Homework 1

Q_1:

Hospitals:

Q_i: The queue of available seats for hospital h_i

H: The queue of hospitals w/t at least one available seat

Cnt[i]: next in pref. list

C[i]: The capacity of h_i

P_i: Pref. list of h_i

Students:

P_{S_j}: Pref. list of S_j

Inv_j: Inverse pref. list of S_j

assigned[j]: -1 if not assigned, ≥ 0 hospital id

Seat: assigned seat

```
for each hospital hi
        add {1, ..., C[i]} to Qi
        Cnt[i] = 0
for each student Sj
        assigned[j] = -1
add {h1, ..., hm} to H
while (H is not empty)
    hi = H.delete()
    Sj = Pi[Cnt[i]] // next in pref. of hi
    if (assigned[j] == -1)
        assigned[j] = i
        Seat[j] = Qi.delete()
    else if (Invj[i] < Invj[assigned[j]])
        // release the current seat
        k = assigned[j]
        Qk.add(Seat[j])
        if (|Qk| == 1)
            H.add(hk)
        // accept the new proposal
        Seat[j] = Qi.delete()
        assigned[j] = i
    Cnt[i] += 1
    if (|Qi| >= 1)
        H.add(hi)
// end of while
```

- first Type of instability will not occure, because hospitals propose in the order of their pref.
  ↳ proposes to S_j first
- 2nd Type does not occure because hospitals propose based on their pref., Students switch only they recv. a proposal from a hospital which they prefer to current one

- The algorithm terminates after all hospitals reach the end of their lists. (worst_case)
- all seats get occupied.

- worst Case analysis

$O(n\,m)$

Correct Alg.: 20 pts
opt. implementation: 5 pts
Stability Proof: 5 pts

Home work 1, Q2:

I: $(n+a)^b = \sum_{i=0}^{b} \binom{b}{i} n^i a^{(b-i)}$

$= n^b + \cdots + a^b$

$= \Theta(n^b)$

II: Let $\log n = m$

$\Rightarrow T(n) = T(2^m) = S(m)$

$T(\sqrt{n}) = S(m/2)$

$\Rightarrow S(m) = 2S(m/2) + m$

$= \Theta(m \log m)$

$\Rightarrow T(n) = \Theta(\log n \log\log n)$

III: Let $\log n = m$

$\Rightarrow \log(n/2^k) = \log n - \log 2^k$

$= m - k$

$T(n) = T(n/2) + \log n$

$= m + (m-1) + T(n/2^2)$

$\cdots$

$= m + (m-1) + \cdots + (m-m) + T(n/2^m)$ (1)

$= \Theta(m^2) = \Theta(\log^2(n))$

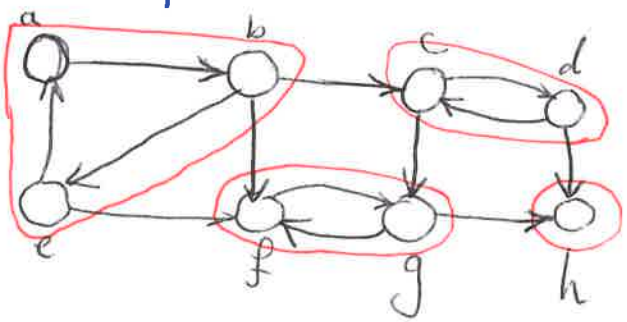
IV: $T(n) = 7T(n/3) + \Theta(n^{1/2})$

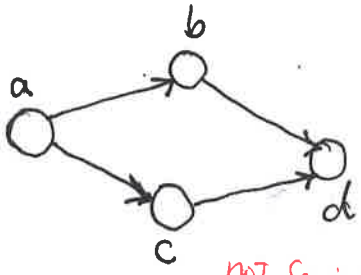
$7 > 3^{1/2}$

$\Rightarrow T(n) = \Theta(n^{\log_3 7})$

Q3:

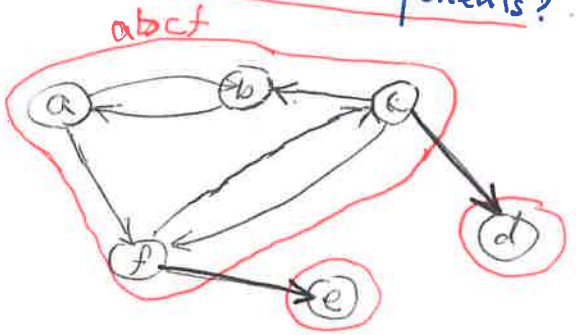
(A) Understanding the problem w/t a few examples





NOT Semi-Connected

(B) Hint: How about finding Strongly Connected Components?



Observation:

i: Strongly Connected Components form a DAG ← why?

ii: In the DAG of components if there is no path b/w two components, then the original graph is not Semi-connected.

Homework1 Q3-Cntud.

- Two nodes in a DAG are connected if there is a path b/w them.

→ if there is no path b/w two nodes their index in the topological order can swap.

→ if there is no edges b/w two nodes n_i and n_{i+1} in the topological order, their indexes can get swapped
↳ There is no path b/w them

⇒ Conclusion: if b/w every pair of strongly connected components in the formed DAG, there is no edge, The graph G is not Semi-Connected otherwise, it is.

Algorithm:

G_{DAG} = StronglyConnectedComponents(G)
V_{sorted} = TopologicalOrder(G_{DAG})
for i = 0 to $|V_{sorted}|-1$
  if $V_{sorted}[i] \rightarrow V_{sorted}[i+1] \notin G_{DAG}.edges$
    return False
return True

↖ $O(n+m)$

- understanding the challenge (that you cannot simply run forward / backward Traversal) : 10pts
  (b/w)
- Observation that Strongly connected components there should be a path: 10pts
- the conclusion about the DAG and the Algorithm: 10pts

Bonus:

- Let A be the Adjacency matrix of graph G
- Claim: $A(I-A)^{-1}$ contains the number of paths b/w all pairs of points.

$B = A + A^2 + A^3 + \dots$

$AB + A = \underbrace{A(A + A^2 + A^3 + \dots) + A}_{B}$

$\Rightarrow AB + A = B$

$A = B - AB = B(I-A)$

$\Rightarrow B = A(I-A)^{-1}$

↖ Computing this takes $O(n^{2.807})$