Part 2 - Anesthesiologists allocation optimization

*Run Part2.py to run this.*

**Room Allocation:**

In favor of choosing a room for each surgery, since we only have two limitations, that there will not be two surgeries at the same time in the same room and that we do not exceed 20 rooms, a simple algorithm that accepts the hours of a surgery, and puts it in the first available room will do the job, it is important to be consistent with the order of the rooms, that is, if we have 3 surgeries 7:00-8:00, 8:00-9:00, 7:00-9:00, if we do it out of order, we will put 8:00-9:00 in room 1, 7:00-8:00 in room 2, then we will not be able to put 7:00-9:00 in one of the rooms and we will have to open a new room , but if we always try to put the surgery in room 1 first, then we will always succeed in getting the above two surgeries into two rooms.

**Anesthesiologists Allocation:**

In order to produce the cheaper work arrangement I tried several approaches, as a start I created a work arrangement with the following logic:

Take an surgery, and calculate what its cost will be if you give it to anesthesiologist number 1, for example if the anesthesiologist has already been working for an hour, and we give him another one hour surgery right after, the cost is zero because we are already paying him for five hours anyway, if he has already been working for eight and a half hours and we will add an one hour surgery (immediately after the last surgery) the cost is 1.25 because for the first half hour he will receive a normal price and for the second half hour he will receive additional hours.

If he already works 11.5 hours, and we add another hour to him, the cost is infinite because it is forbidden to exceed 12 hours.

Now we will check it on all the anesthesiologists, and we will choose the cheapest anesthesiologist, for example if there is someone with 6 hours and there is an gap between their work and the additional (one hour) surgery of half an hour then the cost is 1.5 but if there is someone with an interval of only fifteen minutes, the cost is 1.25, and we'll prefer him.

If the cost is above a certain threshold, we will call for a new anesthetic, that is, if we enter a 3-hour analysis and see that the cost is more than 3+a threshold, we will create a new anesthetic.

The algorithm produced reasonable results but not good enough, especially towards the end of the day many anesthesiologists remained with shifts shorter than five hours which increased the costs.

In order to improve the result, I tried several directions, among them an algorithm that checks all the possibilities intelligently, but it only worked in a reasonable time until 17-18 surgeries, after that there were too many possibilities (once the algorithm detects a direction is not possible, for example that two surgeries at the same time for the same anesthetist, he does not continue to check the next possibilities of this direction. And still the amount of tests is very large and it is impossible to finish all 114 surgeries)

Other options I checked are to take the arrangement I created earlier and try to match between anesthesiologists, that is, if it is possible to combine two anesthesiologists to become one anesthesiologist and thus lower the price (for example, two anesthesiologists who received three hours and are adjacent to each other, their cost will drop from 10 to 6) and thus look for the best match every time and progress one by one, in a similar way I did by moving each time one surgery for one anesthesiologist to another, each time choosing the transition that improves the cost function in the best way (similar to Gradient Descent) until we reach a situation where it is impossible to progress, in principle it could work if the price function was convex but both methods did not work well.

Since from what I saw the main problem was at the end of the shift, I chose two approaches. One direction is that as I previously created the work arrangement from left to right (from seven in the morning onwards) so now I worked in both directions, I took the first half of the surgeries and created a work arrangement for it, and then I took the other half and created a work arrangement for it when I start from the end to the beginning, inserting the most recent surgery and then the one before it, so I had two work arrangements that one, its biggest issue is mainly at the end of his time (about the middle of the day) and the other his problem is mainly at the beginning of his time (also about the middle of the day) and if there are two anesthesiologists with shifts shorter than five hours It will be possible to match them. I checked all the possible matches (possible: does not exceed 12 hours, and the price of an anesthesiologist that is the result of a match of two anesthesiologists whose cost is greater than the cost of both - will also be disqualified) and then I went through all the combinations of the matches to get the combination that would lower the price in the optimal way. This improved the results somewhat and I came to a total cost of 164.75.

A second option I tried is with dynamic programming, I created the work arrangement as explained above, then using dynamic programming I started to change my last choice, for example if we reached a certain cost by the choices as explained above, we go a step back and instead of choosing the ideal anesthesiologist we choose the anesthesiologist number 2 - that is, the anesthesiologist that costs almost the lowest and then check if it lower the cost function, and thus check all the anesthetics from best to worst (not including impossible directions such as shift over 12 hours) then go back another step and again try the choices from best to worst . If during the progress we see that we have already passed the minimum price we have achieved so far this direction is also rejected. This approach improved the results somewhat, theoretically if it ran for very long time it would reach the optimal solution, but in practice it brought a reasonable result of 164.5. (I stop running after a few seconds, even trying to run for 5 hours did not improve the result)

I guess the right direction is linear programming but I only have basic familiarity with it at the moment and I didn't know how to use it to carry out the current problem.

The following figures are:

1. A shift that is calculated in both directions - from the morning towards the afternoon and from the afternoon towards the morning.

2. The same shift after matches (only one match was suitable).

3. The result after dynamic programming.





