TP n° 8 – Recherche de motifs

esimon@esimon.fr thomas.h@ens.fr
thomash.fr/teaching/

Le but de ce TP est de compter le nombre de fois qu'un motif m apparait dans un texte t.

1 Oups !

La première partie des épreuves pratiques de l'ENS est vraiment embêtante, vous allez donc la faire. On définie les suites suivantes [1] :

$$u_0 = 42$$
$$u_{k+1} = 15091 \times u_k \bmod 64007$$
$$v_k = u_{k+1} \bmod 16$$

$$\alpha_k = \begin{cases} A & \text{si } v_k = 0 \\ B & \text{si } v_k = 1 \\ C & \text{si } 2 \leq v_k \leq 3 \\ D & \text{si } 4 \leq v_k \leq 7 \\ E & \text{si } 8 \leq v_k \end{cases}$$

Et enfin P_n la séquence de n lettres $(\alpha_k)_{0 \leq k < n}$.

Question 1. Écrire une fonction `gen : int -> string` qui renvoie P_i. Vérifiez que `gen 5` est bien `"EDECD"`.

2 Comment le faire mal [2]

Question 2. Écrire une fonction `comptage_naif : string -> string -> int` qui compte le nombre de fois qu'une chaine de caractère apparait dans un texte. Pour cela vous utiliserez deux indices, un pour conserver la position de la fenêtre de recherche et un pour conserver la position dans le motif. Vérifiez que le nombre d'occurrence de la chaine `"DADA"` dans P_{10000} est bien 3.

Question 3. Quelle est la complexité de votre algorithme dans le pire cas ? En moyenne ?

Remarquez que si la dernière lettre de la fenêtre n'apparait nulle part dans m, vous pouvez déplacer la fenêtre de $|m|$. Si la dernière lettre de la fenêtre apparait en position $|m| - 2$, vous pouvez déplacer la fenêtre de 2, etc.

Question 4. Écrire une fonction `table_naive : string -> int vect` qui étant donné un mot retourne un vecteur indiquant quel décalage à effectuer en fonction de la dernière lettre de la fenêtre.

Question 5. Écrire une fonction `comptage_table : string -> string -> int` qui compte en utilisant la table calculée à la question précédente pour déplacer la fenêtre de recherche. Vérifiez que P_5 est présent 16 fois dans P_{5000}.

Question 6. Quelle est la complexité de votre algorithme ? Donnez un exemple sur lequel l'algorithme fait beaucoup de comparaison et un exemple sur lequel il fait moins de comparaison que l'algorithme naïf.

1. Durant l'épreuve vous avez votre u_0 personnel.
2. Cette partie est tirée d'une épreuve pratique ENS de 2010.

3 Comment le faire bien

Un *bord* d'un mot u non vide est un mot différent de u qui est à la fois préfixe et suffixe de u. Le *bord maximal* de u est l'unique bord de longueur maximale, on le note $\beta(u)$ [3].

Question 7. Montrer que la suite $(\beta^k(u))_{k\geqslant 1}$ converge et que l'ensemble de ses termes est exactement l'ensemble des bords de u.

Soit $u = x_0x_1 \dots x_{n-1}$ un mot de longueur $n > 0$. Pour $i = 1, \dots, n$, on note $\ell_i = |\beta(x_0 \dots x_{i-1})|$ la longueur du bord maximal du préfixe de longueur i de u.

Question 8. En trouvant une relation de récurrence sur les ℓ_i, écrire une fonction `table_kmp : string -> int vect` telle que `table_kmp` u détermine en temps $O(n)$ le vecteur `[|`ℓ_0 `;`ℓ_1 `;...;`ℓ_n`|]` [4] pour un mot u de longueur n (la valeur ℓ_0 n'étant pas significative). Vérifiez que le dernier élément de `table_kmp` P_{35} est bien 3.

Question 9. Écrire une fonction `comptage_kmp : string -> string -> int` qui compte en utilisant la table construite à la question précédente. Remarquez que vous n'avez pas à recommencer la recherche en début de motif à chaque fois que vous déplacez la fenêtre de recherche. Quelle est la complexité de votre algorithme ?

Question 10. À quoi ressemble la table retournée par `table_kmp (mot ^"#" ^texte)` ? [5]

4 Comment af90cd0ff2

Question 11. Une autre manière de compter le nombre de motif dans un texte est d'utiliser un hachage roulant. Pour cela, on utilise une fonction de hachage h qui à partir de $h(u_0u_1 \dots u_n)$ et de u_{n+1} peut calculer $h(u_1u_2 \dots u_nu_{n+1})$ en temps $O(1)$. Ensuite, il suffit de hacher le motif et de vérifier seulement les positions où le hash de la fenêtre correspond au hash du motif. Écrivez la fonction `comptage_rk : string -> string -> int` qui implémente cet algorithme (dit de « Rabin-Karp ».)

Question 12. Quelle est sa complexité ? En utilisant la fonction $h(u_0 \dots u_n) = \sum_i u_i * 5^i$ avec $u_i \in [0, 4]$, on peut compter en $O(n)$, non ? [6] Pourquoi ?

3. L'ensemble des bords de *ababa* est $\{\varepsilon, a, aba\}$ donc $\beta(ababa) = aba$.
4. Vous savez les enfants, là vous êtes en train de faire de la programmation dynamique.
5. Si seulement j'avais inversé l'ordre des questions.
6. Non.