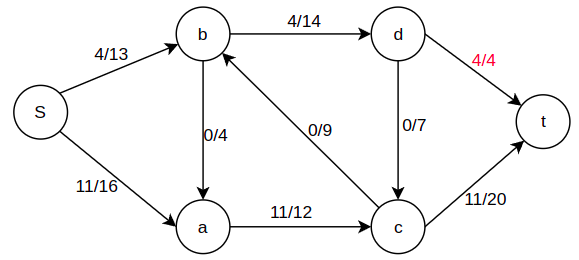
1.

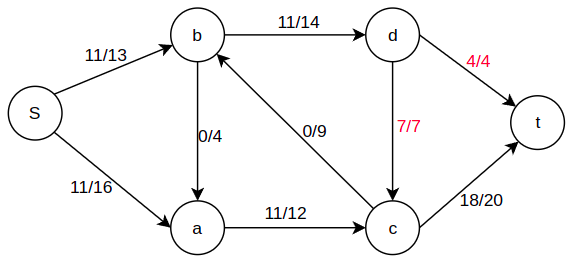


a.

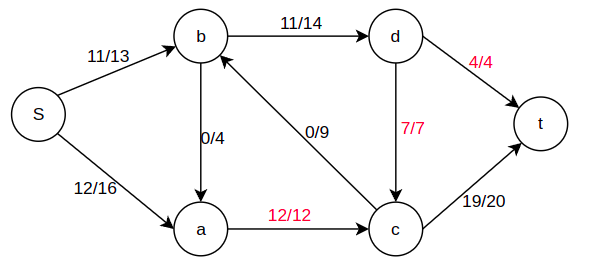
Continue from the current flow

Find a path from s→t (ignore the max flow edges): s→b→d →c→t

Increase the flow on that path:



The next path: s→a→c→t



Until no path found, end the loop, the maximum flow is 23

b. The min cut is: (a,c)-12 (d,c)-7 (d,t)-4

Total:23

2.

a. We can construct a bipartite graph G = (P, D, E)

with two sets: P and D

P denotes as person, D denotes as night, E is the set of edges in Si that means person pi is able to cook the dinner dj ,the edge is (pi, dj)

Assume there is no feasible dinner scheduler, that means there is at least 1 dinner cannot be done by anyone. In that case, at least there is a missing edge indicate on a vertex in D, there’s no way a matching perfect is constructed like that.

b. The problem here: there is a missing edge night (pi, dk) OR (pj, dk), we need to find a match person to cook the dinner dk

First step, of course if (pi, dk) or (pj, dk) edges are feasible, then we found a perfect match. Otherwise, need to iterate all edges in “almost perfect match” scheduler, this take O(n) time.

Each edge, Ex: (pu, dv) check whether there are feasible edge (pu , dk) and [edge (pi , dv) or edge (pj , dv)]

If match, then we can change (pu, dv) to (pu , dk) and [edge (pi , dv) or edge (pj , dv)]

4.

It can be verified the result in polynomial time, it’s NP.