Fonction `sort_students`

```
void sort_students(int** students_rank, int** students_array, int
students_number, int grades_number) {
    int i = 0, j = 0;
    for (i = 0; i < grades_number; i++) {
        int* grades = (int*) malloc(students_number * sizeof(int));
        for (j = 0; j < students_number; j++) {
            grades[j] = students_array[j][i];
        }
        bubblesort(grades, students_number);
        for (j = 0; j < students_number; j++) {
            students_rank[j][i] = find_rank_student(students_array[j][i],
grades, students_number);
        }
        free(grades);
    }
}
```

Analyse de la complexité :

1. **Boucle externe** :
    - Cette boucle s'exécute pour chaque grade, soit `grades_number` itérations.
2. **Allocation dynamique (`malloc`)** :
    - Elle est appelée une fois par itération de la boucle externe. Selon l'hypothèse donnée, sa complexité est O(1)O(1).
3. **Boucle interne pour copier les grades** :
    - Cette boucle s'exécute studentsnumberstudents_number fois pour chaque itération de la boucle externe.
    - Complexité : O(studentsnumber)O(students_number).
4. **Appel à `bubblesort`** :
    - La complexité du tri à bulles est O(studentsnumber2)O(students_number^2), car c'est un tri quadratique.
5. **Boucle interne pour appeler `find_rank_student`** :
    - Appelée studentsnumberstudents_number fois.
    - La fonction `find_rank_student` inclut un appel à `bubblesort` (déjà analysé) suivi d'une recherche dans le tableau trié.
    - Recherche : O(studentsnumber)O(students_number).
    - Donc, la complexité de `find_rank_student` est O(studentsnumber2)O(students_number^2) (dû à `bubblesort`).

    Ainsi, pour chaque appel à `find_rank_student`, la complexité totale de cette boucle est studentsnumber · O(studentsnumber2)=O(studentsnumber3)students_number \cdot O(students_number^2) = O(students_number^3).
6. **Libération de mémoire (`free`)** :
    - Selon l'hypothèse donnée, sa complexité est O(1)O(1) et elle est appelée une fois par itération.

Complexité totale :

- Pour chaque itération de la boucle externe :
    - Copie des notes : O(studentsnumber)O(students_number)
    - Tri avec `bubblesort` : O(studentsnumber2)O(students_number^2)
    - Boucle interne avec `find_rank_student` : O(studentsnumber3)O(students_number^3)
    - Total pour une itération : O(studentsnumber3)O(students_number^3).
- La boucle externe s'exécute gradesnumbergrades_number fois, donc la complexité totale est :

  O(gradesnumber · studentsnumber3)O(grades_number \cdot students_number^3)

Résultat final :

Complexité de `sort_students` : O(gradesnumber · studentsnumber3)O(grades_number \cdot students_number^3).