H = [[1,2,0],[2,1,0],[2,1,0]]
R = [[0,1,2],[1,2,0],[0,2,1]]

0 prop 1 : maybe
1 prop 2 : maybe
2 prop 2 : pref 2 , rej 1
1 prop 1 : pref 1 , rej 0
0 prop 2 : pref 0 , rej 2
2 prop 1 : pref 1 , rej 2
2 prop 0 : maybe
j P C F
0 2 1 2
1 1 1 1
2 0 2 0

H = [[0,1,2],[1,0,2],[1,0,2]]
R = [[1,2,0],[0,1,2],[2,1,0]]

0 prop 0 : maybe
1 prop 1 : maybe
2 prop 1 : pref 1 , rej 2
2 prop 0 : pref 2 , rej 0
0 prop 1 : pref 0 , rej 1
1 prop 0 : pref 1 , rej 2
2 prop 2 : maybe
j P C F
0 1 1 1
1 0 1 0
2 2 2 2

H = [[0,2,1],[2,0,1],[0,2,1]]
R = [[1,2,0],[0,2,1],[2,0,1]]

0 prop 0 : maybe
1 prop 2 : maybe
2 prop 0 : pref 2 , rej 0
0 prop 2 : pref 0 , rej 1
1 prop 0 : pref 1 , rej 2
2 prop 2 : pref 2 , rej 0
0 prop 1 : maybe

j	P	C	F
0	1	2	1
1	0	1	0
2	2	1	2

2. Give a matching preference system with size 3 for which the propose-reject algorithm always finds a stable matching, or explain why this is not possible.

If you give any system of size 3. the propose-reject algorithm will return a stable matching.

So any system of size 3 is a correet answer.

3.
```
def myunion(x,y,P):
  rootx = findGP(x,P)
  rooty = findGP(y,P)
  P[rootx] = rooty

def findGP(x, P):
  px = P[x]
  if x==px: return x
  gx = P[px] #grandparent
  while px != gx:
    P[x] = gx
    x = px
    px = gx
    gx = P[gx]
  return px
```

P represents 8 components of size 1:

j	0	1	2	3	4	5	6	7
P[j]	0	1	2	3	4	5	6	7

Now show P after myunion(2,3,P):

P[j] 0 1 3 3 4 5 6 7

... and then after myunion(3,4,P):

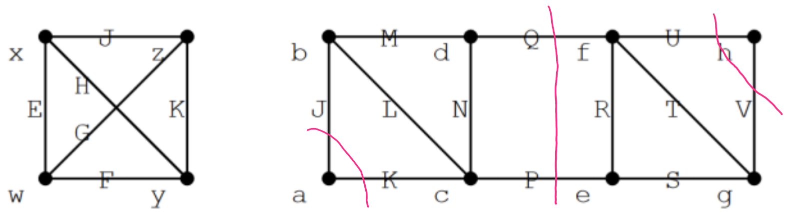
P[j] 0 1 3 4 4 5 6 7

... and then after myunion(4,5,P):

P[j] 0 1 3 4 5 5 6 7

... and then after myunion(5,6,P):

P[j] 0 1 3 4 5 6 6 7



4. Recall: a *cut* of a graph is a partition of the node set into two non-empty subsets. E.g. on the small graph (above left), {{w,x}, {y,z}} is a cut with cross-edges {F,G,H,J}. RKMC is the randomized Kruskal min cut algorithm: unless otherwise stated, its input is a uniform-random permutation of the edges.

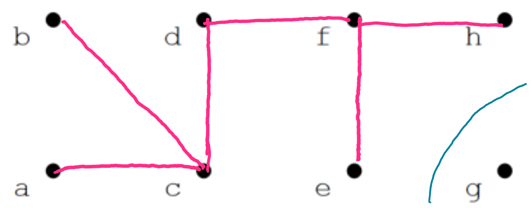
a) For the big graph, give each min cut (partition and cross-edges) . . .

partition: {{a}, {b, c, d, e, f, g, h}} cross-edges: {J, K}

{{h}, {a, b, c, d, e, f, g}} {U, V}

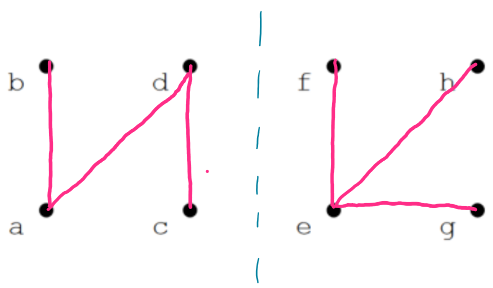
{{a, b, c, d}, {e, f, g, h}} {Q, P}

b) . . . and give the forest (draw on the nodes below) and cut (partition and cross-edges) found by RKMC when edges are input in order LUQRKNTVSJMP.



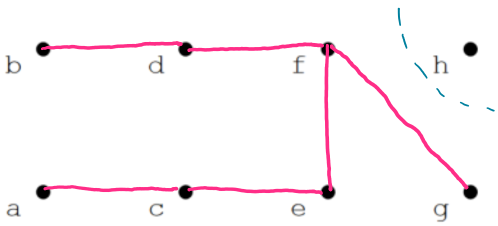
partition: {{a, b, c, d, e, f}, {g}}

cross-edges: {S, T, V}



partition: {{a, b, c, d}, {e, f, g, h}}

cross-edges: {Q, P}



partition: $\{\{a, b, c, d, e, f, g\}, \{h\}\}$

cross-edges: $\{ U, V \}$

3. **[2 marks]** Let G be a connected graph with a cut $\{X, Y\}$ with $G[X]$ (the subgraph of G on the node set X) connected but $G[Y]$ disconnected, with exactly two components $G[Y_1]$ and $G[Y_2]$. Prove or disprove: $\{X, Y\}$ is a min cut of G.

$\boxed{\text{False.}}$ Proof:

Because G is connected and $G[Y]$ is disconnected, there must be at least one edge connecting $G[X]$ and $G[Y_1]$. Denote these edges as E_1.

Similarly, there must be at least one edge connecting $G[X]$ and $G[Y_2]$. Denote these edges as E_2.

From the above, we see that both $\{X \cup Y_2, Y_1\}$ & $\{X \cup Y_1, Y_2\}$ are cuts of G, with sizes $|E_1|$ and $|E_2|$, respectively. We also see that cut $\{X, Y\}$ has size $|E_1| + |E_2|$. Since $|E_1| + |E_2| > |E_1|$ and $|E_1| + |E_2| > |E_2|$, $\{X, Y\}$ cannot be a min cut. (There are at least two smaller ones.)