Patient-Drug Solution Algorithm

This algorithm works by building a ragged array representation of all of the possible combinations of drugs. In each cell of this array is a list of all of the rows of the matrix that can potentially make this combination. Additionally a corresponding structure is built that keeps track of the number of times each combination has been used

This algorithm works in 3 steps:

1. Build the ragged array structure that this algorithm processes

2. Make initial combination selections. Ie for each patient, choose 2 drugs to specific criteria

3. Make the remaining selections for each patient

The details of each step are as follows:

1. Build a ragged array representation of all of the possible combinations of drugs. In each cell of this array is a list of all of the rows of the matrix that can potentially make this combination. Additionally a corresponding structure is built that keeps track of the number of times each combination has been used

2. The combinatorial matrix is iterated over looking for various criteria to make its initial selections. In particular making initial combinations out of drugs have not been chosen is prioritized highest (even if they exist in different; e.g. it would try to avoid doing D1D3 if D1D2 were already made). If completely impossible to avoid it will allow 1 previous use of a drug, then 2, etc. Its second priority is to make combinations as unique to a patient as possible. That is, all other things equal, if combination D_1D_2 can be made by P_1 and P_2, but combination D_1D_3 can only be made by P_1 it will choose P_1's initial pair to be D_1D_3.

3. The remaining selections for each patient are chosen by iterating through the combinatorial structure, considering each combination P1 could make, and determining if that would result in the maximum number of combinations.

The time complexity of each step is:

1. **$O(P*D^2)$**
2. **$O(P^3*D^2)$**
3. **$O(M^4*P*D^2)$**

Where **P** is the number of patients, **D** is number of drugs, and **M** is maximum number of drugs per patient.

Each section is at least $\mathbf{O(P*D^2)}$ because step will iterate through every combination of drugs for each patient. Obviously initialization doesn't do anything beyond this so it has nothing extra to add on.

Step 2 has an additional $\mathbf{P^2}$ because of the criteria it uses to determine what the initial pair is. Because it tries minimize the overlap between drugs (not just pairs) and the number of people who can make that pair, it could in the worst case have to iterate through $\mathbf{P^2}$ times. However this could only be the case if the number of potential drugs for each patient is extremely low and **D** is extremely large compared to **P**.

Step 3 is heavily dependent on **M** because it iterates through all of the remaining choices; that is, selections number 3, 4, …, **M** for each patient. It then tries to maximize the number of new combinations possible which, for selection number **k** could be at most **k**. It will first iterate through the combinatorial structure seeing if it could make a match of that would result in **k** new combinations, then **k – 1**, **k – 2**, …, **1** new combinations. Finally for every one of these iterations it builds the list that contains all possible combinations for that iteration, which is an $\mathbf{O(M^2)}$ operation.