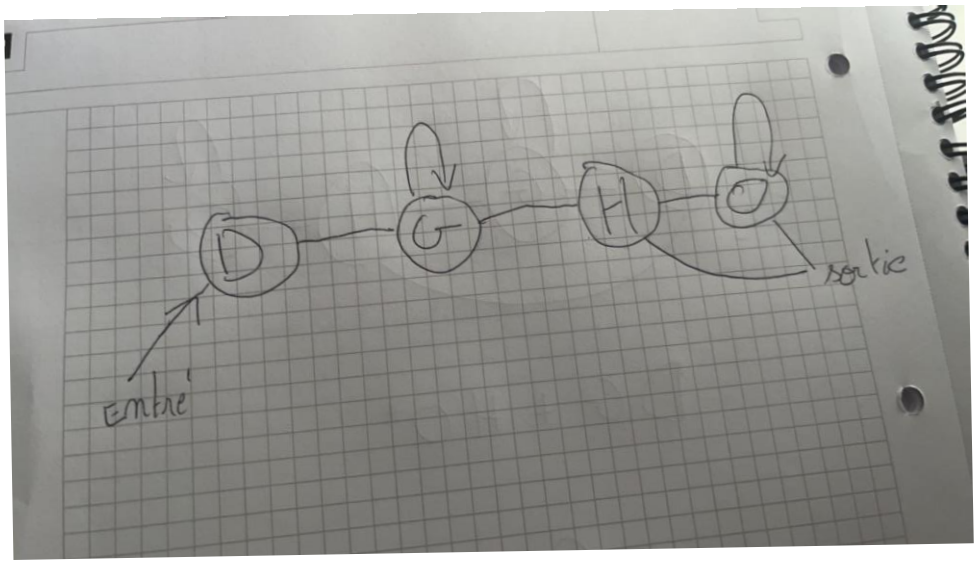
Projet ALgostruct 2:

Mehdi Oudghiri L2Y 20001692

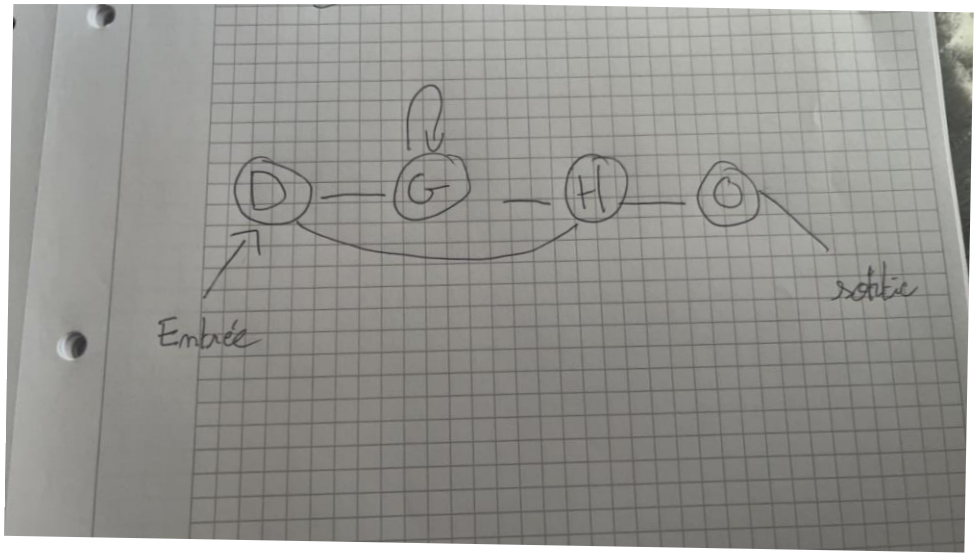
Etape 1 :

Motif DG+HO*



État	D	G	H	O
q0	q1	-	-	-
q1	-	q2	-	-
q2	-	q2	q3	-
q3	-	-	-	q3

Motif DG*HO?



État	D	G	H	O
p0	p1	-	-	-
p1	-	p2	-	-
p2	-	p2	p3	-
p3	-	-	-	p4
p4	-	-	-	-

```
Tableau : HDGGDGHDGDDGHGDGGGOHOODGDHDHOOGGOHHOODODDOHHGHOHOODOHDDHOGDHDHDDDHODOODOHHHODGGDDHOH
HODGDGDGODGOHDOGODGHHOGODOOGGHOOGDGHGGODGDOODOGDOHHGGODGHOHOGHOOHDGOGDOHDHGDGHDDDHHGHHODHGDOOO
HGOODDOOHOHODOHDDHOHDG
```

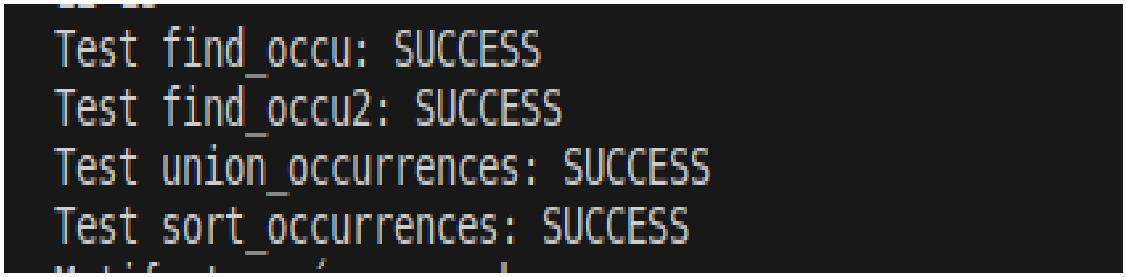
Etape 2 et 3:

```
Occurrences du motif DG+HO* :
4 7
10 13
101 104
117 120
138 142
159 162
Occurrences du motif DG*HO? :
4 7
10 13
24 26
26 30
54 57
58 60
60 62
64 67
80 83
101 104
117 120
138 142
156 158
159 162
164 166
171 173
194 197
```

Etape 4 et 5 :

```
Union des occurrences des motifs DG+HO* et DG*HO? :
194 197
171 173
164 166
159 162
156 158
138 142
117 120
101 104
80 83
64 67
60 62
58 60
54 57
26 30
24 26
10 13
4 7
```

Etape 6, 7 et 8:



Dans le code fourni, il y a plusieurs fonctions de test pour vérifier si les fonctions implémentées fonctionnent correctement. Voici une explication pour chacune d'entre elles :

1. test_find_occu() : Cette fonction teste si la fonction find_occu() trouve correctement les occurrences du motif "DG+HO*". Dans ce test, un tableau de test "DGHO" est créé, et on s'attend à ce qu'il y ait une occurrence du motif. Le tableau d'occurrences attendu est créé avec cette occurrence, puis la fonction find_occu() est appelée avec ce tableau de test. Le tableau d'occurrences renvoyé par find_occu() est comparé au tableau attendu pour vérifier si la fonction fonctionne correctement.
2. test_find_occu2() : Cette fonction teste si la fonction find_occu2() trouve correctement les occurrences du motif "DG*HO". Dans ce test, un tableau de test "DH" est créé, et on s'attend à ce qu'il y ait une occurrence du motif. Le tableau d'occurrences attendu est créé avec cette occurrence, puis la fonction find_occu2() est appelée avec ce tableau de test. Le tableau d'occurrences renvoyé par find_occu2() est comparé au tableau attendu pour vérifier si la fonction fonctionne correctement.
3. test_union_occurrences() : Cette fonction teste si la fonction union_occurrences() fonctionne correctement pour fusionner les occurrences de deux motifs différents. Dans ce test, deux tableaux d'occurrences (occurences1 et occurences2) sont créés. Le tableau d'occurrences attendu pour l'union de ces deux tableaux est également créé. La fonction union_occurrences() est appelée avec les deux tableaux d'occurrences, et le tableau d'occurrences renvoyé est comparé au tableau attendu pour vérifier si la fonction fonctionne correctement.
4. test_sort_occurrences() : Cette fonction teste si la fonction sort_occurrences() trie correctement les occurrences dans l'ordre décroissant. Un tableau d'occurrences non triées est créé, ainsi qu'un tableau attendu avec les occurrences triées. La fonction sort_occurrences() est appelée avec le tableau non trié, puis le tableau trié est comparé au tableau attendu pour vérifier si la fonction fonctionne correctement.

Ces fonctions de test permettent de s'assurer que les différentes fonctions du programme fonctionnent correctement et renvoient les résultats attendus. Elles sont appelées dans la fonction main() pour vérifier les fonctions avant de les utiliser dans le programme principal.

Etape 10 :

Le programme semble correct et devrait fonctionner. Pour déterminer la taille maximale du tableau sur lequel le programme peut fonctionner, vous devez exécuter le programme avec différentes tailles de tableaux comme indiqué (2'000, 20'000, 200'000, 2'000'000, 20'000'000, 200'000'000, etc.) et observer les performances et les problèmes potentiels.

La limite sera probablement déterminée par les ressources disponibles sur votre ordinateur, telles que la mémoire RAM et la capacité du processeur. Lorsque la taille du tableau augmente, la consommation de mémoire augmente également, car le programme doit stocker de plus en plus d'occurrences et de tableaux intermédiaires. Si la mémoire est épuisée, le programme pourrait se bloquer ou être interrompu par le système d'exploitation.

La capacité du processeur peut également limiter la taille maximale du tableau. À mesure que la taille du tableau augmente, le temps de traitement pour la recherche d'occurrences, le tri et l'union augmente également. Si le temps de traitement devient trop long, cela peut également être considéré comme une limite.

Pour déterminer la limite exacte, vous devez exécuter le programme avec différentes tailles de tableaux et observer les performances et la consommation de ressources. Vous pouvez également utiliser des outils de profilage pour surveiller la consommation de mémoire et la charge du processeur pendant l'exécution du programme.

Étape 11 :

Pour que le programme puisse fonctionner avec des tableaux beaucoup plus grands, il est recommandé de remplacer la constante TAB_SIZE par une variable allouée dynamiquement lors de l'exécution.