Structures

```
Class Student:
    id: number
    height: number
    conflicts: set
```

Setup

```
read heights_line_number

for each line from line 2 to heights_line_number:
init Student with height and index

for each line in conflict_lines
    add student 2 to student's 1 conflict set
    add student 1 to student's 2 conflict set
```

Function place_students_optimized(students)

```
trier students par taille décroissante  % O(n log n)
initialiser student_line vide           % O(1)

pour chaque student dans students       % O(n)
  initialiser placed à faux             % O(1)
  pour i de 0 à len(student_line)       % O(n)
    si i == len(student_line) ou
       student non en conflit avec student_line[i]   % O(1)
      et si i == 0 ou student_line[i - 1] non en conflit avec student  % O(1)
        insérer student à i dans student_line        % O(n)
        placer à vrai                                % O(1)
        sortir de la boucle                          % O(1)

  si non placé
    student_line = place_student_height(student, student_line) % O(n^2) (en pire cas)

initialiser res à une chaîne vide       % O(1)
pour chaque student dans student_line   % O(n)
  ajouter id de student à res           % O(1)

retourner res                           % O(1)

total O(n^2) en pire cas, en cas mou le nombre de conflits est petit on a O(nlogn)
```

Function `place_student_height(student, student_line)`

```
pour chaque i de 0 à len(student_line)    % O(n)
  si student_line[i].height < student.height
    insérer student à la position i      % O(n)
    retourner nouvelle student_line      % O(1)

ajouter student à la fin de student_line % O(1)
retourner student_line                   % O(1)

total: O(n^2) (pire cas si on rentre dans le if)
```