```
// Algorithm to find maximum matchings in bipartite graphs
Input:  Graph G=(V,E)
Output: Matching M of maximum cardinality
// Raises exception if G is not bipartite
// Algorithm works even if G is not connected

1. Use a maximal spanning forest of G to classify nodes as outer and inner:
   Mark a node as outer, its neighbors as inner, and their neighbors
   as outer, etc.  If there is any edge of G that connects two outer
   nodes or two inner nodes, then raise exception "Not bipartite"

2. Initialize: msize <-- 0;  for each u in V do u.mate <-- null

3. Find a maximal matching of G using a greedy algorithm:
   for each edge e=(u,v) in E do
       if u.mate = null and v.mate = null then
           u.mate <-- v;  v.mate <-- u;  msize++

4. // Repeat: find augmenting path, increase size of matching
   while true do
       Create an empty queue Q of vertices  // for outer nodes only
       for each u in V do
           u.seen <-- false;  u.parent <-- null
           if u.mate = null and u is an outer node then
                u.seen <-- true;  Q.Enqueue(u)

       while Q is not empty do
           u <-- Q.Dequeue()
           for each edge (u,v) in adj[u] do
               if not v.seen then
                  v.parent <-- u;  v.seen <-- true
                  if v.mate = null then  // augmenting path has been found
                     processAugPath(v)
                     go to the beginning of Step 4
                  else
                     x <-- v.mate;  x.seen <-- true;  x.parent <-- v
                     Q.Enqueue(x)
      // Q got empty, but no augmenting path was found
      Break out of the outer while loop of Step 4 and go to Step 5

5. Output current matching as a maximum matching


//Helper function to increase size of matching, using an augmenting path
processAugPath(u)
// u is a free inner node with an augmenting path to the root of the tree
   p <-- u.parent;   x <-- p.parent
   u.mate <-- p;  p.mate <-- u
   while x != null do    // go up two steps from outer node x
       nmx <-- x.parent;  y <-- nmx.parent
       x.mate <-- nmx;  nmx.mate <-- x
       x <-- y
   msize ++  // Matching size increases by 1
   return
```