CORRIGÉ

1re

Corrigé détaillé
4 exercices

Algorithmique — Complexité et algorithmes fondamentaux

> Ce qui est noté au contrôle, ce n'est pas le résultat seul : c'est le raisonnement posé, puis le calcul mené jusqu'au bout avec l'unité.

1 Vocabulaire. Répondre en une phrase précise.

1. a. La **croissance du coût** quand la taille de l'entrée augmente. **Non** : c'est une **loi d'évolution**, pas une durée en secondes.
2. b. **O(1)**, O(log n), **O(n)**, O(n log n), O(n²).
3. c. Que le tableau soit **trié**.
4. d. Le **tri fusion** est en **O(n log n)** dans tous les cas ; le **tri par insertion** en **O(n²)** dans le cas défavorable.
5. e. Le **plus court chemin** depuis un sommet de départ, à condition que les arêtes portent des **poids positifs**.

2 Comparer les ordres de grandeur. On raisonne en nombre d'opérations.

1. a. Ordre de grandeur :
$$1000 \times 1000 = 1000000$$
2. Soit environ **un million** d'opérations.
3. b. Environ **dix**, puisque dix divisions par deux couvrent plus de mille éléments :
$$2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 1024$$
4. c. Rapport :
$$1000000 \div 10 = 100000$$
5. Soit **cent mille** fois plus d'opérations.
6. d. Le coût est multiplié par **quatre** : doubler n multiplie n² par 2 × 2.
$$2 \times 2 = 4$$
7. e. Il augmente d'**une seule** étape : passer de 1 000 à 2 000 demande une division par deux supplémentaire. C'est toute la différence entre une loi quadratique et une loi logarithmique.

3 Un développeur applique une recherche dichotomique à un tableau non trié. Analyser.

1. a. **Non**. L'algorithme compare, élimine une moitié et termine normalement : rien dans son déroulement ne signale l'anomalie.
2. b. Il peut renvoyer « **absent** » alors que l'élément **est présent** : il l'a éliminé avec une moitié sur la foi d'une comparaison qui ne voulait rien dire.
3. c. Parce qu'elle est **silencieuse** : un plantage se voit et se corrige, un résultat faux se propage. Le programme continue, et l'erreur n'apparaît que sur certaines entrées — donc parfois jamais pendant les tests.
4. d. Trier coûte **O(n log n)**, chercher une fois en **O(n)** sans tri. Pour une **seule** recherche, le tri n'est pas rentable. Il le devient dès qu'on effectue **de nombreuses** recherches sur le même tableau : le coût du tri est payé une fois, chaque recherche coûte ensuite O(log n).

4 Choisir un algorithme. Raisonner.

1. a. Parce que sur une petite entrée, les **constantes** dominent : un algorithme quadratique bien optimisé peut battre un O(n log n) mal implémenté. L'ordre s'inverse **nécessairement** au-delà d'une certaine taille.
2. b. Sur de **petits tableaux** et sur des données **presque triées**, où il est très efficace — c'est d'ailleurs pourquoi certains tris performants l'emploient sur les petits fragments.
3. c. De la **mémoire supplémentaire** pour la fusion : il n'opère pas sur place.
4. d. Parce qu'il **fige** la distance d'un sommet dès qu'il le traite, en supposant qu'aucun chemin ultérieur ne pourra faire mieux. Avec un poids **négatif**, un détour peut réduire la distance après coup, et l'hypothèse tombe.
5. e. Que le choix dépend de **contraintes** : taille des données, mémoire disponible, état initial des données, nombre de recherches à effectuer. Un algorithme est adapté **à un contexte**, et annoncer un vainqueur sans préciser lequel revient à répondre à une question qu'on n'a pas posée.

Corrige-toi toi-même. Compte tes points exercice par exercice : c'est ainsi qu'on voit où l'on perd, et ce n'est presque jamais là où on le croit.

Ce qui est évalué	Points	Ce qui fait perdre le point
Définir la complexité comme loi de croissance	4 pts	La confondre avec un temps d'exécution

Déduire une complexité de l'imbrication des boucles	4 pts	Compter le nombre de boucles au lieu de leur imbrication
Exiger un tableau trié pour la dichotomie	5 pts	L'appliquer à un tableau quelconque
Comparer les tris selon leurs contraintes	4 pts	Désigner un tri vainqueur absolu
Énoncer la condition de Dijkstra	3 pts	L'appliquer à des poids négatifs