Danial Rizvi
DR28944

Lab 1 Report

Part 1

a) Assuming half the jobs are full-time and half the workers are hard-working let us prove by contradiction that in every stable matching a hard worker receives a full-time job.

Assume that a matching contains at least one pair where a hard worker receives a part-time job. In order for this to occur at least one lazy worker must have received a full-time job. If *h* = hard worker, *f* = full-time job, *l* = lazy worker, and *p* = part-time job, then this matching contains the pairs (*h, p*) and (*l, f*). This is an unstable match because hard worker *h* prefers full-time job *f* over its current part-time job *p* AND full-time job *f* prefers hard worker *h* over lazy worker *l*. Both hard worker *h* and full-time job *f* have motivation to leave their current pair and thus such a matching will always result in unstable pairs. Therefore, we have proven by contradiction that every hard worker must receive a full-time job if half the jobs are full-time and half the workers.

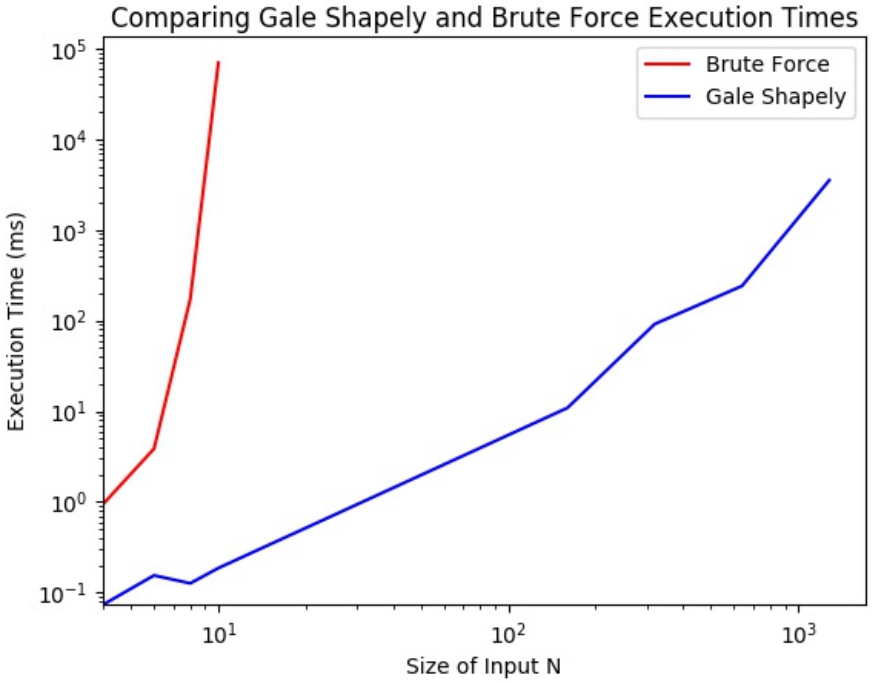
b) The run-time complexity of the brute force algorithm in Big O notation is $O(n!n^2)$. This is because the brute force algorithm checks for all n! permutations of possible matchings and requires n^2 iterations for each matching to determine whether it is stable or not.

c)
```
WHILE(there remains an unmatched worker)
  FOR(each worker W 0 to n)
    WHILE(worker is unmatched AND has not applied to every job J 0 to m in its preference list)
      IF(Job J in preference list is open)
        Match worker W and job J
        Make W a matched worker
      ELSE IF(Job J is taken AND Job J prefers worker W to its current worker W')
        Match worker W and job J
        Make W' an unmatched worker
```

d) This version of the Gale-Shapely Stable Marriage algorithm will always terminate in at most n^2 iterations because in the worst-case scenario every worker will have to apply to every job which becomes n^2. The algorithm is correct because every worker will apply to each job in the order of their preference list and thus receive the most preferable job that they can receive. Similarly, every job will begin by accepting the first worker that applies to their job and will reject their current worker if a preferable worker applies to their job. In this way, all of the matchings will be stable since both workers and jobs will receive the most optimal pairings possible.

e) The run-time complexity of this efficient algorithm in Big O notation is $O(n^2)$. Again this is because the algorithm will always terminate in at most n^2 iterations because in the worst-case scenario every worker will have to apply to every job.

f) From the graph below it is quite easy to see the differences in run-time between the Brute Force and Gale-Shapely algorithm implementations. This is because the brute force algorithm has an exponential/factorial growth while the Gale-Shapely algorithm has polynomial growth.



Part 2

Please see code in the dr28944_Rizvi_Danial folder.

Part 3

Please see code in the dr28944_Rizvi_Danial folder.

Part 4

a) No, there does not always exist a perfect matching with no weak instability. This can be disproven by a simple counter-example. Assume the worker/job preferences below:

	Job 1	Job 2
Worker 1	1	2
Worker 2	1	2

Worker's Preferences

	Worker 1	Worker 2
Job 1	1	1
Job 2	1	2

Job's Preferences

In this scenario both possible matchings lead to a weak instability. In the first possible matching [(W1, J1) , (W2, J2)] Worker 2 prefers Job 1 over Job 2 but Job 1 does not have a preference

over the two workers. In the second possible matching [(W1, J2) , (W2, J1)] Worker 1 prefers Job 1 over Job 2 but Job 1 again does not have a preference over the two workers. Thus, there will not always exist a perfect matching that does not have a weak instability.

b) Yes, there does always exist a perfect matching with no strong instability. This is because the Gale-Shapely algorithm will still try to optimize both groups (Jobs and Workers) preferences so that workers and jobs receive the most optimal/preferred pairing that is possible. Thus, there will always exist a perfect matching with no strong instabilities.

c) The algorithm from part 1c will remain the same except that now each worker will try to be matched according to the new preference list format to accommodate for ties. For example, if there are 5 jobs and Worker 0's preference list is {1, 3, 2, 0, 1}, then Worker 0 will "apply" to jobs in one of the following orders: {J3, J4, J0, J2, J1} or {J3, J0, J4, J2, J1}. This is because Worker 0 has an equal preference for Job 0 and Job 4. Additionally, if Worker 0 applies to Job 3 and Job 3 does not have a preference between its current worker and Worker 0 then Job 3 will NOT break its current matching and Worker 0 will apply to the next job on its preference list. This is to avoid entering an infinite loop where Job 3 keeps switching between two workers.