CORRIGÉ

Tle

Corrigé détaillé
4 exercices

Le paradigme diviser pour régner

> 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. **Diviser**, **régner**, **combiner**.
2. b. Des sous-problèmes de **tailles comparables**, et une **combinaison peu coûteuse**.
3. c. Parce qu'elle ne garde **qu'une** des deux moitiés : il n'y a pas deux résultats à recoller.
4. d. **O(n log n)**, dans **tous** les cas — les moitiés sont égales par construction.
5. e. Du choix du **pivot** : bien choisi, O(n log n) en moyenne ; systématiquement mal choisi, **O(n²)**.

2 Comparer les ordres de grandeur pour n = 1 000 000. On prendra log n ≈ 20.

1. a. Produit :
$$1000000 \times 20 = 20000000$$
2. Soit **20 millions** d'opérations.
3. b. Carré :
$$1000000 \times 1000000 = 1000000000000$$
4. Soit **mille milliards**.
5. c. Rapport :
$$1000000000000 \div 20000000 = 50000$$
6. Soit **cinquante mille** fois plus d'opérations.
7. d. Que l'un des deux termine en une fraction de seconde et l'autre en des heures — sur la **même machine**. Aucun processeur plus rapide ne comble un écart de cet ordre : c'est la **loi de croissance** qui décide, pas le matériel.

3 Un développeur affirme que son algorithme est rapide « parce qu'il applique diviser pour régner ». Discuter.

1. a. **Non**. C'est une **méthode de conception**, pas une garantie de performance.
2. b. Un découpage **déséquilibré**, où un sous-problème contient presque tout. Le **tri rapide** systématiquement mal pivoté dégénère ainsi en **O(n²)**, comme un tri élémentaire.
3. c. La **combinaison** : si recoller coûte autant que résoudre directement, le découpage n'apporte rien.
4. d. Poser la relation de **récurrence** — coût des sous-problèmes plus coût de combinaison — et la résoudre, par exemple avec l'**équation de maître**.
5. e. Par exemple : « Mon algorithme découpe en deux parties de tailles **comparables** et les recombine en temps **linéaire** ; la récurrence donne donc O(n log n). » L'affirmation devient **vérifiable**, alors que la première était une reconnaissance de forme.

4 Tri fusion et tri rapide. Analyser leurs compromis.

1. a. Parce que les deux moitiés sont **égales par construction** : la condition de tailles comparables est satisfaite quelles que soient les données. Aucune entrée ne peut le faire dégénérer.
2. b. De la **mémoire supplémentaire** pour la fusion : il n'opère pas sur place.
3. c. Sa combinaison est **gratuite** : après partition autour du **pivot**, les deux parties sont déjà à leur place relative, il n'y a rien à recoller.
4. d. Parce que l'**équilibre des tailles** en dépend entièrement, et avec lui la complexité — de O(n log n) en moyenne à **O(n²)** au pire. Toute la qualité de l'implémentation tient à cette stratégie.
5. e. **Non**. Le **tri fusion** offre une **garantie** au prix de mémoire ; le **tri rapide** est souvent plus rapide en pratique et sans garantie. Le choix dépend de ce qu'on privilégie : la **prévisibilité** ou la **performance moyenne** — et de la mémoire disponible.

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
Citer les trois étapes et les deux conditions	4 pts	Réciter le principe sans ses conditions

Refuser une complexité annoncée par reconnaissance de forme	5 pts	Conclure du paradigme à l'efficacité
Expliquer la garantie du tri fusion	4 pts	L'attribuer à autre chose qu'aux tailles égales
Relier le pivot à la complexité du tri rapide	4 pts	Annoncer O(n log n) sans réserve
Poser une relation de récurrence	3 pts	Annoncer une complexité de mémoire