**Licence Info**
**Algorithmique Avancée**
**TD 1: 2006**
**Corrigé**

1. Utiliser l'algorithme décrit en cours (celui de Karatsuba) pour mutiplier les deux entiers `172 204 370 251` et `284 160 293 042`; on considère ces nombres comme des nombres de 4 chiffres à base 1000; les multiplications de base de deux nombres inférieurs à 1000 se feront à la main ou par calculatrice.

   **réponse** $a_1 * a_2$ est `172 204` $*$ `284 160`;
   $b_1 b_2$ est `370 251` $*$ `293 042`;
   $(a_1 + b_1) * (b_1 + b_2)$ est `542 455` $*$ `577 202`.

   premier sous-problème: `172 204` $*$ `284 160`;
   les trois sous-sous-problèmes sont: `172`$*$`204` (`48848`), `204`$*$`160` (`32640`) et `376`$*$`444` (`166944`).
   on soustrait les deux premiers du troisième: `85456` et on fait l'addition

```
48 848 000 000
        32 640
    85 456 000
--------------
48 933 488 640
```

   deuxième sous-problème: `370 251` $*$ `293 042`;
   `370`$*$`293` = `108410`
   `251`$*$`042` = `10542`
   `621`$*$`335` = `208035`
   soustraction: `208035-108410-10542=89083`

```
108 410 000 000
         10 542
     89 083 000
--------------
108 499 093 542
```

   troisième sous-problème: `542 455` $*$ `577 202`;
   `542`$*$`577` = `312734`
   `455`$*$`202` = `91910`
   `997`$*$`779` = `776663`
   soustraction: `776663- 312734-91910` = `372019`

```
312 734 000 000
         91 910
    372 019 000
--------------
313 106 110 910
```

Retour au problème de départ: $(a_1 + b_1) * (a_2 + b_2) - (a_1b_1 + a_2b_2)$

```
313 106 110 910
157 432 582 182
---------------
155 673 528 728
```

et l'addition finale:

```
48 933 488 640 000 000 000 000
                108 499 093 542
        155 673 528 728 000 000
-------------------------------
48 933 644 313 637 227 093 542
```

2. L'algorithme *minmax* utilise $3n/2 - 2$ comparaisons pour trouver le minimum et maximum de $n$ nombres ($n$ pair).

   (a) Combien sont utilisées quand $n$ est impair?
   L'algorithme décrit en cours ne traite que les cas de $n$ pair; si on le modifie de telle façon que les sous-problèmes de taille 3 sont traités le plus efficacement possible, à savoir avec nombre de comparaisons: 2 minimum, 3 maximum et 8/3 en moyenne, le total devient: $3n/2 - 5/2$ minimum, $3n/2 - 3/2$ maximum, $3n/2 - 11/6$ en moyenne.
   (Les valeurs moyennes dépendent d'une supposition que le tableau est ordonné aléatoirement.)

   (b) Pour l'algorithme plus naif qui compare chaque nombre après le deuxième avec le maximum des précédents et (s'il était inférieur au maximum) avec le minimum, le nombre de comparaisons dépend de l'ordre des données; Quels sont les nombres minimum et maximum de comparaisons?
   minimum: n-1, maximum: 2n-3.

   (c) On veut démontrer que tout algorithme basé sur les comparaisons doit utiliser au moins le même nombre que *minmax*. On va utiliser une méthode de **l'adversaire** : chaque fois que l'algorithme effectue une comparaison, l'adversaire donne une réponse qui est cohérente avec les réponses précédentes mais choisie afin d'obliger l'algorithme à prendre le maximum de temps; ici l'adversaire devrait essayer de garder le plus grand nombre possible d'éléments du tableau qui pourraient être le minimum et/ou le maximum.
   Donner une stratégie pour l'adversaire qui oblige tout algorithme à effectuer au moins le nombre de comparaisons de *minmax*.
   La bonne stratégie de l'adversaire dépend d'une classification des éléments du tableau comme: soit *isolé* aucune comparaison effectuée, soit *candidat maximum* toutes les comparaisons effectuées le donne

supérieur, soit *candidat minimum* toutes les comparaisons effectuées le donne inférieur, soit *autre*. La stratégie quand l'algorithme compare deux éléments `A[i]` et `A[j]` est

- si les réponses déjà données impliquent l'ordre des deux, la réponse cohérente avec cet ordre
- si les deux sont de la même classe, une réponse arbitraire
- si l'un des deux est un candidat maximum, répondre qu'il est supérieur
- si l'un des deux est un candidat minimum, répondre qu'il est inférieur
- sinon une réponse arbitraire

On peut démontrer par récurrence que le nombre de comparaisons pour achever la recherche à partir d'une situation donnée est au moins $3 \times n_i/2 + n_c - 2$ où $n_i$ est le nombre d'éléments isolés et $n_c$ est le nombre de candidats.
Conclure que tout algorithme nécessite ce nombre, au moins dans le pire cas.

(d) Pour étudier le comportement d'un algorithme en moyenne, on peut supposer que tous les $n!$ ordres possibles des $n$ données ont la même probabilité; avec cette supposition, est-il vrai que tout algorithme doit faire autant de comparaisons que *minmax* en moyenne?
Oui. On peut démontrer que la fonction $3 \times n_i/2 + n_c - 2$ est réduite en moyenne par 1 ou moins à chaque comparaison effectuée. Par exemple, si `A[i]` et `A[j]` sont respectivement candidat max et candidat min, une réponse `A[i]>A[j]` ne réduit pas la fonction mais `A[i]<A[j]` la réduit par 2; il faut démontrer que la probabilité du premier cas est au moins 1/2.

3. Sur une page web j'ai trouvé la remarque suivante sur l'algorithme de Strassen (je traduis de l'anglais mais les auteurs ne sont pas des anglophones) *Aujourd'hui ce résultat n'a aucune importance réelle pour les matrices de réels ou nombres complexes parce que nos ordinateurs effectuent une multiplication dans presque le même temps qu'une addition. Donc, il ne fait pas de sens de remplacer 8 multiplications et 4 additions par 7 multiplications et 18 additions.*

   (a) Expliquez pourquoi cette remarque est complètement stupide!
   Pour les matrices de $n \times n$ il faut comparer le temps de multiplier ou additionner deux sous-matrices de taille $n/2$ et pas le temps pour un élément de base.

   (b) Un algorithme raisonnable utiliserait la méthode de Strassen pour les grandes matrices et basculerait sur la méthode simple pour les petites; si on ne compte que le temps des opérations de base (addition multiplication et soustraction des éléments des matrices) et que

toutes ces opérations prennent le même temps, quel est le bon seuil entre les grandes et les petites matrices?
Une multiplication par la méthode simple prend $n^3$ multiplications et $n^3 - n^2$ additions d'éléments de base. Donc la méthode Strassen est préférable à la méthode simple pour 2 matrices $2n \times 2n$ si l'économie d'une de ces multiplications vaut plus que 14 additions, c'est-à-dire si $2n^3 - n^2 > 14n^2$, ce qui est vrai pour $n >= 8$.