

Objectifs

À la fin de cette séance, vous devriez être capable de :

- évaluer le coût d’un algorithme simple dans le pire et dans le meilleur des cas ;
- comparer des algorithmes selon leur complexité ;
- évaluer la qualité d’un algorithme selon sa complexité.

Exercice 1 : Itérations emboîtées (30 min)

Compter le nombre d’opérations *Schtroumpfer* exécutées par chacun des algorithmes suivants.

- (1) pour $i = 1$ à n faire
 - (2) pour $j = 1$ à n faire
 - (3) *Schtroumpfer*()
- (1) pour $i = 1$ à n faire
 - (2) pour $j = 1$ à i faire
 - (3) *Schtroumpfer*()
- (1) pour i de 5 à $n-5$ faire
 - (2) pour j de $i-5$ à $i+5$ faire
 - (3) *Schtroumpfer*()
- (1) pour $i = 1$ à n faire
 - (2) pour $j = 1$ à i faire
 - (3) pour $k = 1$ à j faire
 - (4) *Schtroumpfer*()
5. Plus généralement, que pouvez-vous dire de la complexité d’un algorithme en observant le nombre de boucles emboîtées ?

Exercice 2 : Recherche séquentielle (30 min)

On étudie un algorithme de recherche séquentielle dans une table. On se place dans le cas où il n’y a pas d’hypothèse sur le fait que la table est ordonnée ni sur la présence de l’élément cherché dans la table.

1. Spécifier et écrire proprement un tel algorithme.
2. Il s’agit maintenant d’évaluer le nombre de comparaisons effectuées lors de la recherche.
Déterminer les cas favorables et défavorables et les nombres de comparaisons correspondants.
Démontrer qu’ils correspondent bien respectivement au minimum et au maximum du coût de votre algorithme.

Exercice 3 : Valeurs numériques et ordres de grandeurs (30 min)

On suppose qu’on travaille sur une machine capable d’effectuer environ un milliard d’opérations par seconde.

Calculer (**sans calculatrice**) le temps nécessaire approximatif pour exécuter des programmes dont les coûts sont donnés ci-dessous, avec des données de différentes tailles en entrée :

↓ Coût de l’algorithme \ Taille des données →	1 000	1 000 000	1 000 000 000
n			
$n \log_2 n$			
$n + 1\,000\,000$			
$\frac{n^2}{1\,000} + 1\,000n$			

- Lesquels de ces algorithmes sont utilisables :
 - à chaque chargement d’une page web ?
 - à chaque démarrage d’une machine ?
 - pour produire les plans d’une usine ?
 Quelle(s) conclusion(s) plus générale(s) en tirez-vous sur les ordres de grandeurs respectifs de ces coûts ?
- Tracer sur une même figure les allures des courbes des fonctions $\log_2(n)$ et \sqrt{n} sur une échelle assez grande ($n = 10\,000$ par exemple).
Même consigne pour $10^{10} \times n^3$ et 2^n , avec n compris entre 0 et 50.

Exercice 4 :

On considère l’algorithme suivant :

- (1) $a := T[0]$
- (2) pour $i = 1$ à $n-1$ faire
- (3) si $T[i] > a$ faire
- (4) traiter $T[i]$
- (5) $a := T[i]$

- Que calcule-t-il ? Expliciter les données, le résultat.
- Expliciter le modèle de coût. Quel est son coût minimal, son coût maximal ?
- Peut-on espérer écrire un algorithme qui calcule le maximum d’un tableau avec strictement moins de comparaisons que celui-ci ?

Membres du groupe

—
—
—
—

Exercice 2 : Recherche séquentielle

On étudie un algorithme de recherche séquentielle dans une table. On se place dans le cas où il n’y a pas d’hypothèse sur le fait que la table est ordonnée ni sur la présence de l’élément cherché dans la table.

1. Spécifier et écrire proprement un tel algorithme.
2. Il s’agit maintenant d’évaluer le nombre de comparaisons effectuées lors de la recherche. Déterminer les cas favorables et défavorables et les nombres de comparaisons correspondants. Démontrer qu’ils correspondent bien respectivement au minimum et au maximum du coût de votre algorithme.