Since the goal of this problem was to get the highest possible armor for the amount of crowns that we had, I approached this problem by first getting the highest armor we could get and reduce the cost armor piece by armor piece until out financial constraint was met.

One of the things that I was constantly thinking of when working on the solution to this problem was the scale. Regardless of how many armor categories we need to work with, I wanted to make sure this wasn’t a problem, so with this in mind I didn't want to go the route of brute-forcing my way into the solution because we could potentially be working with many different armor categories and many items within each category.

Instead, I began by first figuring out how many armor categories we are working with and then categorizing each armor piece into their respective category. The end result is a list of lists where each sub-list represents an armor category, additionally I sorted each of these lists according to their armor value and cost. This would mean that the highest armor available for a category would appear at the end of the list and would therefore allow me to view the dataset as a list of priority queues insteads of a list of unordered lists where the next item in the queue would be the next best piece of armor in terms of its armor value.

Next I’d construct geralt’s armor by popping the queues of each armor category. This would mean that I currently have the highest possible armor in the dataset. I performed a quick check to see if we meet out financial constraint. if not then I’d begin to reduce the cost until we’ve met the constraint.

When reducing the cost of our armor set, the most important factor of this process is selecting which piece of armor should be swapped out. So before we begin this process I made sure to filter out the queues such that no armor in the queue would have a greater cost that the one we currently have equipped. Once this condition was met, I began the selection process. The selection process is based on the difference in value of the ratio between armor value and armor cost between the armor piece we currently have equipped and the armor piece next up in the queue. After calculating the difference for every armor piece in our currently equipped set, the smallest difference value would determine which armor piece would be swapped out because the smallest difference would result in the least armor value deviation from what we currently have. This approach works because since we start off with the best possible armor set, deviating as little as possible after each iteration in the loop would mean we have the best possible armor for the money.