Problem (Life Imitates Art, TAs Grade Homework). Suppose you are a TA for `CS381`[1]. You and your fellow TAs (there are $n \in \mathbb{N}$ TAs total) are very stressed because there is a backlog of $k \in \mathbb{N}$ homeworks that need to be graded. To grade these k homeworks, Professor Kent has split the TAs up into k groups to handle each homework. Crucially, each group has one leader, who handles the most work (delegation, rubric construction, etc.).

Unfortunately, he used ChatGPT, so some TAs are in more than one group. He decided that the best we could do was to enforce a 'fairness requirement', where no single person leads too many groups.

Formally, there are k homeworks and n TAs. Homework i for $i \in [k]$ is graded by TA group $S_i \subseteq [n]$ with some leader $j \in S_i$. The fairness requirement is that any TA $j \in [n]$ who is in groups $S_{j_1}, \ldots, S_{j_m}$, does not lead more than

$$\left\lceil \frac{1}{|S_{j_1}|} + \frac{1}{|S_{j_2}|} + \cdots + \frac{1}{|S_{j_m}|} \right\rceil = \left\lceil \sum_{j \text{ s.t. } i \in S_j} \frac{1}{|S_j|} \right\rceil$$

of those groups. Given the k groups $S_1, S_2, \ldots, S_k \subseteq [n]$ as input, give a polynomial time algorithm (the faster the better) that *always* finds a leader assignment following the fairness requirement, or show that a poly-time algorithm for this problem implies a poly-time algorithm for SAT.

*Solution:*Form a capacitied graph $G = (V, E, C)$ as follows:

- **Vertices V.** Add a source s and a sink t. For each group S_i, add a vertex v_i. For each TA, create vertices p_i.
- **Edges E.**
  1. (Single Leader). Connect $s \to v_i$ for each $i \in [k]$ with capacity 1.
  2. (TA grouping). For each $i \in [k]$ and each $j \in S_i$, connect v_i to p_j with capacity 1.
  3. (Fairness Req.). For each $j \in [n]$, connect $p_j \to t$ with capacity

$$\left\lceil \sum_{i \text{ s.t } j \in S_i} \frac{1}{|S_i|} \right\rceil.$$

Then, to solve this problem, first run a max (s,t)-flow algorithm (either Edmunds-Karp or Ford-Fulkerson). For each $i \in [k]$: if the resulting flow puts flow through a $v_i \to p_j$ edge, then assign TA p_j to lead group S_i. Output the assignment, which we claim satisfies fairness and is consistent.

Runtime. Constructing the flow graph takes $O(nk)$ time. Since this a graph with $(n+k+2)$ vertices, $(k+2n)$ edges, and maximum flow at most k, we can find the max flow in $O(k^2+nk)$ time using Ford-Fulkerson. Note that using Edmunds-Karp here takes strictly worse $O(n^3+k^3)$ time.

[1]For me, this is very easy.

Correctness. We want to show that this algorithm always finds a satisfying fair assignment for any groups $S_1, \ldots, S_k$ of n TAs. We show (1) the flow is exactly k, and (2) this implies that the assignment found is fair and consistent (only one leader assigned per group).

First, to show that the max flow is at least k, we show the existence of a k flow in the graph. From the source s, push 1 flow through each v_i for $i \in [k]$. Then, from each v_i, send $\frac{1}{|S_i|}$ to p_j, for each $j \in S_i$. In total, there is $|S_i| \times \frac{1}{|S_i|} = 1$ flow leaving each v_i. Lastly, observe that the flow entering p_j is

$$\sum_{i \text{ s.t } j \in S_i} \frac{1}{|S_i|} \leq \underbrace{\left\lceil \sum_{i \text{ s.t } j \in S_i} \frac{1}{|S_i|} \right\rceil}_{\text{capacity of } p_j \to t}$$

so we are able to send all the flow entering p_j to the target t. Since each $p_j \to t$ edge has $\sum_{i \text{ s.t } j \in S_i} \frac{1}{|S_i|}$ flow through it, the final total flow is

$$\begin{aligned} \sum_{j \in [n]} \sum_{i \text{ s.t } j \in S_i} \frac{1}{|S_i|} &= \sum_{i \in [k]} \sum_{j \in S_i} \frac{1}{|S_i|} \\ &= \sum_{i \in [k]} \frac{|S_i|}{|S_i|} = k. \end{aligned}$$

Lastly, since there is k capacity leaving the source, the flow is at most k, so we can conclude the flow is equal to k.

Now, we show fairness and consistency. First observe that since the constructed graph G only has integral capacities, the flow through any edge is also integral. Since the flow is exactly k, and there is exactly 1 capacity from the source s to each group vertex v_i (so the flow is either 0 or 1 through this edge), each unit of flow is in a one-to-one correspondence with some specific group i. Again, since flow is integral, each edge $v_i \to p_j$ for $i \in [k]$ and $j \in S_i$ has flow either 0 or 1, so each group vertex v_i only sends flow through a single person vertex v_j. Thus, the flow $(s \to v_i \to p_j \to t)$ always corresponds to exactly one assignment of TA j to group i. Lastly, this assignment is fair since the last edge $p_j \to t$ encodes the fairness constraint exactly.

Problem Exercise 22.1**, part** 1**.** Recall the array-backed lists from Section 22.3, where we showed how to support `append` in $O(1)$ amortized time via the "doubling trick". We would like to support a `remove` operation with similar amortized time. Consider the following approach:

> After removing the element from the array and decreasing the size counter, if the size of the list is now less than half of the capacity of the array, allocate a new array of half the capacity and copy the elements of the list over.

Show that this does not achieve $O(1)$ amortized time. That is, give a sequence of n `append`/`remove` operations on which the data structure would take $\Omega(n^2)$ time.