CS 357 Assignment #1

Greg Plaxton

August 24, 2011

There are two parts to this assignment. The first part, described in Section 1 below, consists of a programming task that is due by 8pm on Wednesday, August 31. Students are allowed to work on the first part with a partner, and are strongly encouraged to do so. Each team should turn in only one solution for the first part. If you are having trouble finding a partner, send me an email and I will match you up with someone else in the same situation. Please be sure to read the document entitled "Guidelines for Programming Tasks", which may be found in the Assignments section of the class website. This document includes a description of the slack time policy for programming tasks.

The second part, described in Section 2 below, consists of textbook exercises, and is due at the beginning of class on Wednesday, September 7. Each student should work on this part separately.

1 Programming & Problem Solving

A *matching* of a (undirected) graph $G = (V, E)$ is a subset M of E such that exactly $2|M|$ vertices in V are incident on (i.e., an endpoint of) some edge of M.

A *maximum-cardinalty matching (MCM)* of a graph G is a matching M of G such that $|M| \geq |M'|$ for all matchings M' of G.

The *weight* of a matching M of an edge-weighted graph is defined as the sum of the weights of the edges in M. A classic problem in combinatorial optimization is to determine a maximum-weight matching of a given edge-weighted graph.

The *weight* of a matching M of a vertex-weighted graph is defined as the sum of the weights of the $2|M|$ vertices that are incident on some edge of M. Remark: We can view a vertex-weighted graph $G = (V, E)$ as inducing a weight for each edge (u, v) in E that is equal to the sum of the weights of vertices u and v. Under this view, our definition of the weight of a matching M in a vertex-weighted graph conforms with our definition of the weight of M in the induced edge-weighted graph. End of remark.

A *maximum-weight MCM (MWMCM)* of an edge-weighted or vertex-weighted graph G is an MCM of G with weight at least as high as that of any other MCM of G.

A graph $G = (V, E)$ is *bipartite* if the set of vertices V can be partitioned into two sets V_0 and V_1 such that every edge in E has one endpoint in V_0 and one endpoint in V_1. Bipartite

graphs arise in numerous practical applications, and many efficient algorithms have been developed that exploit their special structure.

In most applications involving bipartite graphs, the bipartition of the vertices is explicitly specified as part of the input. In such cases, we generally prefer to view a bipartite graph as a triple (U, V, E) where U denotes the vertices on the "left" side, V denote the vertices on the "right" side, and every edge in E has one endpoint in U and one endpoint in V.

A bipartite graph $G = (U, V, E)$ is *convex* if it is possible to define a total order over the vertices of V such that the following condition holds: For any vertex u in U, and any vertices v, v', and v'' in V such that (u, v) belongs to E, (u, v') belongs to E, and $v < v'' < v'$, the edge (u, v'') belongs to E.

In the *Programming & Problem Solving* component of this course, we will develop efficient matching algorithms for vertex-weighted convex bipartite graphs. We adopt the following notation for working with convex bipartite graphs, or CBGs.

We refer to each "left" vertex of a CBG as a *ping*. Formally, a ping is a four-tuple of integers. For a ping $u = (i, a, b, w)$, we define $id(u)$ as i, $\min(u)$ as a, $\max(u)$ as b, and $weight(u)$ as w.

We refer to each "right" vertex of a CBG as a *pong*. Formally, a pong is an ordered pair of integers. For a pong $v = (i, w)$, we define $id(v)$ as i and $weight(v)$ as w.

We represent a CBG as a pair (U, V) where U is a set of pings, no two of which share the same first component, and V is a set of pongs, no two of which share the same first component.

Given a CBG (U, V) we define the following total order over U: For any pings u and u' in U, the inequality $u < u'$ holds if either (1) $\max(u) < \max(u')$ or (2) $\max(u) = \max(u')$ and $id(u) < id(u')$.

Given a CBG (U, V) we define the following total order over V: For any pongs v and v' in V, the inequality $v < v'$ holds if $id(v) < id(v')$.

The edge set of a CBG (U, V) is represented implicitly: There is an edge between ping u in U and pong v in V if $\min(u) \leq id(v) \leq \max(u)$.

Given a CBG $G = (U, V)$, we define the following total order over the set of edges of G: For any two edges (u, v) and (u', v') in G, the inequality $(u, v) < (u', v')$ holds if either (1) $v < v'$ or (2) $v = v'$ and $u < u'$.

Given a CBG $G = (U, V)$, we define a total order over the set of all MCMs of G as follows. Let M and M' be MCMs of G. Let α be the $|M|$-tuple consisting of the edges of M, arranged in ascending order. Let α' be the $|M|$-tuple consisting of the edges of M', arranged in ascending order. Then the inequality $M < M'$ holds if α lexicographically precedes α'.

For any matching M of a CBG G, we define $weight(M)$ as the sum, over all edges (u, v) in M, of $weight(u) + weight(v)$.

1.1 Your Task

Your program will read input from standard input, and write output to standard output. The first line of the input contains a nonnegative integer k that specifies the number of instances

to follow. The integer k is followed by k "input blocks". Your program will produce k "output blocks", one for each input block. Each input block specifies a CBG G, and the corresponding output block lists all of the MWMCMs of G.

For this assignment, it is recommended that you implement a "brute force" algorithm to compute the MWMCMs of a given CBG. Doing so should make it easy to ensure that the input-output behavior of your program is correct. You do not need to worry about the scalability of your algorithm, as we will only test it on input CBGs with a small number of vertices (at most ten, say).

1.1.1 Format of an Input Block

Each input block specifies a CBG $G = (U, V)$. The first line of an input block contains two integers m and n that specify $|U|$ and $|V|$, respectively. Each of the next m lines contains four integers specifying the four components of a ping in U. After these m lines, each of the next n lines contains two integers specifying the two components of a pong in V.

1.1.2 Format of an Output Block

We now describe the output block corresponding to an input block that encodes CBG G. Let ℓ denote the number of MWMCMs of G. Then the output block consists of $\ell + 1$ lines. The value of ℓ is printed on the first line. The ℓ MWMCMs of G are printed in ascending order on the next ℓ lines, one MWMCM per line. Each MWMCM M should be printed as follows. The edges of M are printed in ascending order, with a single blank separating successive edges. To print an edge (u, v) of M, print $id(u)$, followed by a colon, followed by $id(v)$.

2 Textbook Exercises

1. Exercise 2.4, page 67.
2. Exercise 2.8, page 69.