cmput 304 2023 study questions 3 (revised Oct 4)

1.
```
def myfind(v,P): # simple find
  while P[v] != v:
    v = P[v]
  return v

def myunion(x,y,P): # simple union
  rootx = myfind(x,P)
  rooty = myfind(y,P)
  P[rootx] = rooty

def findGP(x, P): # gp compress
  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
```

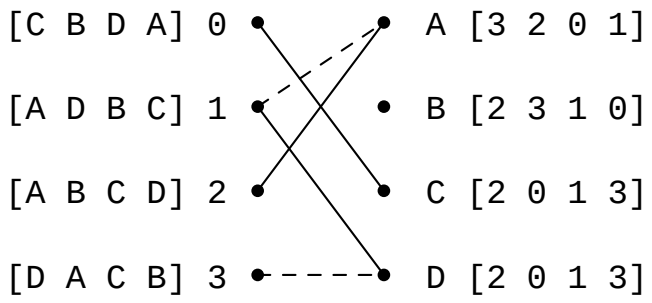
a) Here is the parent list P representing 10 components of size 1. Show P after each of these commands: `myunion(0,1,P)`, `myunion(1,2,P)`, ..., `myunion(8,9,P)`.

j	0	1	2	3	4	5	6	7	8	9
P[j]	0	1	2	3	4	5	6	7	8	9

b) Repeat this question if, in `myunion`, `myfind` is replaced with `findGP`.

j	0	1	2	3	4	5	6	7	8	9
P[j]	0	1	2	3	4	5	6	7	8	9

2. Here is a diagram of the propose-maybe-reject stable matching algorithm after some number of rounds. a) For each proposer (hospital), in the next round, what new proposals are made? Explain carefully. b) After the new proposals are made, what rejections are made by maybe-rejecters (residents)? Explain carefully.

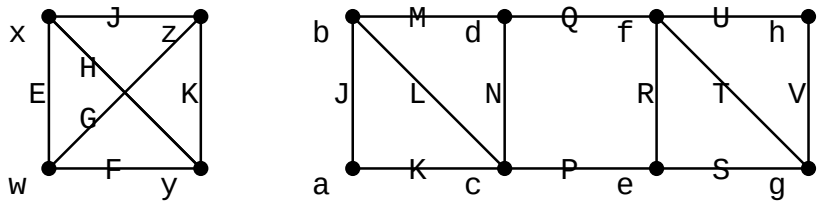


3. a) For each of the following H and R, show the output printed by `m=propose_reject(H,R)`. Hint: check your answer using `sm.py` in class github repo `graphs/stable_match`.

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

```
def propose_reject(H,R):
  n = pref_system_size(H,R)
  F,C = [None] * n, [0 for j in range(n)]
  rejection, rounds = True, 0
  while rejection:
    rejection, rounds = False, rounds + 1
    print('     round', rounds)
    for j in range(n):
      h_choice = H[j][C[j]] # current H proposal
      if F[h_choice] == None: # R has no proposals
        F[h_choice] = j
        print(' ',j,' prop ',h_choice,': maybe',sep='')
      elif F[h_choice] != j: # R has two proposals
        r_maybe = F[h_choice] # R's current proposal
        if prefers(R[h_choice], j, r_maybe):
          r_reject, r_maybe = r_maybe, j
          F[h_choice] = r_maybe
        else:
          r_reject = j
        print(' ',j,' prop ',h_choice,
              ': pref ',r_maybe,', rej ', r_reject, sep='')
        C[r_reject] += 1 # H[j_reject] consider next pref
        rejection = True # a proposal was rejected
  P = [H[j][C[j]] for j in range(n)]
  print('\nj  P  C  F   ',rounds,'rounds')
  [print(j, P[j], C[j], F[j], sep='  ') for j in range(n)]
  return P
```

b) Prove/disprove: for each preference system with size 3, the propose-reject algorithm always finds a stable matching in which at least one proposer gets their first choice.



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) ...

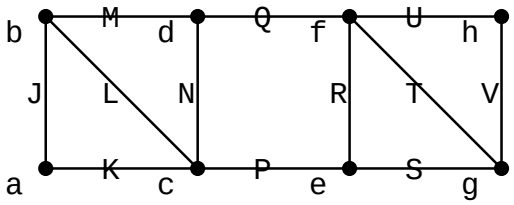
   b) ... and give the forest (draw on the nodes below) and cut (partition and cross-edges) found by RKMC when edges are input in order LUQRKNTVSJMP.

   b ● d ● f ● h ●

   a ● c ● e ● g ●

5. 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.

Hint: see `https://webdocs.cs.ualberta.ca/~hayward/304/asn/22/q8p1sol.pdf`



6. For the above graph, give a cut (partition and cross-edges) that will never be found by RKMC

7. . . . and give the theoretical lower bound on the probability that RKMC finds a min cut after 1 execution.

8. Can you improve on your answer to the previous question? Explain.

9. Let p be the probability that 1 RKMC execution finds a min cut on a graph G. Which of these describes the number of executions t we should iterate RKMC in order to guarantee that we find a min cut with probability at least .9? Justify briefly.

   a) the smallest t such that $pt \geq .9$
   b) the smallest t such that $p/t \geq .9$
   c) the smallest t such that $(1-p)^t \geq .9$
   d) the smallest t such that $1-(1-p)^t \geq .9$
   e) the smallest t such that $(1-p)^{1/t} \geq .9$
   f) the smallest t such that $1-(1-p)^{1/t} \geq .9$

answers

1. Run the code.

2. a) 0 matched (no new proposal), 1 matched (n n p), 2 matched (n n p), 3 proposes to A.

   b) A prefers new 3 to current 2, so A rejects 2.

3. a) Run the code.

   b) There are many different correct answers: here is one.

   If the set of hospital-first-choices has size 3 the PR-matching gives each hospital their first choice: we don't want that. If the set of hospital-first-choices has size 1 (each hospital prefers the same resident), then one hospital is guaranteed to get that resident, and so their first choice: we don't want that. So two hospitals have first choice say A, and the other has first choice say B, so we have this preference system:

```
[A,?,?]  0     A  [?,?,?]
[A,?,?]  1     B  [?,?,?]
[B,?,?]  2     C  [?,?,?]
```

   We want A eventually to be matched with 2, so try A prefs [2,1,0].

```
[A,?,?]  0     A  [2,1,0]
[A,?,?]  1     B  [?,?,?]
[B,?,?]  2     C  [?,?,?]
```

   Now complete the preference lists so that 0,1 do not match A and 2 does not match B, say like this:

```
[A,B,?]  0     A  [2,1,0]
[A,C,?]  1     B  [?,?,2]
[B,A,?]  2     C  [?,?,?]
```

   The PR matching is 0B, 1C, 2A. Any of these systems work:

```
[A,B,C]  0     A  [2,1,0]
[A,C,B]  1     B  [?,?,2] or [0,2,1]
[B,A,C]  2     C  [?,?,?]
```

4. See `https://webdocs.cs.ualberta.ca/~hayward/304/asn/22/q8p2sol.pdf`

5. See `https://webdocs.cs.ualberta.ca/~hayward/304/asn/22/q8p2sol.pdf`

6. RKMC always returns a cut $\{X, Y\}$ where $G[X]$ and $G[Y]$ are connected. There are many other cuts, e.g. with $X = \{a, g\}$ and Y all other nodes.

7. `2/(11*12) = 1/66 = 0.15...`

8. There are exactly 3 min cuts, with respective cross-edge sets {J,K}, {P,Q}, {U,V}.

   So the execution of RKMC finds a min cut if the initial sorting of edges ends J,K or K,J or P,Q or Q,P or U,V or V,U. The probability of each of these six events is exactly 1/12 times $1/11 = 1/132$. These events are all independent, so the probability that exactly one of them occurs is $1/132 + 1/132 + \ldots + 1/132 = 6/132 = 1/22 = 0.045...$

   This answer can be improved further: e.g. in the case where the two components found by the algorithm span the left 4-clique and the right 4-clique, the remaining edges of the two cliques can occur after the cross-edges, e.g. any permutation starting (J,R,T,N,V,M,... works, because at this point we have found a min cut. The exact probability will be greater than 1/22.

9. See `https://webdocs.cs.ualberta.ca/~hayward/304/asn/22/q8p2sol.pdf`