Technische Universität München
Department of Computer Science
Chair for Efficient Algorithms
Prof. Dr. Sami Khuri

Summer semester 2000
Solutions 6
June 20, 2000

Efficient Algorithms and Data Structures II

Problem 1 (10 credits)

Show that a graph is bipartite if and only if it has no odd length cycles.

Solution.

An odd-length cycle is not bipartite. Hence, if graph G is bipartite, it cannot contain an odd-length cycle.

Assume G has no odd-length cycles. Consider a breadth-first-search tree of G rooted at some node v. Colour the starting vertex red and then all its neighbours on the next level blue. Alternately colour the remaining levels red and blue so that consecutive levels have different colours. Claim: The colouring gives a bipartition, i.e. every edge of G has one red endpoint and one blue endpoint. This is clearly true for edges between consecutive levels. There cannot be edges from one level to a non-adjacent level (BFS property). That leaves edges within the same level. But such an edge implies that G has an odd cycle — if (u, w) is an edge on one level, then let P_u and P_w be the paths from u and w to their least common ancestor in the BFS tree. The cycle formed be P_u, P_w, and (u, w) has length $|P_u| + |P_w| + 1 = 2|P_u| + 1$, i.e. it is an odd cycle. This contradicts our assumption.

Problem 2 (10 credits)

Consider the following greedy algorithm for finding a maximal matching: Start with an arbitrary edge as the initial matching. Find another edge that does not have a vertex in common with the current matching. If one does exist, add it to the current matching. Repeat until no more edges can be added.

- **a.** What is the running time of this algorithm on a graph with n vertices and m edges?
- **b.** Give an example of a graph where the algorithm does not find a maximum matching.
- **c.** Show that the matching found by the algorithm always has at least half as many edges as a maximum matching.

Solution.

- **a.** Each edge is examined at most twice. $O(m)$.
- **b.** A path of three edges.

c. Consider the symmetric difference of the greedy matching and an optimal matching (that is, the edges contained either in the greedy matching *or* in the optimal one, but not in both of them). This consists of paths and even-length cycles (a node can have degree one or two, and on cycles, edges from both matchings must alternate). The two matchings have the same cardinality on all cycles and on even-length paths. On any odd-length path, the greedy matching has at least half as many edges as the optimal matching.

Problem 3 (10 credits)

Recall that a k-regular graph is a graph in which each vertex has degree k.
Show that every k-regular bipartite graph $G = (V, E)$ with vertex partition $V = L \cup R$ and $k > 0$ has a complete matching.
(*Hint: Use Hall's theorem for bipartite graphs.*)

Solution.

Counting the edges by endpoints in L and endpoints in R shows that $k|L| = k|R|$, so regularity implies that $|L| = |R|$. Hence it suffices to show that Hall's condition holds (a complete matching exists iff $|A| \leq |N(A)|$ for every $A \subseteq L$); a matching saturating L will be a complete matching. Consider $S \subseteq L$, and suppose there are m edges between S and $N(S)$. Since G is k-regular, $m = k|S|$. Since these m edges are incident to $N(S)$, we have $m \leq k|N(S)|$. Hence $k|S| \leq k|N(S)|$, so $|N(S)| \geq |S|$. Having chosen $S \in L$ arbitrarily, we have established Hall's condition.

Problem 4 (10 credits)

Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$, and let G' be its corresponding flow network. Give a good upper bound on the length of any augmenting path found in G' during the execution of the Ford-Fulkerson algorithm.

Solution.

By definition, an augmenting path is a simple path from s to t in the residual graph G'_f. Since G has no edges between vertices in L and no edges between vertices in R, neither does the flow network G' and hence neither does G'_f. Also, the only edges involving s or t connect s to L and R to t. Note that although edges in G' can go only from L to R, edges in G'_f can also go from R to L. Thus any augmenting path must go

$$s \to L \to R \to \cdots \to L \to R \to t$$

crossing back and forth between L and R at most as many times as it can do so without using a vertex twice. It contains s, t, and equal numbers of distinct vertices from L and R – at most $2 + 2\min\{|L|, |R|\}$ vertices in all. The length of an augmenting path is thus bounded above by $2\min\{|L|, |R|\} + 1$.