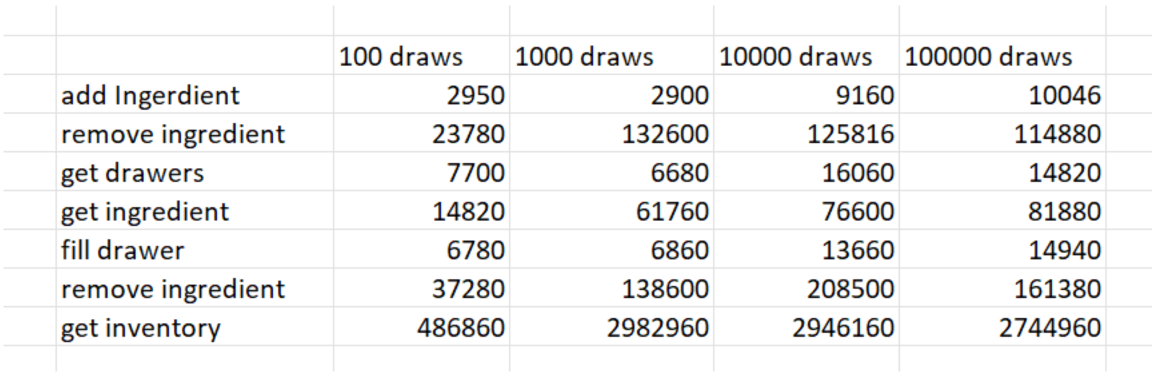
cosc201 Assignment 2

To the board of interdimensional Potion Makers Authority. I have a system that would work wonderfully for your potion management system. I have a potionarium that keeps track of the ingredients in each draw along with very useful methods such as addingredient(), filldrawer(), removeingredient(), removeingredients(), getdrawers(), getIngredients(), and getInventory. Every method is constant time per item excluding the time to shallow copies to protect the potionarium from unauthorized access. The only methods that is not constant time are filDrawer() and removeIngredients(), as they process a whole set and the time taken will depend on the size of the set passed in. However, as they process an entire set, they achieve constant time per item. To achieve constant time, I have traded space for time and made 2 hash maps for Drawers and inventory which are inversed of each other. Where drawers include what ingredients, each draw has, and Inventory includes all the ingredients and what draws they are currently in. This way, we can change getInventory() to constant speed and getDrawers() to constant speed as well unfortunately, the method itself is not constant time as we include a copy which is not constant time, so we trade time complexity for security. I have done error checking for things such as if u call getIngredients() on a drawer that does not exist, it will return null. I have also made it that when the last items of a draw or inventory is removed, the draw or inventory is removed and set to null so we don't need to allocate memory for items that are empty. I have tabled the speed of the methods as shown below, this is done by generating 100 – 100,000 draws each with up to 50 random ingredients with the ingredients being words generated by 3 random characters. timing how long it takes for each. The numbers represent how many nanoseconds were used to perform the action where there is already an x number of draws.

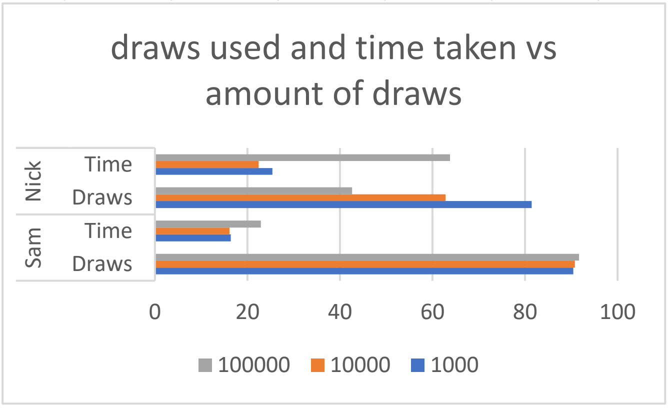


	100 draws	1000 draws	10000 draws	100000 draws
add Ingerdient	2950	2900	9160	10046
remove ingredient	23780	132600	125816	114880
get drawers	7700	6680	16060	14820
get ingredient	14820	61760	76600	81880
fill drawer	6780	6860	13660	14940
remove ingredient	37280	138600	208500	161380
get inventory	486860	2982960	2946160	2744960

As seen above, all the methods work very fast and get inventory being the only one that gets to the milliseconds. But they are not constant time due to the shallow copy of things such as the output of getdrawers(), getIngredients() and things such as a copy of the input of ingredients not fill drawers. This way if the set is changed outside of the potionarium, the potionarium itself will not be affected. This potionarium is a very fast and efficient and is perfect for a group of potion masters like yourself. It is very fast, efficient and secure. One thing that I could add if the board wanted was a way to sort the ingredients or some sort of structure where ingredients used together commonly or items that appear quite frequently would be grouped together which will make the potion masters report easier as they will access less draws as the ingredients are bundled together.

For Potion Masters, I have made an algorithm that is more effective for you guys compared to the standard Sam potion Master class. I have made a Novel Nic that instead of getting the lowest number drawer, we collect from the draw with the most matching items in the ingredients list. This means that Novel Nic will overall go through less draws as it will find draws that include ingredients from the ingredients list passed in. I have simulated the situation shown below with different amounts of draws where each draw contains up to 50 ingredients and the ingredients being randomly generated 3-character words and the ingredients list being a sub list size 100 of the inventory so no missing reports are returned.

		1000	10000	100000
Sam	Draws	90.4	90.8	91.7
	Time	16.4	16.1	22.9
Nick	Draws	81.4	62.8	42.6
	Time	25.4	22.4	63.8



On the table, the top numbers of 1000, 10000, 100000 refer to the amount of filled draws in the potionarium. Draws refer to how many draws were opened for a ingredient list of 100 and Time would be the time taken in milliseconds to process the report. This is data is performed 10x and averaged out to reduce the overall variance in the data.

As seen from the graphs above, Sam remains constant with the amount or draws accessed and time taken however Nic accesses less drawers at all stages of the experiment and with the more draws the better Nic performs, though is also slower compared to Standard Sam. This is the tradeoff for reducing the number of draws. I think for this context having the lower number of draws is significantly more important. As the increase in time is yet to even hit a second compared to the time saved for saving potentially up to more than half of the draws used where there are 100,000 draws. Which on average saves the time of a potion master going through around 50 draws. Which makes up for the small time increase of Nic. Both potion masters will do everything else the same, where they will print missing reports when ingredients are missing, remove from the correct draw when a report goes through, etc. In this experiment, I used 3-character words that are randomly generated and placed them into a draw without context or frequency. If the potionarium were to have a context or a frequency of ingredients. Novel Nic would be even faster. As similar used items may be grouped together and novel Nic would be able to access that draw immediately decreasing the draws even more. Whilst Standard Sam would still be searching from the beginning which may not be grouped.

Overall, I hope you can retain me as the back end of the management system. In future I can create more potion masters that may have different specifications e.g. a potion master that creates a report as fast as possible instead of one that accesses less draws. Along with other potion masters that are requested by the board.