Stable Marriage

Samuel Bush

September 2020

1 Questions

1. Explain SM or a popular variant of interest and why it is important in two paragraphs.

   (a) The stable marriage problem is essentially when you have to match up any two lists of things in order of preference to each other. So if you have for example a list of adult males and a list of adult females and you had to figure out how to best match them according to their individual preference to each other you would use the Gale-Shapley algorithm. This algorithm will match pairs until all pairs are stable. A stable pair is when each of the individuals that have been paired up do not have a potential partner that is higher on there preference list that also that same individual higher on their preference list than the person they are already paired to.

   Why it's so important is because there are a number of situations in life where we need to pair potential matches using preferential lists with the best overall fit for each pair. Such as hospital residency programs or assigning users to servers for internet services. This will ensure that the performance will be better overall for everybody instead of just a few people.

2. Give the original Gale-Shapely algorithm for SM in pseudo code that you write, and use your algorithm to explain its $O(n^2)$ complexity.

   (a) Each item in list A asks to pair with their top preferred choice in list B.

   (b) Items in list B accept pair and decline all other pairs that are lower than on Item B's preferential list than the pair that item B is already paired to.

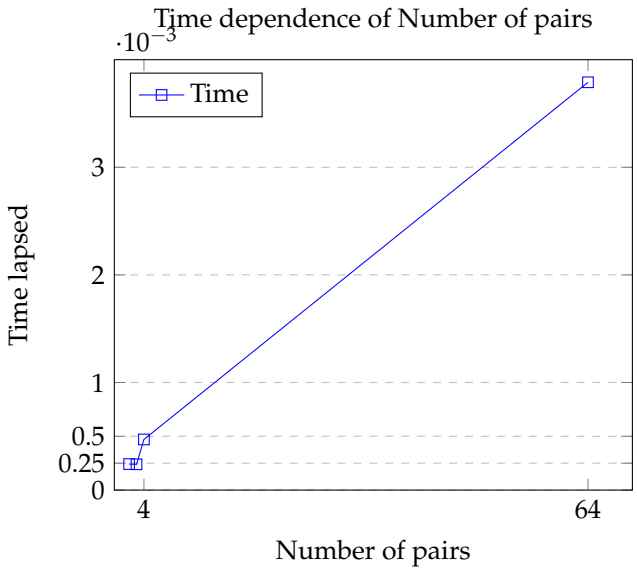
   (c) Each unpaired item in List A ask to pair with their next preferred choice in list B

   (d) Repeat until all paired.

The reason why the algorithm has $O(n^2)$ complexity is because each item in the list has their own list and must iterate through the list if they didn't find a match right off the bat. Worse case scenario the algorithm is $O(n^2)$ complexity because of the nested iteration.

3. Implement and benchmark the Gale-Shapely algorithm on vanilla SM.

   (a) See "GS1.py"



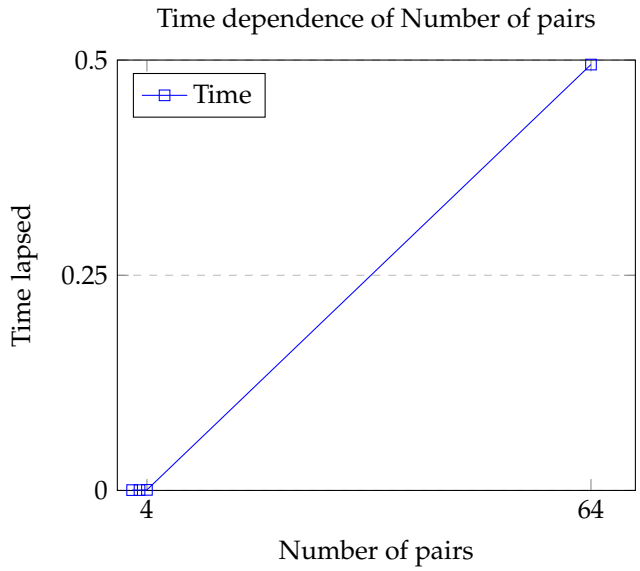
4. Explain why SM variants that allow couples or ties are NP-Hard. List the major variants and reductions used to prove NP-Hardness. Big Note: Do not do the reductions, just list what they are.

   (a) When you take into account Stable Marriage variants that allow couples or ties are NP-Hard because they add an additional step to the complexity that would otherwise be $O(n^2)$. Allowing ties or couples allows for many more possible solutions meaning that finding the right solution gets even more complicated. Variations of Stable Marriage are NP hard because if you can find an efficient way to solve any one of the Stable Marriage variations you will be able to solve the vanilla Stable Marriage efficiently as well. Vanilla Stable Marriage is already an NP-Complete algorithm so a variation of Stable marriage is at least NP-Complete but likely NP-Hard. Some variations of NP-Hard variations of Stable Marriage might not even have a polynomial-time algorithm.

   (b) Major variants include incomplete lists, ties in vanilla and ties on one side in Max Cardinality SMTI, roommates and intern with couples. Reductions include ILP, SAT and Constraint Programming.

5. There exist randomized algorithms and Integer Linear Programming (ILP) algorithms that are Fixed Parameter Tractable (FPT) for matching with ties or couples allowed (or scheduling jobs). Choose and implement and benchmark an FPT algorithm for this problem.

   (a) See "5 FPT.py"



2 Results

```
Time elapsed  0.0037970999999999977
Male " Matched with Female ~
Male # Matched with Female b
Male $ Matched with Female r
Male % Matched with Female i
Male & Matched with Female c
Male ' Matched with Female R
Male ( Matched with Female e
Male ) Matched with Female m
Male * Matched with Female }
Male + Matched with Female v
Male , Matched with Female z
Male - Matched with Female g
Male . Matched with Female Z
Male / Matched with Female {
Male 0 Matched with Female Y
Male 1 Matched with Female h
```

```
Male 2 Matched with Female x
Male 3 Matched with Female p
Male 4 Matched with Female |
Male 5 Matched with Female n
Male 6 Matched with Female _
Male 7 Matched with Female \
Male 8 Matched with Female V
Male 9 Matched with Female w
Male : Matched with Female y
Male ; Matched with Female S
Male < Matched with Female P
Male = Matched with Female o
Male > Matched with Female s
Male ? Matched with Female ]
Male @ Matched with Female u
Male A Matched with Female X
Male B Matched with Female Q
Male C Matched with Female ‘
Male D Matched with Female l
Male E Matched with Female j
Male F Matched with Female q
Male G Matched with Female a
Male H Matched with Female k
Male I Matched with Female W
Male J Matched with Female [
Male K Matched with Female t
Male L Matched with Female ^
Male M Matched with Female U
Male N Matched with Female d
Male O Matched with Female f

Time elapsed  0.49479739999999994

Male list before [['"', ['x', '|', 'y', 'q', 't', 'm', 'p', '‘', '{', 'v', '[', 'V', ']', 'S
'l', 't', 'y', 'U', '[', 'k', 'm', 'Q', 'V', ']', 'n']], ['I', ['r', 'w', 'W', 'Q', 'x', 'Z'
'g', 'Q', 'i', 'w', '‘', '[', 'V', 'x', '~', 'o', 'c', 'U', ']', 'Y', 'n', 'd']], ['L', ['‘'
Female list before [['P', ['&', ';', 'B', "'", '"', '0', '#', 'K', 'I', 'C', '+', 'J', '>',
'-', "'", 'A', '0', ',', 'F', 'H']], ['^', ['#', '+', '<', '-', "'", '8', '/', 'I', '6', '9'
"'", '2', '7', ',', 'I', 'F', '9', 'B', ':', '"', '%', 'L']], ['a', ['-', 'M', 'H', 'N', '2'
'G', '0', '6', '(', ',', 'E', 'H', '%', 'L', 'A', '<', '=', 'D', '*', '-', '$', ')']], ['d',
['6', '#', '8', 'H', '<', '.', '@', 'C', 'N', 'D', 'B', '(', '-', '=', ')', '4', '*', 'E', '
Deleted ['d', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't'
Male list after FPT [['"', ['P', 'Q', 'R', 'U', 'V', 'X', 'Y', 'Z', '[', '_', '‘', 'b']], ['
'T', 'U', 'Y', '\\', ']', 'c']], ['>', ['P', 'Q', 'Z', '_', 'a', 'b']], ['?', ['Q', 'R', 'S'
'V', 'X', '[', '\\', 'a', 'b']], ['D', ['T', 'V', 'Y', 'Z', '[', 'b', 'c']], ['E', ['P', 'T'
['O', ['T', 'V', 'Y', 'Z', '_', 'b']]]
```

```
Female list after FPT [['P', ['&', ';', 'B', "'", '"', '0', '#', 'K', 'I', 'C', '+', 'J', '>
```