Algorithms and Data Structures (WS15/16)

Example Solutions for Units 29 & 30

Problem 1

In graph theory, an orientation of an undirected graph is an assignment of a direction to each edge, turning the initial graph into a directed graph. (Wiki)

Idea: For each edge, think of orientation of $e = (u, v)$ as adding the outdegree by 1 to either u or v. (or think of each edge as a competition game, each vertex as a team, the winner gets outdegree, then we get the same question as in sheet 7(b))

To find orientation in $G = (V, E)$, using max-flow, we construct $G' = (V', E')$ with

1. Create $|E|$ vertices on the left, each corresponds to each edge $e_i \in G$.
2. Create $|V|$ vertices on the right, each corresponds to each vertex $v_i \in G$.
3. Create vertex s and edges connects from s to every vertex on the left, each with edge capacity 1.
4. Create vertex t and edges connects from every vertex v_i on the right to t with edge capacity d_i.
5. Create edges between left and right side, each from vertex $e_i = (v_j, v_k)$ to two vertices v_j and v_k. (No need to bound capacity)

Find max-flow in graph G' from s to t. If the value of max-flow equal to $\Sigma_{i=1}^{n} d_i$ $(= |E|)$, then there exists the orientation that outdegree of vertex v_i becomes d_i for all $i = 1, ..., n$.

The algorithm is efficient because the running time is polynomial on the size of vertices, $O(|V'||E'|^2) = O((n^2)(n^2)^2) = O(n^6)$.

Problem 2

see end of file

Problem 3

Given an algorithm for *minimum-cost perfect matching* for bipartite graph $G = (A \cup B, E)$ with non-negative edge costs $c_e \geq 0$, we want to compute:

- *minimum-cost* ***maximum*** *matching* - W.l.o.g, assume $|A| \geq |B|$, let S be a set of vertices of size $|A| - |B|$. Let $B' = B \cup S$ (disjoint union), so we have $|A| = |B'|$. Construct a complete bipartite graph $G' = (A \cup B', E')$ with same edge cost for any existing edges $(u, v) \in E$ and very large edge cost for any non-existing edges $(u, v) \notin E$. Observe that,

minimum-cost perfect matching of G' corresponds to *minimum-cost maximum matching* of G because the algorithm will try to avoid as many non-existing edges as possible to minimize the total cost, which means it will try to match existing edges as many as possible.

- ***maximum-cost*** *perfect matching* - We can change edge cost $c(e)$ to be $-c(e)+max_{e\in E}\{c(e)\}$. This way, the larger edge cost will turn into smaller edge cost.
- *minimum-cost perfect matching for* ***general edge costs*** - We can change edge cost $c(e)$ to be $c(e) + max_{e\in E}\{|c(e)|\}$ in order to make all edge costs non-negative.

Remark: For the second and the third question, we can add a constant without changing the optimal solution because any perfect matching always have the same size of solution.

Problem 4

see end of file

Problem 5

- **Proof by contradiction:** Assume that an undirected connected tree T has more than one perfect matching. Let's say there are two different perfect matchings M and M' in T. Consider the symmetric difference $M \oplus M'$. Since both M and M' are perfect matchings, we can easily see that every connected component in $M \oplus M'$ will be either an isolated vertex or an alternating cycle. This is true because, for any vertex v, if it matches the same vertex u in M and M' then it will be isolated in $M \oplus M'$. Otherwise, v would exist in an alternating cycle between edges in M and M'. This should be a cycle because both M and M' are perfect matching, hence v would be matched to two different vertices in both. Since T is a tree, we can't have any cycles, hence every connected component in $M \oplus M'$ will be an isolated vertex reaching the conclusion that $M = M'$, which is a contradiction to our initial assumption. Finally, we can easily see that a connected tree with one vertex has no perfect matching, and a connected tree with 2 vertices has one unique perfect matching.
- An alternative proof would be using the previous last fact (about 1 and 2 vertices) as an induction basis to prove the statement using induction on the number of vertices. A hint for this would be to start by matching a leaf v to its neighbor u, then consider the tree $T - \{u, v\}$, applying the induction hypothesis on its components. The main idea is that every leaf must be matched to its unique neighbor
- We want to prove that if a tree T has a perfect matching, then for every $v \in V$, the forest $T - \{v\}$ has exactly one component with an odd number of vertices: Let M be the

unique perfect matching in T, and let u be the vertex matched to v in this matching. Since M is a perfect matching in T, then for any component in the forest $T - \{v\}$ that doesn't contain u, we will have a perfect matching. Hence, all components in the forest $T - \{v\}$, except that containing u, will have a perfect matching and consequently an even number of vertices. So, there is exactly one component, the one containing u, with an odd number of vertices (this last component must have odd number of vertices as the initial tree T has even number of vertices, then $T - \{v\}$ must have an odd number of vertices).

Problem 6

We first observe that the sum of all completion times counts the jobs that are placed last on each machine once, the jobs that are second to last twice and so on. This means that we would in general prefer to to have heavier jobs placed closer to the end of the list of jobs. We will say that job j has position k on machine i if it is the k-th last job scheduled on that machine (e.g. position 1 means he is scheduled last).

Following this insight, we construct a bipartite graph $G = (A \cup B, E)$ where we have $A = \{m_{ik} : i \in [m], k \in [n]\}$ (represents orderings of jobs on machines), $B = \{v_j : j \in [n]\}$ and there is an edge between every job and every possible position on every machine. The weight of an edge e between job j and m_{ik} (position k on machine i) is $k \cdot p_{ij}$. Now, we just need to find a minimum cost maximum matching in this bipartite graph. We have $O(mn)$ vertices and $O(mn^2)$ edges in the graph and if we use the perfect min-cost matching algorithm, we will have to first expand it such that we will have $O(mn)$ vertices on both sides and $O(m^2n^2)$ edges, so the runtime is $O(VE + V^2 \log V) = O(n^3m^3)$.

Exercises for Units 29 and 30

28) In a cycle cover each vertex has exactly one incoming edge and exactly one outgoing edge.

To find a cycle cover of a directed Graph $G=(V, E)$, let's consider undirected Bipartite graph $G'=(V \cup V', E')$, where V' is a copy of V. Edge $e=(u, v) \in E'$ if and only if $u \in V$, $v \in V'$ and directed edge (u, v) belongs to E.

$|V \cup V'|=2n$, $|E'|=|E|=m \Rightarrow$ we can find a maximum matching M in G', in $O(mn)$ time.

Observe that a cycle cover of G gives us a perfect matching in G' and a perfect matching in G' gives us a cycle cover of $G \Rightarrow$ to check if G has a cycle cover we need to check if a maximum matching M is also a perfect matching. and if yes we can easily get a cycle cover of G from M.

Exercises for Units 29 and 30

4) We prove By induction on k

1. Base Case, $k=1 \Rightarrow$ each vertex has degree 1, so G is a perfect matching itself.
2. Induction step. We know that every $k-1$ regular Bipartite graph has $k-1$ edge-disjoint perfect matchings. Let's consider k-regular Bipartite graph $G=(A \cup B, E)$. Let $X \subseteq A$ and let E' be a set of edges connecting vertices from X to the vertices from $\Gamma(X)$. G is k-regular $\Rightarrow |E'| = k \cdot |X|$ and also $|E'| \leq |\Gamma(X)| \cdot k \Rightarrow |X| \leq |\Gamma(X)|$. Using the same argument we can prove that for every $Y \subseteq B$, $|Y| \leq |\Gamma(Y)|$. From this we get that $|A| \leq |\Gamma(A)| \leq |B|$ and $|B| \leq |\Gamma(B)| \leq |A|, \Rightarrow$ $\Rightarrow |A| = |B|$. This means that G has a perfect matching M. Every vertex in a perfect matching has degree $1 \Rightarrow$ $G' = (A \cup B, E \setminus M)$ is a $k-1$ regular Bipartite graph, so it has $k-1$ edge-disjoint perfect matchings $\Rightarrow$ $\Rightarrow$ G has k edge-disjoint perfect matchings