Complexité et tris

Exercice 1 : Calcul des puissances entières Calculer la complexité de l'algorithme *direct* du calcul de la puissance entière x^n. Même question pour l'algorithme de la *méthode chinoise*.

Exercice 2 : Valeur d'un polynôme Quelle est la complexité des deux algorithmes (calcul direct et schéma de Horner) permettant d'évaluer un polynôme de degré n dont on connait les coefficients dans la base canonique $\{1, x, \ldots, x^n\}$.

Attention : Dans le premier cas (calcul direct), la complexité dépend de la méthode utilisée pour calculer les puissances entières ; considérer les deux possiblités, en utilisant les résultats de l'exercice 1.

Exercice 3 : Suite de Fibonacci Calculer la complexité des deux algorithmes récursifs calculant les termes de la suite de Fibonacci.

Exercice 4 : Calcul des C_n^p Pour calculer le nombre C_n^p de combinaisons de p éléments parmi n (p et n étant tous deux positifs ou nuls), on peut utiliser comme relation de récurrence pour $0 < p < n$ soit $C_n^p = p \,.\, C_{n-1}^{p-1} \,/\, n$, soit $C_n^p = C_{n-1}^{p-1} + C_{n-1}^p$, associée à l'égalité $C_0^n = C_n^n = 1$ pour tout n positif (ou nul).

L'inconvénient de la première relation de récurrence est qu'elle utilise des multiplications entières (qui risquent de dépasser la limite de représentation des entiers) et des divisions entières (qui, si elles sont faites trop tôt, tronquent le résultat). L'inconvénient de la seconde relation de récurrence est qu'elle contient deux appels récursifs.

La meilleure solution consiste à utiliser un tableau pour mémoriser les valeurs déjà calculées. On remarquera que ce tableau n'a pas besoin d'être bidimensionnel : il suffit de mémoriser l'ensemble des C_n^p pour $0 < p < n$ afin de calculer les C_{n+1}^p.

Écrire différents algorithmes récursifs et itératifs permettant de calculer C_n^p pour n et p donnés, et comparer les complexités de ces algorithmes.

Exercice 5 : PGCD de deux entiers Soient deux entiers positifs a et b tels que $a < b$. Montrer que l'algorithme calculant le PGCD de a et b en utilisant le reste de la division euclidienne (fonction `mod`) effectue au plus $2\lceil \log_2 a \rceil$ divisions entières (où $\lceil x \rceil$ désigne le premier entier supérieur à x). On pourra utiliser l'inégalité $(b \bmod a) \leq (b-1)/2$, qui est vérifiée lorsque $0 < a < b$.

Exercice 6 : Tri par sélection d'un tableau Trier par ordre croissant les éléments d'un tableau de réels en recherchant le minimum des valeurs encore non triées. Déterminer la complexité d'un tel algorithme.