TP M2 BI
Transformée de Burrows Wheeler et FM-index

V.Levallois, C. Lemaitre, P. Peterlongo

2024

Tableau des suffixes

Pour ce TP, nous utiliserons une implémentation du tableau des suffixes disponible ici : `http://bioinformatique.rennes.inria.fr/tools_karkkainen_sanders.py` adapté de `http://code.google.com/p/pysuffix/`.

Voici un exemple d'utilisation :

```
from tools_karkkainen_sanders import simple_kark_sort
s = 'GGCGGCACCGC$'
sa = simple_kark_sort(s)
print('i\tSA\tf\tSuffixes')
for i in range(len(s)): print(f'{i}\t{sa[i]}\t{s[sa[i]]}\t{s[sa[i]:]}')
```

Produira la sortie suivante :

```
iSAfSuffixes
011$$
16AACCGC$
210CC$
35CCACCGC$
47CCCGC$
58CCGC$
62CCGGCACCGC$
79GGC$
...
```

Implementation BWT

Q 1. Codez (et testez...) la fonction $get\_bwt(s, sa)$ renvoyant la transformée de burrows wheeler bwt du texte s dont le sa a été calculé.

Q 2. Codez la fonction $get\_n(bwt)$ renvoyant un dictionnaire n tel que $n[\alpha]$=nombre d'occurences de caractères plus petits que α dans bwt. C'est aussi l'indice de la ligne où apparait le premier suffixe débutant par α. Par exemple $[G] = 7$.

Q 3. Codez la fonction $get\_r(bwt)$ renvoyant un tableau tel que $r[i]$= rang dans bwt du caractère $bwt[i]$. Exemple : $r[3]$ est égal à 2 car $bwt[3]$ est le deuxième 'G' apparaissant dans bwt.

Q 4. Codez la fonction $left\_first(\alpha, k, n)$ (appelée LF en cours/TD) qui renvoie la ligne l telle que $sa[l]$ est la position du $k^{\text{ième}}$ suffixe (ordonnés dans l'ordre lexico) débutant par le caractère α. Plus simplement, avec f tel que $f[i] = s[sa[i]]$, $lf(\alpha; k)$ est la ligne où le $k^{\text{ième}}$ α apparaît dans f.

Q 5. Codez la fonction $bwt\_2\_seq(bwt, n, r)$ qui retrouve en temps linéaire la séquence s dont la bwt est originaire.

Q 6. Codez la fonction $contains(p, bwt, n, r)$ qui renvoie *true* si p apparait dans la séquence s dont la bwt est originaire, *false* sinon.

Q 7. Codez la fonction $nb\_occurrences(p, bwt, n, r)$ qui renvoie le nombre d'occurrences de p dans la séquence s dont la bwt est originaire, *false* sinon.

Validations sur des exemples *jouets* (e.g. $s = GGCGGCACCGC\$$)

Q 8. Vérifiez le plus de conditions possibles (recherche du premier ou dernier mot du texte indexé, recherche du mot vide, recherche d'un mot plus long que le texte, ...)

Application sur une portion du génome d'E. coli.

Télécharger le fichier `bioinformatique.rennes.inria.fr/data/ecoli_sample.fa` qui continent les premiers 150 kbp d'E. coli. Construire la 'BWT' sur la sequence contenue dans ce fichier.

Q 9. Recherchez par exemple le pattern 'CGCTCTGTGTGACAAGCCGGAAACCGCCCAG' (une sous séquence des premiers 150 kbp d'E. ecoli) via la BWT.

Q 10. Appliquez l'algo naif et l'algo de Karp Rabin sur ces mêmes données et comparez les résultats.

Pire et meilleur des cas

Q 11. Rappelez quel est le pire des cas des méthodes évoquées précédemment. Proposez des tests et des résutlats.

Q 12. Rappelez quel est le meilleur des cas des méthodes évoquées précédemment. Proposez des tests et des résutlats.

Bonus

Q 13. **[Bonus]** Codez la fonction $occurs(s, sa, p)$: indiquant si p a des occurrences dans s en utilisation une approche dichotomique (en $O(|p| \cdot \log |s|)$).

Q 14. **[Bonus]** Ajoutez les résultats par recherche dicchotomique à tous les résultats précédemment obtenus.

Q 15. **[Bonus (facile)]** Codez la fonction $get\_occurrences(p, bwt, n, r, sa)$ qui renvoie la liste des occurrences de p dans la séquence s dont la bwt est originaire.

Q 16. **[Bonus (délicat)]** Proposez et implémentez un moyen de sous échantillonner les ranks pour limiter l'impact mémoire, au prix d'une augmentation des temps de query.