University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour, M. Fuchs, A. Mályusz

Algorithm Theory
Sample Solution Exercise Sheet 10

Due: Friday, 10th of January, 2025, 10:00 am

Exercise 1: Sinkless Orientation *(7 Points)*

(a) Let $B = (U \cup V, E)$ be a bipartite graph and assume that for every $A \subseteq U$, we have $|N(A)| \geq |A|$. Show that this implies that there exists a matching of size $|U|$ in B (i.e., a matching that matches every node in U). *(3 Points)*

(b) Let $G = (V, E)$ be a graph with minimum degree at least two (i.e., each node has at least two incident edges). Use the prior statement to show that there is a way to orient the edges of G such that each node of G has at least one out-going edge (this is known as a sinkless orientation).

Hint: Sinkless orientation can be seen as matching nodes and edges. *(3 Points)*

(c) Let us now assume that the graph G has minimum degree 4. Show that there exists an orientation in which each node has at least one in-coming and at least one out-going edge.
Hint: Use part (b). *(1 Point)*

Sample Solution

(a) We have the following equivalence (contra-position):

$$\Big[\forall A \subseteq U : |N(A)| \geq |A| \Rightarrow B \text{ has a matching of size } |U|\Big]$$
$$\text{if and only if} \quad \Big[B \text{ has } \textbf{no} \text{ matching of size } |U| \Rightarrow \exists A \subseteq U : |N(A)| < |A|\Big]$$

Therefore it suffices to prove the second line. Consider a bipartite graph B with *no* matching of size $|U|$. We construct a flow network from B (as we have seen in the lecture), by connecting a source s to all nodes in U and all nodes in V to a sink t, directing all edges from s to t and setting all edge capacities to one.

Since a maximum flow implies a matching of the same size, the maximum flow in this network must also be smaller than $|U|$. Due to the max-flow-min-cut theorem, the minimum cut $C = (A', B')$ (with $A', B' \subseteq U \cup V \cup \{s, t\}$) has size smaller than $|U|$ as well. For brevity, let $|C|$ denote the size of $C = (A', B')$ (number of edges from A' to B'), i.e., $|C| < |U|$.

W.l.o.g. let A' be the set that contains s. We know A' can not be empty nor can it encompass all nodes (by definition of a cut). Further we know that $A' \neq \{s\}$ and $A' \neq U \cup V \cup \{s\}$ (since those cuts would have size $|U|$ and would thus not be minimum). Therefore $A := A' \cap U$ is not empty.

Let x be the number of edges from s to B'. Since by construction we have one edge from s to each node in U we know that $x = |U \setminus A| = |U| - |A|$ $(*)$. Let z be the number of edges from A to B'. Let y be the number of edges from $A' \setminus (A \cup s) = A' \cap V$ to B'. Then $y = |A' \cap V|$ since by construction each node in V has exactly one edge to the sink t.

Taken together x, y, z represent the size of the cut C, i.e., $x + y + z = |C| < |U|$ $(**)$. Further we know that $|N(A)| \leq y + z$ $(***)$ since $N(A)$ can not contain more nodes then the edges going

from A to B' (which is z), plus the nodes in $A' \cap V$ (which is y). We obtain

$$|N(A)| \overset{(***)}{\leq} y + z \overset{(**)}{<} |U| - x \overset{(*)}{=} |A|$$

(b) One can set up the problem of finding such an orientation as a bipartite matching problem as follows. The bipartite graph has a node v for each node of G (let's call this L) and it has a node e for each edge of G (let's call this R). Nodes v and e are connected in the bipartite graph $(L \cup R, E')$, if v is a node of the edge e in G. A matching in this bipartite graph is a pairing of nodes and edges in G such that each edge of G is assigned to at most one of its two nodes.

If the matching contains (v, e), node v in G orients the edge e as an out-going edge for itself. In order to guarantee the existence of a sinkless orientation, we thus need to show that the constructed bipartite graph has a matching that matches each node v. Note that each node in L has degree at least 2 (because G has minimum degree at least 2) and each node in R has degree exactly 2 (because an edge of G has exactly two incident nodes). Therefore, for a given subset $A \subseteq L$, we have at least $2|A|$ outgoing edges from A to R. And since each node in R has degree two, the $2|A|$ outgoing edges of A go to at least $|A|$ different nodes in R. Hence $|N(A)| \geq |A|$ for every $A \subseteq L$.

(c) After constructing the bipartite graph as before, we split each node $v \in L$ into two nodes v_1 and v_2, which each get at least two of the edges of v (we partition the edges evenly among v_1 and v_2). Minimum degree 4 implies that each node in L in this new bipartite graph has minimum degree 2 and we can thus compute a bipartite matching which matches each $v \in L$. Each node v now has two edges, which it can orient freely.

Exercise 2: Maximality Helps *(5 Points)*

Let $B = (U \cup V, E)$ be a bipartite graph. For finding a maximum matching on bipartite graphs, we would like to speed up the Ford Fulkerson approach. Recall that in the max flow reduction approach, we try to improve our matching by iteratively searching for an augementing path, so what if instead we can find a good bunch of disjoint augementing paths? Indeed in here we consider our bipartite graph B as input (and not its flow network representation) and the following algorithm: We start with an empty matching M, then repeat the following 2 steps until no more augmenting paths with respect to the matching M in B can be found:

1. Search for a ***maximal*** *set of* ***vertex-disjoint shortest*** *augmenting paths* $\{P_1, P_2, \ldots, P_k\}$
2. Update M by flipping every matched edge in $P_1, P_2, ..., P_k$ to unmatched and vice versa.

Finally, return M.

Remark: Maximal above means that no more such paths can be added.

(a) Show that after $O(\sqrt{n})$ repetitions of the 2 steps above, M is a maximum matching of B. *(3 Points)*

(b) Argue that the running time of the algorithm can be $O(m\sqrt{n})$, where as usual m is the number of edges and n is the number of nodes. *(2 Points)*

Sample Solution

(a) We need to show that after $O(\sqrt{n})$ repetitions, no more augmenting paths w.r.t. M exists and by this point we would have a maximum matching. The rough idea is to notice that after one repetition of steps 1 and 2, the length of a shortest augmenting path w.r.t M increases by at least 1. If we can show this, we deduce that after $O(\sqrt{n})$ repetitions of steps 1 and 2, the length of a shortest augmenting path is at least $\sqrt{n}$. Thus, the number of remaining augementing paths is at most $\frac{n}{\sqrt{n}} = \sqrt{n}$, so we can then apply step 2 on a vertex disjoint subset of these $\leq \sqrt{n}$

augementing paths, hence no more augmenting paths exist anymore (*) and the final matching is now maximim.
(*) Because in general any two augmenting paths can only intersect on an unmatched node or on an odd alternating path starting and ending with a matched edge, so even the augementing paths that intersected with any of the $\leq \sqrt{n}$ augementing paths will no longer be augmenting after applying step 2 (moreover, with a bit of extra thought one can also see that no new augementing paths are also created). Hence no more augmenting paths exist in the graph anymore.

It is left to show that after one repetition of steps 1 and 2, the length of a shortest augmenting path increases by at least 1, let P be the *maximal set of vertex-disjoint shortest augmenting paths found in step 1* and let $k > 0$ be their length i.e. the length of a shortest augmenting path in step 1. Moreover, after running steps 1 and 2, assume the length of a shortest augmenting path has now decreased i.e. there exists an augementing path p' of length equal to $k' < k$. Thus, if p' was vertex disjoint with the all paths in P', then k' should've been the length of a shortest augmenting path in step 1 and not k, a contradiction. Thus p' has to intersect with at least one path $p \in P$, and the first time p' intersects p it has to be over an edge in p that was before step 2 a matched edge i.e. before running step 2, imagine looking at p and p' starting with their corresponding unmatched node from U and alternating between matched and unmatched edges until they intersect, then they must intersect on a matched edge, since if they first intersect elsewhere we will get a contradiction (give it a bit of a thought to see it). So now after applying step 2, p' is no longer alternating, because the matched edge it intersected p with is now unmatched and thus p' is not an augmenting path, which is a contradiction.

(b) We need $O(\sqrt{n})$ repetitions of steps 1 and 2 and after that we will spend an extra $O(\sqrt{n})$ steps to finally reach a maximum matching as explained above. Moreover, for step 1 we can find a maximal set of vertex disjoint shortest augmenting paths with respect to M in $O(m)$ time. To do so, we first perform a modified breath first search: starting from each umnatched node in U, we run a BFS search and construct things level by level over alternating edges between matched and unmatched ones, we do this up to length k (where k is the length of a current shortest augmenting path). By now we have collected all of our current shortest augmenting paths in this BFS representation. Now to extract a vertex disjoint maximal set out of your set of shortest augmenting paths that are collected in your BFS, run a DFS over this BFS representation starting from an umatched node in U and find one augmenting path (it is ofcourse a shortest one too), this will be the first augmenting path in you maximal set, then remove all edges adjacent to this path, and startover ie repeat this DFS starting from another unmatched node from U to find another augementing path (which will be vertex disjoint with the prev augmenting path that we added in our maximal set), repeat until we exhausted the whole BFS representation... by then we have found a maximal vertex disjoint set of augamenting paths of length k. Step 2 will also take $O(m)$, Hence, the total running time is $O(m\sqrt{n})$.

NB This is the famous Hocroft-Karp algorithm for solving bipartite maximum matching.

Exercise 3: 2-Claws *(4 Points)*

We are given a bipartite graph $B = (U \cup V, E)$ on two disjoint node sets U and V; each edge connects a node in U to a node in V. In the following, we define a *2-claw* to be a set of three distinct nodes $\{u, x, y\}$ such that $u \in U$, $x, y \in V$ and there is an edge from u to both nodes x and y.
We consider the following maximization problem on the graph B: Find a largest possible set of vertex-disjoint set of 2-claws. In other words, we want to find a largest possible subset of edges such that every node in U is incident to either 0 or 2 of the edges and each node in V is incident to either 0 or 1 of the edges (i.e., each node is either part of one 2-claw or it is not part of any 2-claw at all). We also assume that in the given bipartite graph B, each node in U has at most 3 neighbors in V.

Give a polynomial-time algorithm which computes a maximum set of vertex-disjoint 2-claws.

Hint: Try to reduce the problem to the maximum matching problem in general graphs.

Sample Solution

To solve the problem, we reduce it to a maximum matching problem. Considering the given graph $B = (U \dot{\cup} V, E)$, we construct a graph $B' = (V', E')$ such that $V' := V$ and E' is explained as follows. For any two vertices $b, c \in V$ which are connected to the same vertex $a \in U$ in B, we connect b and c by an edge in B'. In other words, E' is a set of edges which connects any two vertices of V' in B' where the two corresponding vertices in V has a common neighbor in U in graph B. Now we show that the maximum matching in the general graph B' corresponds to a maximum set of vertex-disjoint 2-claws in the given bipartite graph B. To show the correctness of the reduction we need to prove the following claims.

Claim 1: The number of edges in the maximum matching in the graph B' is the maximum number vertex-disjoint of 2-claws in the graph B.

To prove Claim 1, we first prove the following two claims:
Claim 2.1: If there exist a set of k vertex-disjoint 2-claws in B, then there exist at least k vertex-disjoint edges in B'.
Proof. The claim follows directly from the construction of graph B' that we explained above. For any two vertices in V which have a common neighbour in U there exists an edge connecting the two corresponding vertices of V' in B'. Hence for any 2-claw in B there exist an edge in B'. Moreover, since any two 2-claws are vertex-disjoint, all these corresponding edges in B' are also vertex-disjoint (since, $V' = V$). Therefore, there exist at least k vertex-disjoint edges in B'. □

Claim 2.2: If there exist a set of k vertex-disjoint edges in B', then there exist at least k vertex-disjoint 2-claws in B.
Proof. We prove this claim based on the construction of the graph B' and also the fact that the degree of any vertex in U is at most 3. From the construction of B', the k vertex-disjoint edges in E' represent k distinct pairs of vertices in V such that the two vertices of any pair have a common neighbour in U. However, from the problem definition it is not possible to have two distinct pairs of vertices in V having a common vertex in U, since the degree of any vertex in U is at most 3. As a result, these k pairs of vertices belong to k vertex-disjoint 2-claws. □

Proof of Claim 1. Let us assume that $\mathcal{M}$ is the maximum matching in the graph B'. Based on Claim 2.2, we can conclude that there exist at least $|\mathcal{M}|$ vertex-disjoint 2-claws in B. But we also need to show that there does not exist a vertex-disjoint 2-claws set of size of more than $|\mathcal{M}|$.
Let us assume that there exist a set of vertex-disjoint 2-claws of size $|\mathcal{M}| + 1$ in the graph B. Then from the Claim 2.1, there must exist at least $\mathcal{M} + 1$ vertex-disjoint edges in B'. This contradicts our assumption that $\mathcal{M}$ is the maximum matching in B'. Hence the claim. □

We are now only left to show that how to compute a maximum matching in the general graph B'. For this we use the 'Edmond's Blossom Algorithm' (from the lecture notes) as a subroutine. The running time of Edmond's Blossom algorithm to compute the maximum matching in general graphs is $O(mn^2)$, which is polynomial.

Exercise 4: Special Offer for the Holidays *(4 Points)*

To increase its sales a small kiosk has a special offer: If a customer buys two articles whose prices add up to a value which ends with $11, 33, 55, 77$ or 99 cents, he will receive a voucher, worth the corresponding cent value.

Devise an algorithm which computes an optimal strategy for buying a given collection of goods (here only the price of a good matters).

Sample Solution

Assume that you want to buy n items which are given by the set X. We reduce the problem to an instance of maximum weighted bipartite matching. We create a bipartite graph $G = (X_1 \cup X_2, E)$ where one side of the nodes is formed by all nodes which have an odd cent value, i.e.,

$$X_1 = \{x \in X | \text{the cent value of } x \text{ is odd}\}, \tag{1}$$

and the other side is formed by all nodes which have an even cent value, i.e.,

$$X_2 = \{x \in X | \text{the cent value of } x \text{ is even}\}. \tag{2}$$

We add an edge between $x \in X_1$ and $y \in X_2$ with value $t \in \{11, 33, 55, 77, 99\}$ if and only if the cent values of x and y add up to t.
Now, finding a maximum weighted bipartite matching in this graph is equivalent to grouping X into groups of two elements w.r.t. maximizing the value of vouchers one obtains.
The maximum weighted bipartite matching was solved in the lecture which we use as a black box.