Examen de rattrapage d'Algorithmique du texte

Master 1^{ere} année

Mercredi 25 Juin 2014

Exercice 1. Recherche de motif

1. *Soient les mots $v = aabaaababbbaaaa$ et $u = aaabb$. Appliquez l'algorithme de Boyer-Moore pour connaître le nombre d'occurrences de u dans v. On détaillera les étapes et on donnera le nombre de comparaisons effectuées.*
2. *Utiliser l'algorithme de Aho-Corasick pour construire l'automate $\Sigma * X$, avec*
$$X = \{\varepsilon, aa, baab, abaaa, aaba\}$$

Exercice 2. Codage et compression

1. *Calculez le codage du Huffman du texte suivant :*

   $$daabdcdabccbdbccccac$$

   On représentera le code sous la forme d'un arbre, puis son encodage en binaire. Pour l'encodage en binaire, on séparera les unités de sens par des points.
2. *Appliquez l'algorithme LZW sur le texte*

   $$abdcbcbbbdcbabdcb$$

3. *Supposons que l'on possède le dictionnaire suivant : $\{(A, 1), (E, 2), (N, 3), (R, 4)\}$. Décodez le texte suivant : $3, 1, 5, 7, 6, 2, 4, 2$. A chaque étape, on précisera ce qui a été ajouté dans le dicitonnaire.*

Exercice 3. Algorithme de Rabin-Karp

L'algorithme de Rabin-Karp est une amélioration de l'algorithme naif de recherche de motif dans un texte, qui utilise une fonction de hachage. On va donc commencer par voir un exemple de fonction de hachage utilisable par

cet algorithme. Soit un mot u de longueur m défini sur un alphabet Σ de taille k. Soit la fonction $ind : \Sigma \to \{0, \ldots, k-1\}$. La fonction ind est une bijection des lettres de l'alphabet vers l'ensemble des entiers de 1 à n. Exemple : $\Sigma = \{a, b, c\}$ et $ind(a) = 0$, $ind(b) = 1$ et $ind(c) = 2$. Soit une fonction de hachage $hash : \Sigma^* \to \mathbb{N}$. $hash(u_1 \cdots u_n) = hash(u_1 \cdots u_{n-1}) * k + ind(v_n)$

1. *Calculer $hash(abbca)$, $hash(bbca)$ et $hash(ccab)$.*
2. *Trouvez une façon de calculer $hash(v_{i+1} \cdots v_{i+m})$ en fonction de $hash(v_i \cdots v_{i+m-1})$.*
3. *On suppose qu'on utilise un langage de programmation où les entiers sont codés sur 64 bits. Quelle est la longueur maximale du mot u que l'on peut hacher.*
4. *Proposer une fonction de hachage qui permette de hacher des séquences d'une longueur arbitrairement grande.*

Algorithm 1: Algorithme de Rabin-Karp

Entrées: Un texte v de longueur n et un mot u de longueur m
Sorties: Le nombre d'occurrences de u dans v

```
1 occ ← 0;
2 hu ← hash(u);
3 hv ← hash(v_1 ··· v_m);
4 pour i ∈ {1, ..., n − m + 1} faire
5     si hv = hu alors
6         si v_i ··· v_{i+m−1} = u alors
7             occ ← occ + 1;
8     hv ← Calculer hash(v_{i+1} ··· v_{i+m});
9 retourner occ;
```

5. *Exécuter l'algorithme pour $v = aabaaababbbaaaa$ et $u = aababb$. Pour chaque étape (valeur de i) vous préciserez la valeur de hv et vous compterez le nombre de comparaisons effectuées à ligne 6.*
6. *Quelle est la complexité, en temps et en espace, dans le pire des cas et dans le meilleur des cas, de l'algorithe de Rabin-Karp. Seul une réponse détaillée sera prise en compte.*
7. *Quelle est la complexité moyenne de l'algorithme de Rabin-Karp ? Vous pouvez utiliser les résultats de complexité vu en cours pour démontrer ce résultat, ou le prouver par vos propres moyens.*