation which is $\mathrm{poly}(n)$ because N is certainly at most $n! \leq n^n$. Step 6 takes also $\mathrm{poly}(n)$ time. Notice that when we recurse in step 7 we reduce the number of vertices by 2 so the depth of the recursion is $O(n)$. Therefore the total number of oracle calls is at most $(n-1) \cdot O(n) = O(n^2)$ and the total expected runtime is also bounded by $O(T(n)\mathrm{poly}(n))$ as required.

Random bits. We already argued that $N \leq n^n$ which implies that the number of random bits we use in step 5 is in expectation $O(\log N) = O(n \log n)$. We do such sampling $O(n)$ times across the recursive calls so the total number of random bits we use is $O(n^2 \log n)$.

(c) The key ingredient to solving this problem is to realize that in a planar graph there always exists a vertex whose degree is at most 5. This is because, as we have seen, the number of edges in a planar graph with n vertices is at most $3n - 6$. Therefore the average degree is at most $\frac{2(3n-6)}{n} < 6$ which implies the claim. We change the algorithm from part (b) by choosing the vertex v in step 3 as a vertex whose degree is at most 5. The correctness of the algorithm stays unchanged.

Runtime analysis Since in every recursive step there are at most $1+5 = 6$ oracle calls, the total number of oracle calls across the algorithm execution is $O(n)$. Because there is always a vertex of degree at most 5, the number of perfect matchings in a planar graph is at most $5^{n/2}$. This means that N in step 4 of the algorithm is upper bounded by $5^{n/2}$ and the expected runtime of steps 4-6 is therefore at most $T(n) + O(\log N) = 5T(n) + O(n)$. Note also that steps 1-3 also take only linear time since planar graphs have only linearly many edges and in particular the vertex of degree at most 5 can be found in linear time.

If we let $t(n)$ denote the expected runtime of the modified algorithm SampleMatching we have deduced that $t(n)$ satisfies the recurrence

$$t(n) \le 6T(n) + O(n) + t(n-2).$$

More concretely let c be a constant such that for $n \ge C$, for some large enough constant C, it holds that

$$t(n) \le 6T(n) + cn + t(n-2).$$

Let us prove by induction that $t(n) \le dnT(n)$ for some constant d when $n \ge C$. We don't need to prove the cases $n < C$ since these can be solved in constant time by enumerating all matchings (because C is constant). Also the base case $n = C$ be made to hold by choosing d large enough. For the inductive step we use the recurrence formula together with the induction assumption to conclude that

$$\begin{aligned} t(n) &\le 6T(n) + cn + d(n-2)T(n-2) \\ &\le 6T(n) + cn + d(n-2)T(n) \\ &= 6T(n) + cn - 2dT(n) + dnT(n) \\ &\le dnT(n). \end{aligned}$$

Above $T(n-2) \le T(n)$ since $T(n) \in \Omega(n)$. The last step also follows by choosing d large enough because $T(n) \in \Omega(n)$ then implies that $5T(n) + cn - 2dT(n) < 0$.

Random bits. In one recursive step we use by (a) and our previous remarks at most $O(\log 5^{n/2}) = O(n)$ random bits in expectation. Since there are at most n recursive steps, the total number of random bits is at most $O(n)$.

Solution 2 — 15 points
Small exercises in linear programming

In a basic feasible solution n linearly independent constraints have to hold with equality and the remaining constraints need to be satisfied.

(a) Every $x \in \{0,1\}^n$ is a basic feasible solution since n inequalities hold with equality and they are clearly linearly independent (the identity matrix is non-singular). These are all basic feasible solutions since in order to get n equalities we have to set for every $i \in [n]$ either $x_i = 0$ or $x_i = 1$. Let $I \subseteq [n]$ be the set of all indices $i \in [n]$ so that $c_i < 0$. Then clearly an optimum basic feasible solution $\tilde{x}$ is given by setting $\tilde{x}_i = 1$ if $i \in I$ and $\tilde{x}_i = 0$ otherwise.

(b) Any basic feasible solution has to satisfy the equality $\mathbf{1}^T x = 1$ and $n-1$ coordinates of x have to be zero. Therefore the remaining coordinate has to be 1. The basic feasible solutions are hence all the positive unit coordinate vectors. Let $i = \arg\min_{j \in [n]} c_j$. Clearly the objective function is minimized for the i'th positive unit coordinate vector.

(c) If the constraint $\mathbf{1}^T x \leq k$ does not hold with equality, then we are back to case (a) with the additional constraint that x should have at most $k-1$ coordinates set to 1, i.e., such basic feasible solutions form the set $\{x \in \{0,1\}^n \,|\, \mathbf{1}^T x \leq k-1\}$. If $\mathbf{1}^T x = k$, then in any basic feasible solution $n-1$ additional inequalities have to hold as equalities. Therefore $n-1$ of the coordinates in x are either 0 or 1 and only one coordinate is allowed to be fractional. But since $\mathbf{1}^T x = k$ and since k is an integer, the last coordinate has to be also either 0 or 1. All in all we get that the set of basic feasible solutions is $\{x \in \{0,1\}^n \mid \mathbf{1}^T x \leq k\}$.

Finding the basic feasible solutions with the largest objective function value is equivalent to finding the subset of entries of $\mathbf{c}$ of size at most k that sum to the largest value. We consider the k largest positive entries in $\mathbf{c}$, or fewer if there are not k positive entries, and set the corresponding coordinates in a vector $\tilde{x}$ to 1 and the remaining entries in $\tilde{x}$ to 0. Then clearly $\tilde{x}$ is an optimum basic feasible solution.

(d) For $j \in [m]$ let $\mathbf{a}_j^T$ denote the j'th row of $\mathbf{A}$. We introduce a new variable vector $\mathbf{y}$ with n components. Consider the following optimization problems.

$$\min_{x,y} \mathbf{1}^T y \quad \text{subject to} \quad \begin{cases} y_i \geq |x_i| & \forall i \in [n] \\ \|\mathbf{A}x - \mathbf{b}\|_\infty \leq 1. \end{cases} \tag{1}$$

$$\Leftrightarrow \quad \min_{x,y} \mathbf{1}^T y \quad \text{subject to} \quad \begin{cases} y_i \geq x_i \text{ and } y_i \geq -x_i & \forall i \in [n] \\ |\mathbf{a}_j^T x - \mathbf{b}_j| \leq 1 & \forall j \in [m] \end{cases}$$

$$\Leftrightarrow \quad \min_{x,y} \mathbf{1}^T y \quad \text{subject to} \quad \begin{cases} y_i \geq x_i \text{ and } y_i \geq -x_i & \forall i \in [n] \\ -1 \leq \mathbf{a}_j^T x - \mathbf{b}_j \leq 1 & \forall j \in [m] \end{cases}$$

$$\Leftrightarrow \quad \min_{x,y} \mathbf{1}^T y \quad \text{subject to } y \geq x \text{ and } y \geq -x \text{ and } -\mathbf{1} \leq \mathbf{A}x - \mathbf{b} \leq \mathbf{1}. \tag{2}$$

To get from (1) to the equivalent problem (2) we have used the definition of the max-norm and the toolbox of transformations discussed in the lecture. It is also not hard to see directly that the individual steps are equivalent. The optimization problem (2) is a linear program so it remains to argue why (1) is equivalent to the optimization problem

$$\min \|x\|_1 \text{ subject to } \|\mathbf{A}x - \mathbf{b}\|_\infty \leq 1 \tag{3}$$

given in the problem description. We show that given a feasible solution to either of the problems we can construct a feasible solution to the other problem with at least as good objective function value. If $\hat{x}$ is a feasible solution to (3), then we can find a feasible solution to (1) of equal objective function value: simply set $\hat{y}$ so that $\hat{y}_i = |\hat{x}_i|$ for all $i \in [n]$. Then $(\hat{x}, \hat{y})$ is a feasible solution to (1) with objective function value

$$\mathbf{1}^T \hat{y} = \sum_{i=1}^n \hat{y}_i = |\hat{x}_i| = \|x\|_1.$$

For the other direction let $(\hat{x}, \hat{y})$ be a feasible solution to (1). Then $\hat{x}$ is a feasible solution to (3) and the objective function in (3) is equal to

$$\|\hat{x}\|_1 = \sum_{i=1}^n |\hat{x}_i| \leq \sum_{i=1}^n \hat{y}_i = \mathbf{1}^T \hat{y}$$

and therefore (3) has an optimum of at least as good value as (1). This concludes the proof.

Solution 3 <span style="float:right">15 points</span>

Farkas lemma for s-t paths

You might notice that there is a very short direct proof of Lemma 1 in (c). Even though (a) and (b) are longer than (c) the main point is not just proving Lemma 1 but that of practicing the manipulation of systems of inequalities and seeing that the statement is simply one of many instantiations of Farkas lemma.

(a) The constrains can be interpreted as flow conservation constraints so that for every arc $e = (u, v)$ the variable x_e describes the amount of flow from u to v. Then for $v \in V \backslash \{s, t\}$ the constraint

$$\sum_{e \in \delta(v)^+} x_e - \sum_{e \in \delta(v)^-} x_e = 0$$

is saying that the amount of flow incoming to v is the same as the amount of flow going out from v. For s the corresponding constraint is saying that there is more outgoing flow than incoming flow, i.e., s is a source of flow, and for t the constraints dictate that t is a sink because there is more incoming flow than outgoing flow. It is possible to argue using this flow interpretation but we do not do the proof in such a way.

If there is a directed s-t path P, we can set $x_e = 1$ for every arc e on the path P and $x_e = 0$ for every arc not on P. It is then easy to see that this constitutes a solution to the system by considering separately constraints corresponding to the vertices that are internal vertices on the path P, the endpoints s, t of P and the vertices not on P.

To show the other direction let $\hat{x} \in \mathbb{R}^A$ be a solution to the system given in the exercise description and assume that among all solutions $\hat{x}$ minimizes the total weight $\mathbf{1}^T\hat{x}$. Let $D' = (V, A')$ be a subgraph of D so that A' contains exactly those arcs from A that have positive weight in $\hat{x}$. Observe that D' is acyclic as otherwise we could reduce the weight on all edges of a cycle by some small amount $w > 0$ which would result in a feasible solution with smaller total weight than $\hat{x}$, a contradiction. The weight w can be taken to be the minimum weight of any arc on the cycle.

By considering the constraints we observe that s has at least one outgoing arc in D' and t has at least one incoming arc in D'. Any vertex $v \in V \setminus \{s, t\}$ that is adjacent to an arc in D' is adjacent to at least one outgoing and at least one incoming arc. From these properties and from the acyclicity of D' it then follows that if we start a directed walk from s in D' we will necessarily reach t eventually which proves the existence of an s-t path since D' is a subgraph of D.

(b) Let us write the equality constrains in a matrix form $\mathbf{B}\mathbf{x} = \mathbf{b}$ where $\mathbf{B} \in \mathbb{R}^{V \times E}$, and $\mathbf{b} \in \mathbb{R}^V$. Then for every vertex $v \in V$ and every arc $e = (u, w) \in A$ we have that

$$\mathbf{B}_{v,e} = \begin{cases} 1 & \text{if } u = v \\ -1 & \text{if } w = v \\ 0 & \text{otherwise.} \end{cases} \quad \text{and} \quad \mathbf{b}_v = \begin{cases} 0 & \text{if } v \in V \setminus \{s, t\} \\ 1 & \text{if } v = s \\ -1 & \text{if } v = t. \end{cases}$$

In other words, B is a matrix whose rows are indexed by the vertices V and whose columns are indexed by the arcs A. In the column corresponding to the arc $e = (u, w)$ there is a 1 at the row corresponding to u and a -1 at the row corresponding to w and remaining entries in the column are 0.

Using Farkas lemma II from Lemma 6.5 of the lecture notes we know that exactly one of the two systems $\{\mathsf{B}\mathsf{x} = \mathsf{b}, \mathsf{x} \geq \mathbf{0}\}$ and $\{\mathsf{B}^\mathsf{T}\mathsf{y} \geq 0, \mathsf{b}^\mathsf{T}\mathsf{y} < 0\}$ has a solution where the latter system the vector y contains variables indexed by the vertices. To prove Lemma 1 we therefore have to show that the second system has a solution if and only if there exists a strong s-t cut. We can write out the second system as

$$\begin{aligned} \forall e = (u, w) \in A: \quad & \mathsf{y}_u - \mathsf{y}_w \geq 0 \text{ and} \\ & \mathsf{y}_s - \mathsf{y}_t < 0. \end{aligned} \tag{4}$$

Assume first that S is a strong s-t cut. Then by setting $y_v = 0$ for $v \in S$ and $y_v = 1$ for $v \in V \backslash S$ satisfies the constraints in (4) in particular because the constraints corresponding to the arcs in the cut $C(S)$ are satisfied due to the strong s-t cut property, the constraints corresponding to edges within S and $V \setminus S$ hold with equality and for the last constraint we have $\mathsf{y}_s - \mathsf{y}_t = 0 - 1 = -1 < 0$ as it should.

To show the other direction assume that there exists a solution $\hat{\mathsf{y}}$ to (4). Define $S := \{v \in V \,|\, \hat{\mathsf{y}}_v \leq \hat{\mathsf{y}}_s\}$ as the set of vertices that are assigned a value at most $\hat{\mathsf{y}}_s$ is $\hat{\mathsf{y}}$. We claim that S is a strong s-t cut. Firstly, by definition $s \in S$ and by the second inequality of (4) we have $t \notin S$ so S is an s-t cut. Also from the definition of S we observe that for every $u \in S$ and every $w \in V \setminus S$ it holds that $\hat{\mathsf{y}}_u - \hat{\mathsf{y}}_w < 0$. Since $\hat{\mathsf{y}}$ is a solution to (4) this implies that there can be no arcs $(u, w) \in A$ with $u \in S, w \in V \setminus S$ and therefore S is a strong s-t cut. This concludes the proof.

(c) If there is a directed s-t path P, then there can not exist a strong s-t cut since for any set $S \subseteq V$ with $s \in S$ and $t \notin S$ there is at least one arc of P that goes from S to $V \setminus S$. If there is no directed s-t path, we let S be the set of all vertices reachable from S by a directed path. Then S is a strong s-t cut because no vertex of $V \setminus S$ is reachable from a vertex of S.