Programming HW # 1

In this homework, you will implement a variation of the stable marriage problem, where you will determine a maximum weight matching of vertices in a convex bipartite graph. The following definitions will help specify the problem requirements.

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

- The weight of a matching M of an edge-weighted graph is defined as the sum of the weights of the edges in M.
- 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.}

Maximum Cardinality Matching (MCM): A MCM of a graph G is a matching M of G such that $|M| \geq |M_0|$ for all matchings M_0 of G.

Maximum Weight Maximum Cardinality Matching (MWMCM): A 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.

Bipartite Graph: 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. Let us denote 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.

Convex Bipartite Graph (CBG): 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.

- We refer to each "left" vertex of a CBG as a tic. Formally, a tic is defined using a four-tuple of integers. For a tic $u = (i, a, b, w)$, we dene $id(u)$ 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 tac. Formally, a tac is an ordered pair of integers. For a tac $v = (i, w)$, 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 tics, no two of which share the same first component, and V is a set of tacs, no two of which share the same first component.

Ordering: Given a CBG $G = (U, V)$.

- We define the following total order over U: For any tics u and u' in U, the inequality $u < u'$ holds if either (1) $max(u) < max(u')$ or (2) $max(u) = max(u')$ and $id(u) < id(u')$.
- We define the following total order over V : For any tacs 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 tic $u \in U$ and tac $v \in V$ if $min(u) \leq id(v) \leq max(u)$.
- 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 $u < u'$.
- 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'.

Task for your program: For a given CBG called G described by two sets of vertices U and V as described above, find all of the MWMCMs of G.

Your program will read input from a plain text file of the form <filname>.in, and write output to a file called <filename>.out. The name of the output file must be same as that of input, just the extension should be '.out'. The first line of the input contains a non-negative 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.

Input Block Format: 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 tic in U. After these m lines, each of the next n lines contains two integers specifying the two components of a tac in V.

Output Block Format: The output block corresponding to an input block that encodes CBG G should follow the following format. Let p denote the number of MWMCMs of G. Then the output block consists of $p + 1$ lines. The value of p is printed on the first line. The p MWMCMs of G are printed in ascending order on the next p 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 $id(v)$.

Part 1: Provide a pseudo-code for the algorithm to find MWMCM for an edge-weighted CBG. Do not include the input parsing part; assume that you are given a CBG, and you can iterate over its left and right nodes, as well as over its edges. (15 points).
Part 2: Provide the worst-case runtime complexity for your pseudo-code and explain why. (5 points).
Part 3: Implement your algorithm to generate correct outputs for given inputs as described above. (80 points).