Notion d'algorithme et de complexité

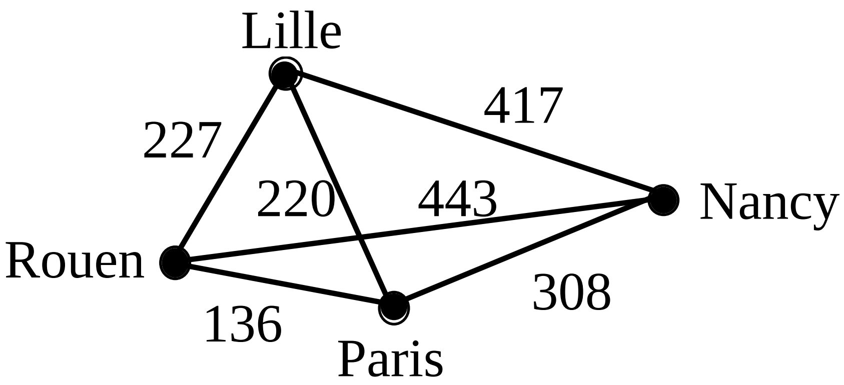
algorithme

- ensemble de règles opératoires dont l'application permet de résoudre un problème en un nombre fini d'opérations
- choix des règles opératoires
- étude préliminaire de l'algorithme
  - préciser les actions sans les détails des opérations
  - vérifier la correction de l'algorithme
  - étudier son efficacité en temps et en espace mémoire

⇨ ***complexité de l'algorithme***

- implantation de l'algorithme, i.e. mise en œuvre

voyageur de commerce



- parcours $(V_1,...,V_n) = \Sigma\, d(V_{i-1},V_i)$

calculer le parcours de toutes les permutations prendre et choisir

- additions: n * n!
  - 5 villes => 600 additions
  - 20 villes => 5 10^{19} additions, soit 1,5 millions d'années
- note: résolution par des heuristiques

opération fondamentale

- peut être complexe, mais de durée indépendante des données
- si plusieurs opérations fondamentales, décompte séparé, et coefficient
- opération de détail prises en compte par absorption
- comparaison entre algorithmes si mêmes opérations
- exemples:
  - addition des distances dans voyageur de commerce
  - comparaison de valeurs dans une recherche ou un tri
  - déplacements de valeurs dans un tri
  - accès à la mémoire secondaire

calcul de la complexité

- évaluer le nombre d'exécutions de l'opération fondamentale

var T: **tableau** [1..N] **de** T_Elem; {rangé par valeurs croissantes}
k : entier;
début k := 1; **tant que** k <= N **et alors** $\boxed{\text{T[k] < x}}$ **faire** k := k+1; **fait**;
retourner k;
fin

- invariant: pour tout j < k, t[j] < x
- sortie de boucle: k = N+1 ou (k≤N et T[k]≥x)
- complexité: au mieux 1, au pire N, en moyenne N/2

{1, ou 2, ou 3,..., N} si k≤N et N si k=N+1

définitions

- Soit D_n l'ensemble des données de taille n
  et $coût_A$ (d) le coût de l'algorithme sur la donnée d

- *complexité au mieux* $Min_A(n)=min\{coût_A(d) \mid d \quad D_n \}$

- *complexité au pire* $Max_A(n)=max\{coût_A(d) \mid d \quad D_n \}$

- *complexité en moyenne* $Moy_A(n)=\Sigma\{p(d)*coût_A(d) \mid d \quad D_n\}$
  où p(d) est la probabilité d'avoir le donnée d

$$\Rightarrow \quad Min_A(n) \leq Moy_A(n) \leq Max_A(n)$$

comparaisons d'algorithmes

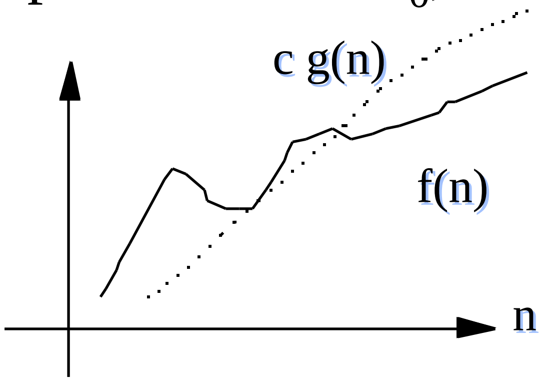

```
var T: tableau [1..N] de T_Elem;
      i, j, k: entier;
début
   i := 1; j := N;
   tant que i < j faire
       k := (i+j) / 2;
       si  T[k] < x   alors i := k+1;
                      sinon j := k; finsi;
   fait;
   si i = N et alors T[i] < x alors
        i := N+1;
   finsi;
   retourner i;
fin;
```

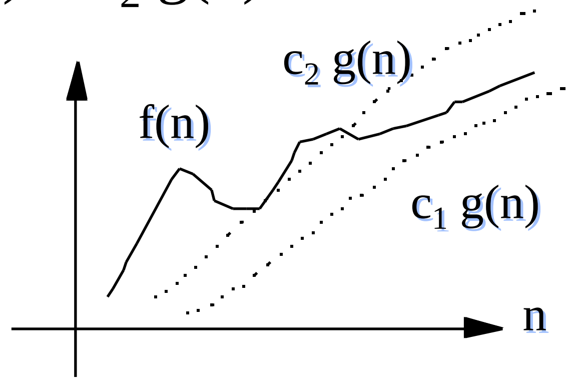
- invariant:
  - $m < i \Rightarrow T[m] < x$
  - $j \neq N \Rightarrow x \leq T[j]$
  - $N \neq 0 \Rightarrow i \leq j$
- arrêt:
  - nouveau(j-i) = ancien(j-i) / 2
- complexité:
  - $\lceil \log_2 N \rceil$

→ meilleur que précédent sauf si début

ordre de grandeur

- l'important est l'ordre de grandeur, et l'évolution en fonction de la taille
- définition: f est *dominée* par g, f = O(g) s'il existe n_0 et c tels que pour tout n > n_0, on ait $f(n) \leq c\, g(n)$
- définition: f et g sont *du même ordre de grandeur*, f = Θ(g) s'il existe n_0 , c_1 et c_2 tels que
  pour tout n > n_0, on ait $c_1\, g(n) \leq f(n) \leq c_2\, g(n)$





temps d'exécution suivant la taille

	1	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
10^2	1µs	7µs	0.1 ms	0.7 ms	10 ms	1 s	$4\,10^7$Ga
10^3	1µs	10µs	1 ms	10 ms	1 s	17 mn	∞
10^4	1µs	13µs	10 ms	130 ms	1.7 mn	12 j	∞
10^5	1µs	17µs	0.1 s	1.7 s	2.8 h	32 a	∞
10^6	1µs	20µs	1 s	20 s	12 j	32 Ka	∞

taille suivant temps d'exécution

	1	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
1 s	∞	∞	1 M.	63 K.	1 K.	100	20
1 mn	∞	∞	60 M.	3 M.	7 K.	400	26
1 h	∞	∞	4 G.	130 M.	60 K.	1.5 K.	32
1 j	∞	∞	90 G.	3 G.	300 K.	4.4 K.	36

complexité en place mémoire

- évaluer la quantité de mémoire nécessaire en fonction de la taille des données

conservation des résultats intermédiaires → diminution de complexité en temps

- mémoire doit être suffisante

compromis espace-temps

conclusion

- algorithme: correct et temps d'exécution raisonnable
- complexité relative à une ou des opérations fondamentales
  - dépend de la taille des données et de leur configuration
    - complexité < n^2 => taille quelconque
    - entre n^2 et n^3 => taille moyenne
    - > n^3 => petite taille
- performance des machines ne change pas l'efficacité
- compromis espace-temps