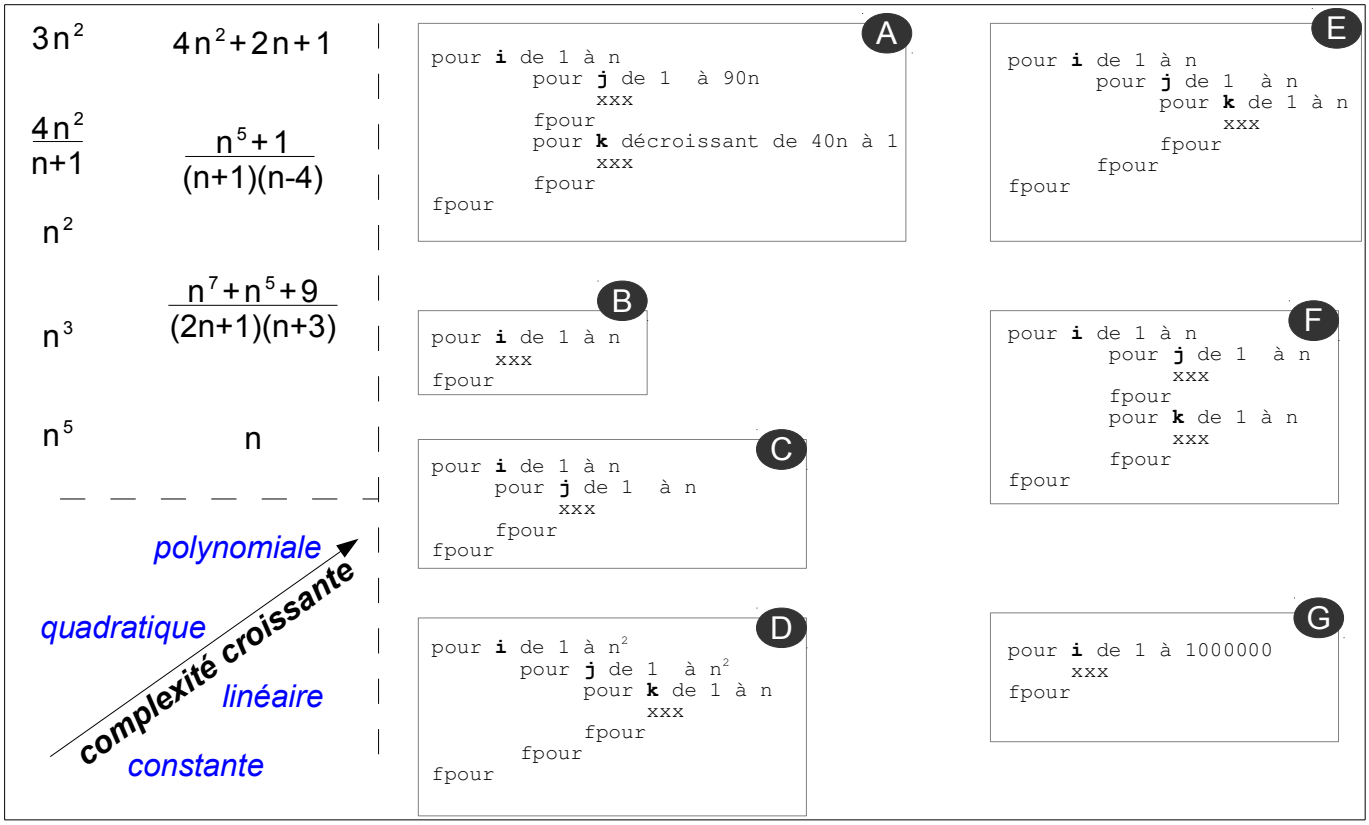
Document de travail pour le TD8
- complexité des algorithmes -

Rappels- calculs et notations

Rappelez ce qu'est :

- un comportement asymptotique
- une classe de complexité
  - logarithmique
  - linéaire ou quasi-linéaire
  - quadratique ou polynomiale
  - exponentielle ou factorielle



Notations : O(1), O(n), O(n²), O(n³), O(n⁴), …, ainsi que O(log(n)), O(n log(n)).

Algorithmes de complexité linéaire	O(n) ou en O(nlog n) algorithmes de tri optimaux
Algorithmes de complexité	O(n²) ou O(n³) multiplication des matrices parcours dans les graphes

Mais aussi :

- exponentielle O(2^n)
- factorielle O(n!)

Question 1- complexité mesurée par le nombre de comparaisons

Travail sur un tableau de *n* éléments : la complexité sera exprimée en fonction de *n*

Exemple 1- **Recherche séquentielle du minimum** dans un tableau quelconque - Nombre de comparaisons et complexité ?

```
pour i de 1 à n
faire comparaison
```

Exemple 2- **Recherche dichotomique** dans un tableau trié - Nombre maximal de comparaisons ? Complexité pour $n=2^k$?

```
si deb <= fin
alors m=indice du milieu
      si t[m] > val
      alors appel récursif sur deb..m
      sinon appel récursif sur m+1..fin
```

```
tant que deb <= fin
faire m=indice du milieu
      si t[m] > val
      alors fin=m-1
      sinon deb=m+1
```

Exemple 3- **Recherche du minimum et du maximum dans un tableau** -
À partir de 3 éléments, les minimum et maximum sont récursivement calculés sur les moitiés de tableaux. Puis le plus petit des deux minimums et le plus grand des deux maximums sont choisis.
Nombre de comparaisons ? Complexité en supposant que $n=2^k$?

Exemple 4- **Tri par sélection** -
À chaque étape : recherche du plus petit élément, que l'on place au début
Nombre de comparaisons et complexité ?
Nombre de déplacements et complexité associée ?

```
pour i de 1 à n-1
faire  mini=t[i]
        pour j de i+1 à n
       faire comparer
       fpour
      déplacer
fpour
```

Question 2- autres mesures de complexité

Exemple 1- **Tri par dénombrement** pour des tableaux ayant peu d'éléments différents -
on suppose que les valeurs sont comprises entre 1 et nbVal
méthode : compter le nombre d'occurence de chaque valeur puis recomposer le tableau
Nombre de passage dans les boucles ?
Complexité correspondante ?

```
pour i de 1 à n
faire incrémenter compte[t[i]]
j=1
pour val de 1 à nbVal
faire pour cpt de 1 à compte[val]
      faire t[j]=val
           val++
           j++
```

Exemple 2- **Tour de Hanoï** -
Nombre de déplacements ?
Complexité correspondante ?

Exemple 3- **Multiplication (égyptienne ou russe?)** de 2 entiers *a* et *b*-
méthode : à chaque étape, multiplier *a* par 2 et diviser (division entière) *b* par 2. Ajouter au produit la valeur de *a* lorsque *b* est impair. Arrêt lorsque *b* est à 1.
On considère qu'à chaque étape il y a une addition sur *a* (*a+a* pour effectuer *a*2*). Si *b* est impair, une seconde addition est effectuée pour mettre à jour le produit.

a (multiplier par 2)	b (diviser par 2)	produit
		0
125	91	125
250	45	375
500	22	-
1000	11	1375
2000	5	3375
4000	2	-
8000	1 (arrêt)	11375 (résultat)

Que se passe-t-il lorsque b est de la forme 2^p ?
Nombre d'additions et complexité, quand b est de la forme 2^p ?

Exemple 4- **Multiplication de matrices et vecteurs** -
Nombre de multiplications nécessaires et complexité pour deux matrices $n*n$?
Nombre de multiplications nécessaires et complexité pour un vecteur n et une matrice $n*n$?
Idem en comptant aussi les additions.

Exemple 5- **Parcours de graphes** -
sur un graphe de n nœuds et m arcs, que pouvez-vous dire sur la complexité des algorithmes cités (comme un O() dépendant de n), à partir des informations ci-dessous :

- une implémentation de l'algorithme de Dijkstra utilisant n^2+m opérations
- une variation de l'algorithme de Floyd construite avec n itérations contenant n^2 itérations
- l'algorithme de Dantzig vu en TD : une itération pour prendre en compte le $k+1^e$ sommet

```
pour k de 2 à n
faire pour s de 1 à k
     faire pour c de 1 à k
          faire ...
```

Question 3- multiplication de Karatsuba

Pour multiplier deux entiers longs (de n chiffres chacun), la méthode classique effectue $n*n$ multiplications individuelles et n additions[1].
Karatsuba a proposé une méthode[2] consistant à diviser en deux le nombre de chiffres à chaque étape. On considère deux entiers a et b, de n chiffres (on suppose que $n=2m$) que l'on peut réécrire sous la forme : $a=fa*10^m+la$ et $b=fb*10^m+lb$.

1) le produit $a*b$ peut s'écrire :
$$(fa*10^m+la)*(fb*10^m+lb) \quad \text{soit} \quad fa*fb*10^{2m}+(fa*lb+la*fb)*10^{2m}+la*lb$$

2) mais en utilisant le fait que $(fa+la)*(fb+lb) = fa*fb+la*lb+fa*lb+la*fb$
on peut écrire le produit $a*b$ sous la forme :
$$\mathbf{fa*fb}*10^{2m}+(\ \mathbf{(fa+la)*(fb+lb)} - \mathbf{fa*fb} - \mathbf{la*lb}\ )*10^m+\mathbf{la*lb}$$

1 *Sans tenir compte des additions et soustractions liées aux retenues.*
2 *Référence : Karatsuba, A. and Ofman, Yu. "Multiplication of Many-Digital Numbers by Automatic Computers." Doklady Akad. Nauk SSSR **145**, 293-294, 1962. Translation in Physics-Doklady 7, 595-596, 1963. et un testeur http://utilitymill.com/utility/Karatsuba_Multiplication*

Quelles seront les multiplications faites pour : 1234*6789 ?
Pour les deux réécritures ci-dessus : combien de multiplications sur *m* chiffres fait-on ?
Qu'en est-il des complexités de ces multiplications ?
Quand utilise-t-on des multiplications de grands entiers (jusqu'à une centaine de chiffres) ?

Annexes

Les calculs de base

n! = n (n-1) (n-2) … 1

$$\frac{n(n+1)}{2} = 1+2+3+\ldots+n$$

$$\frac{n(n+1)(2n+1)}{6} = 1^2+2^2+\ldots+n^2$$ est un $O(n^3)$, de façon générale $\sum_{i=1}^{i=n} i^k$ est un $O(n^{k+1})$

Les logarithmes

log(a*b) = log(a)+log(b)
log(a^n)=n*log(a)
pour $n=2^k$, k=log(n)/log(2) aussi noté $\log_2 n$
pour $n=3^k$, k=log(n)/log(3) aussi noté $\log_3 n$

Notations et unités

Source http://imss-www.upmf-grenoble.fr/prevert/Prog/Complexite/definitions.html

La notation *O(f(n))* décrit le comportement asymptotique d'une fonction *g(n)* par une borne supérieure :
pour de grandes valeurs de *n*, il existe une constante *s* telle que ***g(n) ≤ s * f (n)***

La notation Ω (*f(n)*)[3] décrit le comportement asymptotique d'une fonction *g(n)* par une borne inférieure :
pour de grandes valeurs de *n*, il existe une constante *i* telle que ***g(n) ≥ i * f (n)***

La notation Θ (*f(n)*)[4] permet d'encadrer le comportement asymptotique d'une fonction *g(n)* :
pour de grandes valeurs de *n*, il existe deux constantes *i* et *s* telles que ***i * f(n) ≤ g(n) ≤ s * f(n)***

Les complexités en chiffres

D 'après « Introduction à la complexité des algorithmes », François Morain, page 22
Le temps d'exécution est indiqué pour n=1000000.

flops \ complexité	ln(n)	n	n²	2^n
10^6	0,013ms	1s	278 heures	10000 ans
10^9	0,013µs	1ms	¼ heure	10 ans
10^{12}	0,013ns	1µs	1s	1 semaine

flops : indique le nombre d'opérations en virgule flottante réalisables par seconde (**FL**oating point **O**perations **P**er **S**econd).
ms ou milliseconde (10^{-3} seconde)
µs ou microseconde (10^{-6} seconde)
ns ou nanoseconde (10^{-9} seconde)

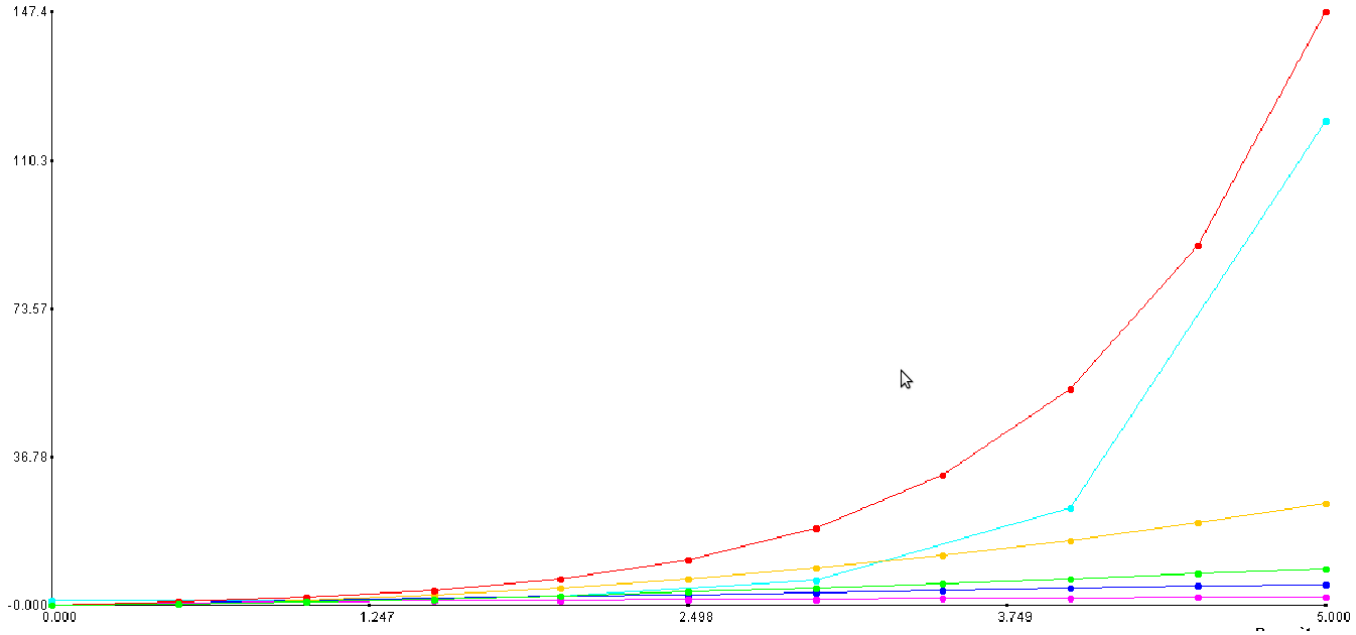
D'après « Complexité et algorithmique », B. Estellon, page 49.
Le temps d'exécution est indiqué pour n=100 avec 10^8 flops :

O(1)	constante	constant
O(log(n))	logrithmique	10^{-7}s
O(n)	linéaire	10^{-6}s
O(n*log(n))	quasi-linéaire	10^{-5}s
$O(n^2)$	quadratique	10^{-4}s
$O(n^3)$	cubique	10^{-2}s
$O(n^p)$	polynomiale	11 jours, pour p=7
$O(2^n)$	exponentielle	10^{14} années
O(n !)	factorielle	10^{142} années

3 *Ω (grand oméga)*
4 *Θ (grand thêta)*

Les complexités en courbe

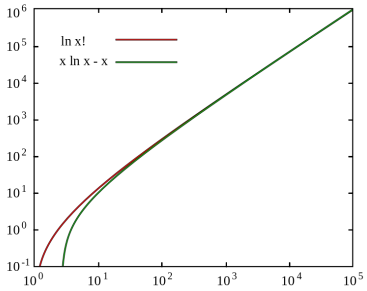
D 'après « Introduction à la complexité des algorithmes », François Morain, page 22
Les courbes correspondent dans l'ordre croissant : log(n) en rose, n en bleu sombre, nlog(n) en vert, n^2 en jaune, n ! en bleu clair, 2^n en rouge.



D'après wikipaedia, page sur l'approximation de Stirling

$$\ln(n!) = n * \ln(n) - n + O(\ln(n))$$

$$n! \simeq \left(\frac{n}{e}\right)^n \sqrt{2\Pi n}$$



Sources

Parmi les ressources disponibles sur le web :

- *Complexité des algorithmes*, Stéphane Grandcolas (université d'Aix-Marseille)
- *Introduction à la complexité des algorithmes*, François Morain (université de Picardie)
- *Complexité d'un algorithme (chapitre 6)*, Karine Deschinkel, *www.e-campus.uvsq.fr*
- *Analyse de la complexité des algorithmes*, Pascal Véron (université de Toulon)
- *Complexité des algorithmes (2)*, Nour-Eddine Oussous, Eric Wegrzynowski (université de Lille-1)
- *Complexité et algorithmique*, B. Estellon (université d'Aix-Marseille)

Analyse de la vitesse de calcul des super-calculateurs : http://www.top500.org/
Evaluations de la multiplication de Karatsuba : http://www.cs.cmu.edu/~cburch/251/karat/
« *Algorithmique (3e édition)* ». Cormen, Leiserson, Rivest, Stein. Ed. Dunod, 2010. pages 19-25, 39-58.