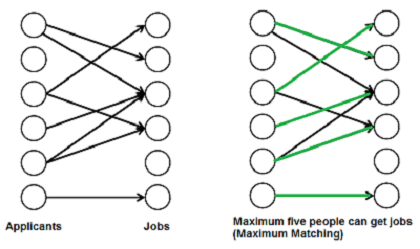


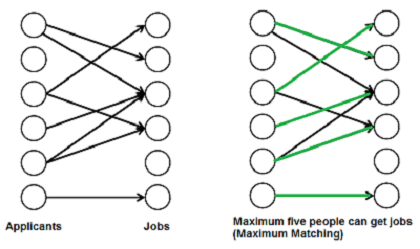
Program 0

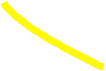
Motivating Example: Consider matching partners. In the graph below, nodes on the left represent employers. Nodes on the right represent job applicants. Arcs indicate acceptable matches, and green arcs represent the final matching. We say that a matching between N people is stable if no pair of unmatched pairs prefer each other to their matches.





If there was a preferences between the choices (with 1 being the best), the left matching (shown in yellow) is preferred.





The Gale-Shapley algorithm is one solution to the matching problem when there are preferences

**Matching**

A round of the Gale-Shapley stable matching algorithm proceeds as follows:

* In each round, one or more employers with open job positions each make a job offer to the applicant they prefer, among the ones they have not yet made an offer to.
* Each applicant who has received an offer evaluates it against their current position (if they have one). If the applicant is not yet employed, or if they receive an offer from an employer they like better than their current employer, they accept the best new offer and become matched to the new employer (possibly leaving a previous employer with an open position). Otherwise, they reject the new offer. \*\*Note, rejecting a previously accepted offer is BAD practice in the real world.
* This process is repeated until all employers have either filled their positions or exhausted their lists of applicants.

.

At this point, the algorithm terminates and returns the current tentative matches. Note also that the algorithm could proceed with either the employer or the applicants proposing; the difference is cosmetic.

You have been given the python code for this algorithm. It assumes the employers do the proposing, but you can easily run the version with the applicants proposing by switching the order of the parameters.

We would consider this to be a multi-agent solution. For simplicity of coding, each proposer takes turns, but you can imagine a parallel version where all proposals in a round occur simultaneously. Each agent is making decisions based on what is good for them.

Gale-Shapley algorithm returns the best-possible stable matching for each member of the proposing side. Consider the following example preferences where lower-case labels are employers and upper case labels are applicants:

Applicant Preferences Employer Preferences A picture containing text, furniture, chest of drawers

Description automatically generated

As it happens, this set of preference profiles has three distinct stable matchings; the first two are obtained from the Gale-Shapley algorithm (with employers and applicants proposing, respectively). The subscripts in the following denote the preference rank of the individual’s match (i.e., A1 means that A is matched to her first choice):

|  |  |  |
| --- | --- | --- |
| A group of letters on a white background  Description automatically generated |  |  |
| Matching 1 (employers proposing) | Matching 2 (applicants proposing) | Matching 3 (alternative stable matching) |

In general, the Gale Shapley algorithm always returns a stable matching that is optimal for the proposing party.

Output

1. Change the output to indicate the sum of all preferences achieved for each group. In Matching 1, the employes achieved (2+2+1+1 = 6) and the applicants achieved (3+3+3+3=12)
2. Create an example for which the results are the same, no matter who proposes.

Hint:

For Pycharm, the files need to be in the same directory as the code.