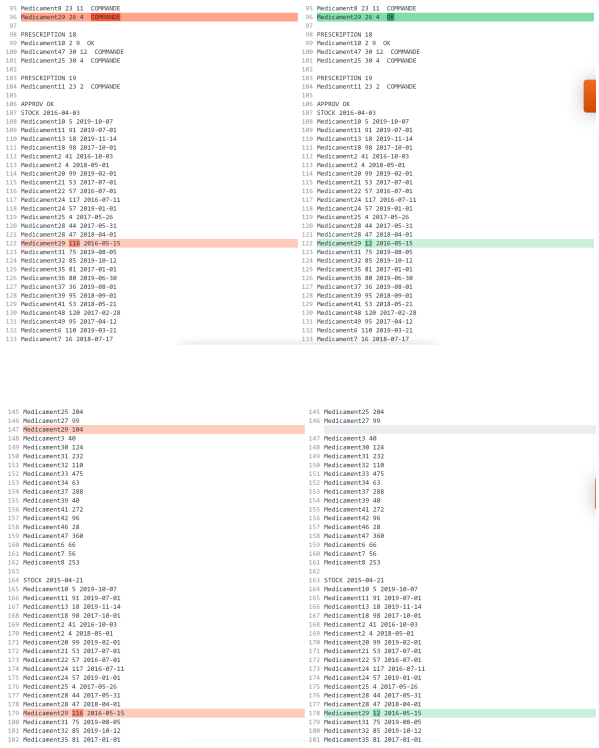
Rapport

Jean-emmanuel Chouinard (20246807), Timothe Payette (20239892)

10 juillet 2023

1 Auto-évaluation

Notre programme fonctionne comme prévu. Il respecte les consignes et le format des sorties demandé. Le seul exemple où notre programme ne concorde pas à 100% est l'exemple 6. Nous jugeons cela normal puisque le stock donné et la date courante permet d'utiliser ces médicaments. Voici des images de ces différences



2 Analyse de la complexité temporelle (pire cas) théorique en notation grand O

2.1 Légende

Voici la légende que nous allons utiliser tout au long de l'analyse:

- n - indique le nombre de types de médicaments différents en stock
- m - indique le nombre d'items sur la prescription
- k - indique le nombre d'items sur la liste de commande
- l - indique le nombre de médicaments à ajouter au stock

2.2 Prescription

Voici notre algorithme pour la fonction "Prescription"

```
for (String partition : partitions) {
    boolean inStock = false;
    boolean inCommand = false;
    Medicament theMedInStock = new Medicament( nom: "none",  nbrParCycle: 0, nbrDeCycle: 0,currentDate);
    Medicament theMedInCommand = new Medicament( nom: "none",  nbrParCycle: 0, nbrDeCycle: 0,currentDate);
    if (partition.startsWith("PRESCRIPTION")) {
        continue;
    }

    String[] med = partition.split( regex: "\\s+");
    Medicament prescription = new Medicament(med[0], Integer.parseInt(med[1]), Integer.parseInt(med[2]), currentDate);

    for (Map.Entry<String, Medicament> entry : stockMed.entrySet()) {
        if (prescription.getNom().equals(entry.getValue().getNom()) &&
                prescription.getNbrDeMed() <= entry.getValue().getNbrDeMed()) {
            inStock = true;
            theMedInStock = entry.getValue();
            break;
        }
    }

    for (Map.Entry<String, Medicament> entry : commandeMed.entrySet()) {
        if (entry.getValue().getNom().equals(prescription.getNom())) {
            inCommand = true;
            theMedInCommand = entry.getValue();
            break;
        }
    }
```

```
    }

    if (inStock &&
            prescription.getNbrDeCycle() <= calculateDaysBetweenDates(currentDate, theMedInStock.getDateStr())) {
        theMedInStock.addNbrDeMed(-prescription.getNbrDeMed());
        if (theMedInStock.getNbrDeMed() <= 0) {
            stockMed.remove(theMedInStock.getKey());
        }
        outputText.append(prescription.getNom()).append(" ").append(prescription.getNbrParCycle()).append(" ").append(prescription.g
    } else if (inCommand) {
        theMedInCommand.addNbrDeMed(prescription.getNbrDeMed());
        outputText.append(prescription.getNom()).append(" ").append(prescription.getNbrParCycle()).append(" ").append(prescription.g
    } else {
        commandeMed.put(prescription.getKey(), prescription);
        outputText.append(prescription.getNom()).append(" ").append(prescription.getNbrParCycle()).append(" ").append(prescription.g
    }
}
writeToOutputFile( content: outputText + "\n");
```

On voit que pour chaque item de la prescription (O(m)), deux boucles sont faites: une sur les médicaments en stock (O(n)) et une sur les médicaments déjà dans la commande (O(k)). Notre fonction se réalise donc en $O(m * (n + k))$.

2.3 Approv

Voici notre algorithme pour la fonction "Approv"

```
public void approv(String parts) {
    String modifiedInput = parts.replaceFirst( regex: "\n",  replacement: "");
    String[] partitions = modifiedInput.split( regex: "\n");

    for (String partition: partitions){
        if (partition.startsWith("APPROV :")) {
            continue;}
        String[] med = partition.split( regex: "[ \t]");
        Medicament medicament = new Medicament(med[0], Integer.parseInt(med[1]),med[2]);

        // Nom + date est deja dans stockMed
        if (stockMed.containsKey(medicament.getKey())) {
            stockMed.get(medicament.getKey()).addNbrDeMed(medicament.getNbrDeMed());
            continue;
        }
        stockMed.put(medicament.getKey(), medicament);
    }
    writeToOutputFile( content: "APPROV OK\n");
}
```

On voit qu'une boucle passe à travers la liste d'ajout (O(l)). De plus, la fonction "contains" de l'arbre est utilisé pour trouver des éléments dans la liste de médicaments en stock (O(logn)). Notre fonction se réalise donc en $O(l * log(n))$.

2.4 Date

Voici notre algorithme pour la fonction "Date"

```
public void date(String parts){
    String modifiedInput = parts.replace( target: "\n", replacement: "");
    String[] partitions = modifiedInput.split( regex: " +");

    currentDate = partitions[1];

    if (commandeMed.isEmpty()) {
        writeToOutputFile( content: currentDate + " OK\n\n");
    } else {
        writeToOutputFile( content: currentDate + " COMMANDES :\n");
        for (Map.Entry<String, Medicament> entry : commandeMed.entrySet()) {
            Medicament entryValue = entry.getValue();

            writeToOutputFile( content: entryValue.getNom() + " " + entryValue.getNbrDeMed() + "\n");
        }
        commandeMed.clear();
        writeToOutputFile( content: "\n");
    }

    ArrayList<Medicament> toRemove = new ArrayList<>();
    for (Map.Entry<String, Medicament> entry : stockMed.entrySet()) {
        Medicament medicament = entry.getValue();
        if (medicament.getDateDate().isBefore(LocalDate.parse(currentDate))) {
            toRemove.add(medicament);
        }
    }
    toRemove.forEach(medicament -> stockMed.remove(medicament.getKey()));
}
```

On voit qu'une boucle permet d'imprimer toutes les médicaments en commande (O(k)). Ensuite, on vérifie si des médicaments ont expirés et on les suppriment de l'arbre ($O(n^2)$). Notre fonction se réalise donc en $O(k * n^2)$.