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 KW45 — HS17

Solution 1: Existence vs. Explicit Construction of Matchings

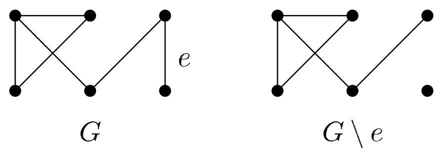
We are given an algorithm A for testing the existence of a perfect matching in a given graph, with running time at most $T(n)$ for any n-vertex graph.

(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.

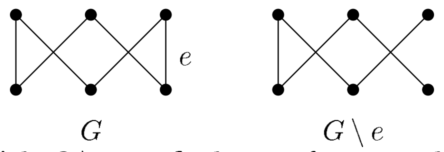
Choose an arbitrary edge e of the graph G. Now we are going to check whether e is part of *every* perfect matching of G. To do that, consider deleting e from G. Denote by $G \setminus e$ the result of the deletion. Then we call $A(G \setminus e)$. We have two cases.

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
       $M \leftarrow \emptyset$
       WHILE $M$ is not a perfect matching of $G$ DO
          $e \leftarrow$ an arbitrary edge in $E$
          IF $A(G \setminus e)$ = 'No' THEN
             $M \leftarrow M \cup \{e\}$
             $G \leftarrow$ the graph obtained by removing the vertices incident to $e$
          ELSE
             $G \leftarrow G \setminus e$
          END
       END
       RETURN $M$
    END
```

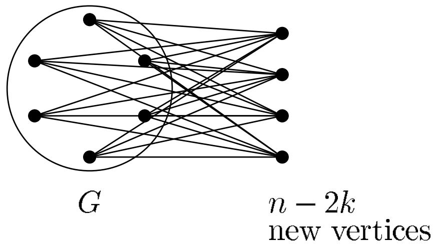
The correctness of the algorithm follows from the discussion above. What is the running time? One call to $A(G)$ takes $T(n)$ time. Then, we will potentially enter the while-loop. In each iteration of the loop at least one edge is removed from the graph and the number of vertices in the graph is always 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. □

Solution 2: Approximating the Minimum Cut

(a) In the lecture, it was pointed out that already the standard BasicMinCut algorithm is implemented using (among other things), an array containing all the degrees of the vertices currently in the graph — for the purpose of efficiently selecting an edge u.a.r. for contraction. The only adaptation needed now is that after each contraction, we scan this array and maintain a global minimal degree ever seen. It is clear that this can be done in linear time per step and thus in $\mathcal{O}(n^2)$ time in total.

(b) We know from the lecture that contractions can only increase the size of a minimum cut but never decrease it. Since the edges incident to any one vertex always *form* a cut, each of the numbers that we could report in this algorithm corresponds to some cut in the original graph, which readily implies the claim.

(c) For $n \leq 2$, the claim is empty. Let us now look at $n > 2$, let us fix a graph G of size n and a cut C of size $\mu(G)$. What is the probability that the event we are looking for occurs? There are two cases. Either, G contains a vertex of degree less than $(1+\alpha)\mu(G)$. In that case, no matter what the further recursion would yield, we will always return a number at most that degree and thus the probability is 1. Or, all vertices in G have degree at least $(1+\alpha)\mu(G)$. Then, there are at least $(1+\alpha)\mu(G)\cdot\frac{n}{2}$ edges in the graph and the probability that we contract one from C is thus bounded by $\frac{2}{(1+\alpha)n}$. But then with the complement of this probability, the new graph G/e will still have a cut of size $\mu(G)$ and by induction, the claim follows.

As for the calculation (which was not required), since $p_\alpha(2) = 1$, we just compute (for n sufficiently large)

$$\begin{aligned}
p_\alpha(n) &\geq \prod_{i=3}^{n} \left(1 - \frac{2}{(1+\alpha)i}\right) \\
&= \exp\left(\sum_{i=3}^{n} \ln\left(1 - \frac{2}{(1+\alpha)i}\right)\right) \\
&\overset{(1)}{\geq} \exp\left(\sum_{i=3}^{n} \left(-\frac{2}{(1+\alpha)i} - \left(\frac{2}{(1+\alpha)i}\right)^2\right)\right) \\
&\overset{(2)}{\geq} \exp\left(-\frac{2}{1+\alpha}\left(H_n - \frac{3}{2}\right) - \frac{2\pi^2}{6}\right) \\
&\overset{(3)}{\geq} \exp\left(-\frac{2}{1+\alpha}\ln n - \frac{2\pi^2}{6}\right) \\
&= \Omega(n^{\frac{2}{1+\alpha}}),
\end{aligned}$$

where (1) uses the inequality $1 - x \geq e^{-x-x^2}$ for $x \leq 0.68$ and $\frac{2}{(1+\alpha)i} < 2/3 < 0.68$ for all $\alpha > 0$ and $i \geq 3$; where (2) uses that $\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6}$ and $\frac{2}{1+\alpha} < 2$ for $\alpha > 0$; and where (3) uses $H_n - 3/2 < \ln n$ (the difference between $\ln n$ and H_n approaches the Euler-Mascheroni constant γ, which is about 0.577).

Solution 3: Bounding the Number of Minimum Cuts

Given a graph $G = (V, E)$, let N be the number of minimum cuts in G. We want to show that $N \leq \binom{n}{2}$ where $n := |V|$.

Let $C_1, \ldots, C_N$ be the minimum cuts in G. Then, we know that for each $i \in \{1..N\}$

$$\Pr[C_i \text{ is found by Karger's algorithm } \textsc{BasicMinCut}(G)] \geq \frac{1}{\binom{n}{2}}.$$

(See Section 3.3 in the lecture notes.)

Now, we observe that for each two distinct indices $i, j \in \{1..N\}$ the events "C_i is found by $\textsc{BasicMinCut}(G)$" and "C_j is found by $\textsc{BasicMinCut}(G)$" are disjoint (i.e. they never happen at the same time). To see this, consider the graph obtained at the termination of $\textsc{BasicMinCut}(G)$. It has only two vertices and these vertices (together with the "contraction history") uniquely determine a partition of the vertex set V. So we cannot get two different minimum cuts from one execution of the algorithm. Therefore, it follows that

$$\begin{aligned}&\Pr[\text{a minimum cut is found by } \textsc{BasicMinCut}(G)]\\ &= \sum_{i=1}^{N} \Pr[C_i \text{ is found by } \textsc{BasicMinCut}(G)] \geq \frac{N}{\binom{n}{2}}.\end{aligned}$$

Since $\Pr[\text{a minimum cut is found by } \textsc{BasicMinCut}(G)] \leq 1$ (because it is a probability!), we obtain $N \leq \binom{n}{2}$.

Solution 4: Submodularity for Min-Cut

(a) Observe that there are three (potentially empty) sets of edges $e = \{u, v\}$ that are important in this scenario:

$$\begin{aligned} E_1 &:= \left\{e = \{u,v\} \mid u \in A \cap B \text{ and } v \in V \setminus (A \cup B)\right\} \\ E_2 &:= \left\{e = \{u,v\} \mid u \in (A \setminus B) \cup (B \setminus A) \text{ and } v \in V \setminus (A \cup B)\right\} \\ E_3 &:= \left\{e = \{u,v\} \mid u \in A \cap B \text{ and } v \in (A \setminus B) \cup (B \setminus A)\right\}. \end{aligned}$$

It is easy to see that $f(A \cap B) + f(A \cup B) = |E_1| + |E_3| + |E_1| + |E_2|$, while $f(A) + f(B) \geq 2|E_1| + |E_2| + |E_3|$, the last inequality holding because some edges between $A \setminus B$ and $B \setminus A$ may appear both in $f(A)$ and $f(B)$.

(b) Let k be the size of a minimum cut. Then by (a) we get $f(A \cap B) + f(A \cup B) \leq f(A) + f(B) = k + k$, which together with $f(C) \geq k$ for any set $C \neq \emptyset, V$, implies $f(A \cap B) = f(A \cup B) = k$.

(c) Suppose towards a contradiction that $S \neq S'$, with $S, S' \subset V$ and $s \in S \cap S'$ are such that $C(S)$ and $C(S')$ are both minimum cuts and $|S| = |S'|$ is minimal. Note that as $t \notin S \cup S'$ because they are both cuts, we must have $S \cup S' \neq V$, and so we can now directly apply part (b) to obtain that $S \cap S'$ is a minimum cut with $|S \cap S'| < |S| = |S'|$, a contradiction.