

Algorithmique — Complexité et algorithmes fondamentaux

NSI · Première
Contrôle n°1 · 60 min · sur
20

Exercice 1 — La complexité

5 pts 14 min

Définir et calculer une complexité

Réponds à chaque question.

LA RÈGLE DU COURS

Une notation décrit une croissance, pas une performance. Un chronomètre sur une petite entrée ne dit rien.

1. Que mesure la complexité d'un algorithme ? (1 pt)

La croissance du coût quand la taille de l'entrée augmente : c'est une loi d'évolution, pas une durée en secondes.

→ La machine employée ne change pas la loi.

2. Qu'ignore-t-on dans la notation, et pourquoi ? (1 pt)

Les constantes multiplicatives et les termes de moindre ordre, parce qu'ils cessent de compter quand l'entrée grandit.

→ Sur une grande entrée, seul le terme dominant décide du coût : $O(3n + 5)$ s'écrit $O(n)$.

3. Range les classes suivantes par coût croissant : $O(n \log n)$, $O(1)$, $O(n^2)$, $O(n)$, $O(\log n)$. (1 pt)

$O(1)$, $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$.

→ C'est l'ordre des classes usuelles, de la moins coûteuse à la plus coûteuse.

4. Pourquoi un chronomètre sur un petit jeu de données trompe-t-il ? (1 pt)

Parce que sur une petite entrée les constantes dominent : un algorithme quadratique bien optimisé peut battre un algorithme en $O(n \log n)$ mal implémenté.

→ L'ordre s'inverse nécessairement à partir d'une certaine taille.

5. Combien d'opérations un algorithme quadratique effectue-t-il environ pour $n = 1\,000$? (1 pt)

Environ un million.

→ L'ordre de grandeur est n multiplié par n , soit $1\,000 \times 1\,000$.

Exercice 2 — La dichotomie

4 pts 11 min

Analyser la recherche dichotomique

Réponds à chaque question.

LA RÈGLE DU COURS

Une recherche élimine la moitié de l'espace à chaque étape, à une condition dont l'oubli ne se voit pas.

1. Comment procède une recherche dichotomique ? (1 pt)

Elle compare à l'élément central : si la valeur cherchée est plus petite, elle élimine la moitié droite, sinon la moitié gauche.

→ Chaque comparaison divise par deux l'espace restant.

2. Quelle est sa complexité, et pourquoi ? (1 pt)

$O(\log n)$, parce que chaque comparaison divise par deux l'espace de recherche.

→ Les découpages par deux introduisent un logarithme.

3. Quelle condition impérative exige-t-elle ? (1 pt)

Le tableau doit être trié.

→ L'algorithme élimine une moitié sur la seule foi de la comparaison centrale.

4. Que se passe-t-il si le tableau n'est pas trié ? (1 pt)

L'algorithme ne plante pas : il répond « absent » pour un élément présent.

→ C'est une faute silencieuse, qui ne se manifeste que sur certaines entrées.

Exercice 3 — Les tris

4 pts 11 min

Comparer deux tris

Réponds à chaque question.

LA RÈGLE DU COURS

Deux tris n'ont ni la même complexité ni le même coût en mémoire. Aucun des deux ne l'emporte partout.

1. Comment procède le tri par insertion, et quelle est sa complexité dans le cas défavorable ? (1 pt)

Il construit la partie triée élément par élément, en insérant chaque nouvel élément à sa place ; sa complexité est en $O(n^2)$ dans le cas défavorable.

→ Il reste efficace sur de petits tableaux et sur des données presque triées.

2. Comment procède le tri fusion ? (1 pt)

Il coupe le tableau en deux, trie chaque moitié, puis fusionne les deux moitiés triées.

→ C'est l'application du principe « diviser pour régner ».

3. Quelle est sa complexité, et dans quels cas ? (1 pt)

$O(n \log n)$, dans tous les cas.

→ Sa performance ne dépend donc pas de l'état initial du tableau.

4. Quel coût le tri fusion a-t-il que le tri par insertion n'a pas ? (1 pt)

Il utilise de la mémoire supplémentaire pour la fusion.

→ Choisir un tri, c'est arbitrer entre temps et mémoire.

Exercice 4 — Dijkstra et le comptage

4 pts 13 min

Appliquer Dijkstra et compter les boucles

Réponds à chaque question.

LA RÈGLE DU COURS

Un algorithme de plus court chemin a une condition d'emploi, et le comptage des boucles suit des règles simples.

1. Que calcule l'algorithme de Dijkstra ? (1 pt)

Le plus court chemin depuis un sommet de départ dans un graphe dont les arêtes portent des poids positifs.

→ Il part d'un sommet unique et dessert tous les autres.

2. Quel est son principe ? (1 pt)

Maintenir pour chaque sommet la meilleure distance connue, traiter à chaque étape le sommet non traité le plus proche, et mettre à jour ses voisins.

→ C'est le traitement par ordre de proximité qui garantit le résultat.

3. Pourquoi la condition de positivité des poids n'est-elle pas un détail ? (1 pt)

Parce qu'avec des poids négatifs, l'algorithme peut figer une distance qu'un chemin ultérieur aurait améliorée.

→ Le sommet traité n'est alors plus définitivement résolu.

4. Quelle complexité donnent deux boucles imbriquées portant chacune sur n ? (1 pt)

$O(n^2)$.

→ C'est l'imbrication qui compte, non le nombre de boucles écrites.

Repérer une erreur et la corriger

Chaque affirmation est fausse. Écris ce qu'il fallait dire.

LA RÈGLE DU COURS

Trois fautes reviennent. Une mesure prise pour une loi, une condition oubliée, et une notation encombrée.

1. « Mon algorithme quadratique est plus rapide sur cent éléments : il est donc meilleur. » (1 pt)

Sur une petite entrée les constantes dominent ; l'ordre s'inverse nécessairement à partir d'une certaine taille.

→ Il faut comparer des lois de croissance, pas des mesures ponctuelles.

2. « La dichotomie plante si le tableau n'est pas trié. » (1 pt)

Elle ne plante pas : elle répond « absent » pour un élément présent.

→ C'est une faute silencieuse, plus dangereuse qu'une erreur visible.

3. « Cet algorithme effectue $3n + 5$ opérations : sa complexité est $O(3n + 5)$. » (1 pt)**Sa complexité s'écrit $O(n)$.**

→ On ignore les constantes multiplicatives et les termes de moindre ordre.

Barème — ce que le correcteur regarde

Les lignes chiffrées couvrent les 5 exercices et totalisent 20 points.

EXERCICE	ATTENDU	POINTS	POINT PERDU SI
Ex.1	Ce que mesure la complexité	5 points — 1 par question	Une complexité exprimée en secondes.
Ex.2	La recherche dichotomique	4 points — 1 par question	La condition de tri oubliée.
Ex.3	Les tris	4 points — 1 par question	Un tri désigné vainqueur absolu.
Ex.4	Dijkstra et le calcul de complexité	4 points — 1 par question	Des poids négatifs admis sans réserve.
Ex.5	L'erreur repérée, et la réponse juste donnée	3 points — 1 par ligne	Une correction qui garde les constantes dans la notation.

ET MAINTENANT ?

16 à 20

Le chapitre est acquis. Entraîne-toi maintenant à évaluer une complexité en cinq temps : la taille de l'entrée, l'imbrication des boucles, les découpages par deux, les constantes ignorées, et le cas précisé.

10 à 15

Refais l'exercice 1 en raisonnant sur de grandes tailles : c'est là que les lois de croissance se séparent.

moins de 10

Reprends la fiche de cours et retiens ce que la complexité mesure : la croissance du coût quand l'entrée augmente, pas une durée en secondes.