Prob. 1	Prob. 2	Prob. 3	Prob. 4

Problem 1.

In the model, we are trying to match phones and base stations, so this problem can be reduced to well-established Hungarian algorithm with some additional constraints.

Here, we can think base stations and phones as a bipartite graph because there are no connections within base stations or phones. Moreover, each phone must be assigned to a distinct base stations. That's why we are searching for one-to-one match. We will use Hungarian algorithm as a black box to find this matching but it can also be solved by Ford-Fulkerson if we add virtual source and sink nodes and think the problem as a min-cost max-flow problem.

Another constraint is distance limitation because if a phone is far away its assigned base station more than given constant c, this assignment isn't valid since phone can not utilize of base station service due to the distance. Therefore, we have a threshold cost model. If a phone is far away from a base station more than given c, then the cost is set to very high value. Otherwise, cost is basically zero because we don't differentiate the cost under distance c.

Initial setup requires us to calculate the distance of each phone to each base station. This can easily be done via two nested for-loops which is $O(n^2)$ complexity. This is fine since it will be dominated by Hungarian algorithm which has a higher complexity. While calculating distance, we also create a cost matrix in the threshold model, 0 or $+\infty$. Cost matrix, in addition to distance calculation, can only increase the multiplier in front of n^2 so complexity is same, $O(n^2)$.

After calculations of distances, we run Hungarian which has a complexity of $O(n^3)$ for n phones and n base stations. If we can get n assignment, we start to driving. Otherwise, *fully connectedness* can not be satisfied so we immediately return false. Throughout the driving, for each unit of driving, we check the distance between the moving phone and its assigned base station. If the distance is below c, there is nothing to do and we continue with next unit of travelling. However, if distance is more than c, we need to change the assignment of the moving phone since it can't be serviced by its previous base station any more. In the cost matrix, we need to update the distance of moving phone to other base stations which can accomplished in $O(n)$. Then, we again run Hungarian. If it can find a n assignment, we can go further, otherwise we return false. This procedure is repeated k times for k units of distance. Finally, we return true.

In short,

$$\textit{Total Cost} = \textit{Cost Setup} + k \cdot \textit{Move} = O(n^2) + k \cdot O(n^3) \leq O(n^3)$$

$$\textit{Move} = \begin{cases} O(1) & \text{if distance } d \text{ is lower than } c \\ O(1) + O(n) + O(n^3) & \text{if distance } d \text{ is higher than } c \end{cases}$$

Problem 2.

First we need to prove that in any man-optimal stable matching, each woman has the worst partner that she can have in any matching (man-optimal = woman-pessimal).
Suppose the above assumption does not hold. Let M_0 be the man-optimal stable matching and suppose there is another stable matching M_1. Let the woman w be matched with man m_0 in M_0 and with m_1 in M_1. Suppose that w prefers m_0 to m_1. Then it is obvious that (m,w) blocks M_1, since m_0 has no other better stable partner than his partner in M_0.
The same way it can be proved that a woman-optimal matching is man-pessimal.
Thus if a matching is both man-optimal and woman-optimal, it is implied that the man-optimal version of the stable matching and the man-pessimal version of the stable matching are equal and so are the woman-optimal and the woman-pessimal versions. But since the pessimal and the optimal matching are the same that implies that there can be only one matching.

Problem 3.

1. Strong instability
   Yes, there always exists a perfect matching without any strong instability.
   An algorithm in polynomial-time that can guarantee such a matching is actually the "man proposes, woman disposes" we've seen in class.
   This algorithm has a wort-case number of rounds of $n^2 - 2n + 2$ so it's running in polynomial-time.
   In the case of ties, we may end up with m prefering w than its final matching and w having m at the top of her ranking. But it means that the final matching of w is a man she likes as much as m not less otherwise it means than m has never proposed to w which contradicts the way the algorithm works.
   Thus with the "man proposes, woman disposes" algorithm we can never have a strong instability.

2. Weak instability
   First we can see that the algorithm used for avoiding strong instabilities doesn't work for weak instability. The explanation of the correctness used before actually shows a weak instability.
   But anyway there isn't any algorithm that can avoid weak instabilities for sure. For instance if you have m_1 and m_2 both prefering w_1 than w_2 and w_1 liking m_1 and m_2 equally (we don't even need to know the preferences of w_2). The two only possible matching without even taking rankings into account would be:

   (a) $m_1 \longleftrightarrow w_1$ and $m_2 \longleftrightarrow w_2$

   (b) $m_1 \longleftrightarrow w_2$ and $m_2 \longleftrightarrow w_1$

   In (a) we can see that there's a weak instability because m_2 prefers w_1 to his matching and w_1 likes m_1 and m_2 equally.
   In (b) there is also a weak instability because m_1 this time prefers w_1 to his matching and w_1 likes m_1 and m_2 equally.
   Thus there is no solution without a weak instability.

Problem 4.