Kiarash Hosseini

1. (Chapter 7 Problems 1 and 2 of the text book) 15 points
   (a) There are three minimum cuts:
   1. $A = \{s\}, B = \{u, v, t\}$
   2. $A = \{s, v\}, B = \{u, t\}$
   3. $A = \{s, u, v\}, B = \{t\}$

   (b) The minimum capacity of a cut for this network flow is 4. (calculations is not requested)

   (c) The value of this flow is 18. This is not a maximum flow in this graph. Consider this flow (repetition is made purposefully)

   $s : f(s, u) = 8, f(s, v) = 8, f(s, x) = 5,$

   $u : f(s, u) = 8, \| f(u, w) = 3, f(u, t) = 5$

   $v : f(s, v) = 8, \| f(v, w) = 5, f(v, x) = 3$

   $x : f(s, x) = 5, f(v, x) = 3, \| f(x, t) = 8$

   $w : f(u, w) = 3, f(v, w) = 5, \| f(w, t) = 8,$

   The outflow of s is 21, and the conservation condition is made as the sum of inflow and outflow of all nodes are equal.

   (d) As shown in 1, a minimum cut is $A = \{s, v, u, w\}, B = \{x, t\}$
   Capacity of this minimum cut is 21, as:
   $c(A, B) = c(s, x) + c(v, x) = c(w, t) + c(u, t) = 5 + 3 + 8 + 5 = 21$

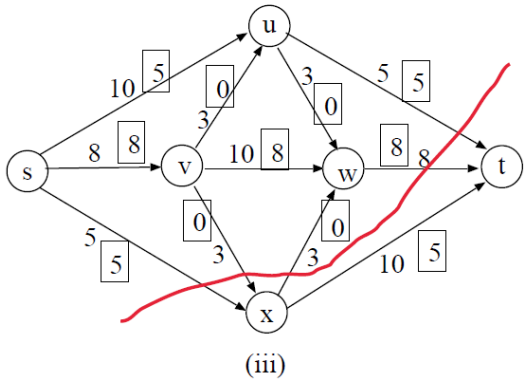

Figure 1: Minimum Cut.

2. (a) Yes, it is a flow on G. It satisfies both conservation and capacity condition.
   Conservation condition:
   $f^{in}(v) = f^{out}(v) = f^{in}(x) = f^{out}(x) = 0$

$f^{in}(u) = 8, f^{out}(u) = 8$
$f^{in}(w) = 3, f^{out}(w) = 3$
The residual graph is given in graph 2

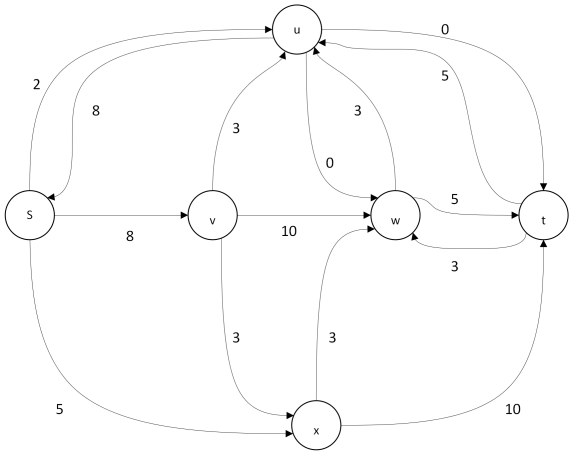

Figure 2: Residual Graph.

(b) Maximum flow: 21
Edge flows:
$s : f(s,u) = 8, f(s,v) = 8, f(s,x) = 5,$
$u : f(s,u) = 8, \|f(u,w) = 3, f(u,t) = 5$
$v : f(s,v) = 8, \|f(v,w) = 5, f(v,x) = 3$
$x : f(s,x) = 5, f(v,x) = 3, \|f(x,t) = 8$
$w : f(u,w) = 3, f(v,w) = 5, \|f(w,t) = 8,$

3. (Chapter 7 Problem 7 of the text book) 15 points
Use the following algorithm based on network flow:

0. for each client i, find the stations for which the client is within the range, M_i /*
Initialize $M_i = \varnothing$
for $i = 1$ to n, **do** /* n times */
for $j = 1$ to k, **do** /* k times */
if $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq r$, **do** /* $O(1)$ */
$M_i = M_i \cup \{j\}$ /* $O(1)$ */
end if
end for
end for

Running time is $O(n \times k)$ *
(For simplicity, in the pseudocode it is mentions that M_i is a set, but it is actually

an array of size k, so that element j is 1 if station j is withing range r of client i, 0 other wise. Similarly, when update M_i, or checking element j of M_i, it can be done in constant time)

1. Define a network flow as follows. /* $O(nk)$ */
   - Add a nodes s and t as source and sink. /* $O(1)$ */
   - Add a node i, for each of n clients. /* $O(n)$ */
   - Add a node u_j, for each of $j = 1, ..., k$ stations. /* $O(k)$ */
   - Add an arc (s, i) for each client i with capacity of 1. /* $O(n)$ */
   - Add an arc (i, u_j) with capacity of 1 if station $j \in M_i$ /* $O(nk)$ */
   - Add an arc (u_j, t) for each station u_j with the capacity L.
2. Find the s-t maximum flow. /* $O(nk \log(L))$ */
3. If the maximum flow of this network is equal to n, then every client can be connected simultaneously to a base station subject to the range and load condition above. Otherwise, the answer is no. /* $O(1)$ */

Running time:
Maximum number of edges: $n + (n \times k) + k$,
Number of nodes: $1 + n + k + 1$
If implementing Scaling-Max-Flow Ford-Fulkerson algorithm, the running time is $O(nk \log(L))$

Proof:
$f(u, t)$ is the number of clients assigned to each station u.
If $f(i, u) = 1$ then client i is assigned to station u.
Client i can be assigned to station u if it is in the range d of that. Thus the range condition is satisfied.
By conservation condition for nodes i, and the fact that all capacity values are integers, for each i, only one (i, u) is equal to one and the rest are zero. Thus each client is assigned to a single station.
The outflow of each node U is less than L, thus the load condition is satisfied.
The inflow of node u is the number of clients assigned to the station u and, by conservation condition, it is equal to (u, t), which the number of clients assigned to each station.

4. (Chapter 7 Problem 8 of the text book) 15 points (a)
   Define Excess Supply as $ES = \sum_{i=O,A,B,AB}(S_i - d_i)$. if $ES < 0$, supplies does not suffice to the demands.
   Define a Circulation with Demand graph as follows. /* $O(1)$ */
   - Add nodes u_i for $i = O, A, B, AB$ and v_i for $i = O, A, B, AB, N$.
   - Define $d(u_i) = -S_i$ and $d(v_i) = d_i$, for $i = O, A, B, AB$

- Define $d(v_n) = ES$
- Add arcs from u_O to $v_O, v_A, v_B, v_{AB}, v_N$ with capacity of ∞
- Add arcs from u_A to v_A, v_{AB}, v_N with capacity of ∞
- Add arcs from u_B to v_B, v_{AB}, v_N with capacity of ∞
- Add arcs from u_{AB} to v_{AB}, v_N with capacity of ∞

Find if there is a feasible circulation in this graph using Scaling-Max-Flow Ford-Fulkerson algorithm.
If yes, the supplies does suffice to the demands

Running time:
There are constant number of edges and nodes, this the algorithm is $O(\log(C))$, $C = \max\{d_i, S_i, ES\}$

Proof:
If total supply is less than demand, $ES < 0$ and the algorithm end in supplies does not suffice to the demands

If supply is larger than or equal to demand, then some of it is not needed, and it goes to node N. In this case, total demand is equal to total supply when node N is considered.

One unit of blood supply can only satisfy one unit of blood demand and vise versa.

The blood transfer is define so that satisfy the biological condition.

The capacity of each edge is the number of blood units that should be assigns from one group to another group or excess supply.

(b) No. To satisfy demand for O and A, only O and A can be used. Total supply of O and A is $50 + 36 = 86$. Total demand for O and A is $45 + 42 = 87$. One unit of demand for either A or O cannot be satisfied.

(a')
We can solve part by also directly using a flow graph as follows. /* $O(n^2)$ */

- add nodes s and t as source and sink
- Add nodes u_i and v_i for $i = O, A, B, AB$. /* $O(n)$ */
- add nodes s and t as source and sink
- Add arcs from u_O to v_O, v_A, v_B, v_{AB} with capacity of ∞
- Add arcs from u_A to v_A, v_{AB} with capacity of ∞
- Add arcs from u_B to v_B, v_{AB} with capacity of ∞
- Add arcs from u_{AB} to v_{AB} with capacity of ∞
- Add arcs from s to u_i with capacity of S_i, for $i = O, A, B, AB$
- Add arcs from v_i to t with capacity of d_i, for $i = O, A, B, AB$

Find the maximum network flow using Scaling-Max-Flow Ford-Fulkerson algorithm.

Running time: There are constant number of edges, this the algorithm is $O(\log(C))$, $C = \max d_i, S_i$

5. (Chapter 7 Problem 14 of the text book) 15 points

Define a network flow G' as follows. /* $O(|V|)$ */

- Add nodes s and t as source and sink. /* $O(1)$ */
- Add arcs with capacity 1 from s to each $s_i \in S$ /* $O(|S|)$ */
- Add arcs with capacity inf from each $x_i \in X$ to t /* $O(|X|)$ */
- Add capacity 1 to each arc in G /* $O(|V|)$ */

Find the maximum network flow from s to t using Ford-Fulkerson algorithm

If the maximum network flow is equal to $|S|$, then set of evacuation routes exists.

As it is not explicitly mentioned, I assume a safe node can accept more than one population nodes. (in case it could accept only one population node, then inf should be changed to 1 and checked if $|S| \geq |X|$)

Running time:
Number of edges: $|V| + |S| + |X|$, which is $O(|V|)$
Using Ford-Fulkerson algorithm, the running time of finding the maximum network flow is $O(|V| \times |S|)$, as the total capacity of edges from s is $|S|$ by construction. Thus, it is $O(m \times n)$ asymptotically as $|S| \leq n$

Proof:
There is one edge from s to each s_i by construction. Thus if maximum network flow is $|S|$, then each population center has a unique path to a safe node.

Each edge in G has capacity of 1, thus can only be used for flow from one population center to a safe zone.

The rest of the proof is similar to that of the theorem in the lecture node that shows "The maximum number edge-disjoint $s \to t$ paths equals the max-flow value".

6. 15 points

(a) Graph 3 shows the outcome when H is reduced to a circulation with demand H'

(b) Graph 4 shows the outcome when H' is reduced to a network flow G

(C) Graph 5 shows the residual graph w.r.t. a flow that circulates two unities from the path including the nodes on the bottom of the graph.

(b) Graph 6 shows the maximum flow of graph G. Graph 7 shows the solution for H based on this flow.

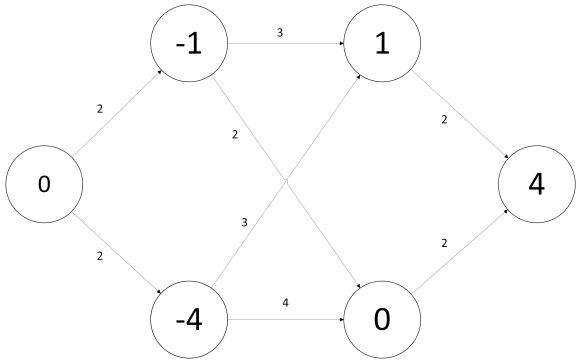

Figure 3: circulation with demand.

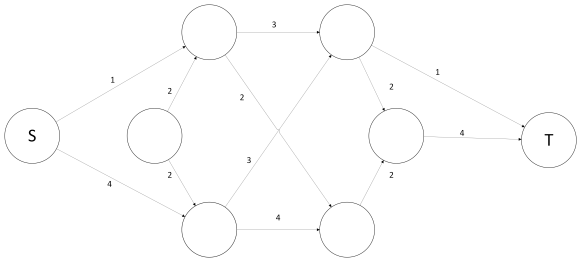

Figure 4: network flow.

7. (Chapter 7 Problem 31 of the text book) 15 points

   I use the following three steps.

   - For each i, find the set of boxes that i can be put inside.
   - Find the maximum number of matching between each box i and set of M_i.
   - Prove that Maximum number of matches is equal to minimum number of visible boxes.

   First Step:
   Input: (x_i, y_i, z_i), $i = 1, 2, ..., n$
   Output: M_i, $i = 1, 2, ..., n$

   Sort boxes by x_i and put the associated s_i in sorted array S /* $O(n \log(n))$ */
   $M_i = \varnothing$, $i = 1, 2, ..., n$ /* $O(n)$ */
   for $i = 1$ to n **do** /* n times */
   for $j = i + 1$ to n **do** /* n times */
   if $S[i] < S[j]$ **do** /* $O(1)$ */
   $M_i = M_i \cup \{S[j]\}$ /* $O(1)$ */
   end if
   end for
   end for
   Return $M_i, i = 1, 2, ..., n$

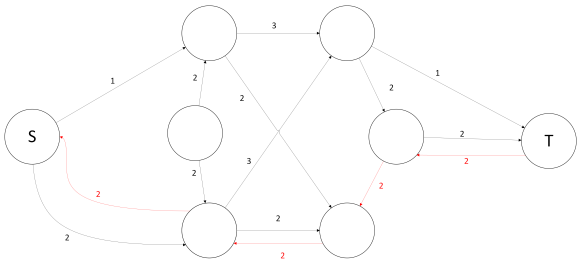

Figure 5: Residual Graph.

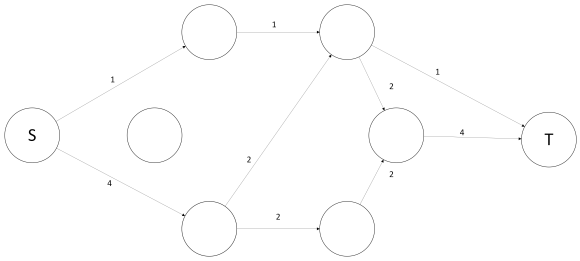

Figure 6: maximum flow.

Running time is $O(n^2)$

(For simplicity, in the pseudocode it is mentions that M_i is a set, but it is actually an array of size n, so that element j is 1 if $s_i < s_j$, 0 other wise. Similarly, when updating M_i, or checking element j of M_i, it can be done in constant time)

If $x_i \leq x_j$, then it is not possible that $s_j < s_j$. That means all elements in which i could be put inside have larger indexes than i in S.

Second Step:
Define a Bipartite Matching Problem as follows. /* $O(n^2)$ */

- Add nodes u_i and v_i for $i = 1, ..., n$. /* $O(n)$ */
- Add an arc from u_i to v_j, if $j \in M_i$. /* $O(n^2)$ */

Find the matching in this bipartite with maximum size after using Ford-Fulkerson algorithm for Network flow algorithm for bipartite matching

Running time:
Maximum number of edges is n^2, thus the running time $O(n^3)$

Proof:
It is a Bipartite, as there is not edge between u_i and u_j, nor v_i and v_j for any i and j.

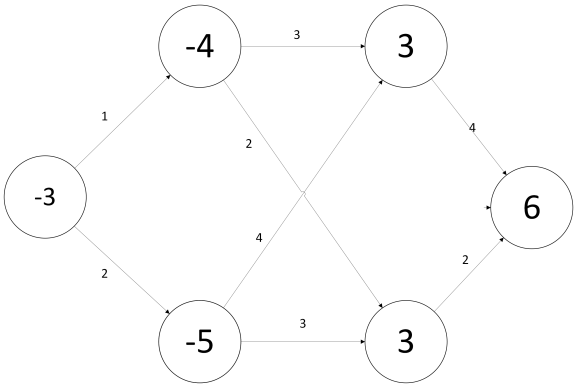

Figure 7: a solution for H .

Each box is put inside at most one box, and each box contain at most one box.
Each time a box is put inside another box, the number of visible boxes is decreased by one. Thus the number of visible boxes is minimized when maximizing incidence putting a box inside another box.
By construction, a box could be put inside another box if has smaller s