Tâche3
Entrée : *Info_tâche3 (un algorithme de tri) **Sortie** : Aucune.

Propagation(Mat, visited, départ, arrivé, stock, degré, degré_max, ligne, nœud_suivant, n)
 	
Afficher 0
  Pour i allant de 0 à degré_max
  Pour z allant de 0 à n
  Transition[z] <- 0
  Pour j allant de 0 à n
  Si propagatio[j][i] != 0
  Transition[j] <- propagatio[j][i]
  Tri (Transition)
  Pour r allant de 0 à n
  Si Transition[r] != 0
  Afficher transition[r]

Algo_critère_arrêt
Entrée : *Info_critère_arrêt **Sortie** : n

Si départ[nœud_suivant] = -1
  Si arrivé[0] != -1
  Stock[degré_max] <- degré
  Ligne <- arrivé[0]
  Degré <- 0
  Degré_max <- degré_max + 1
  Nœud_suivant <- 1
  Pour i allant de 0 à n
  Départ[i] <- (-1)
  Propagation(départ, arrivé, stock, degré, degré_max, ligne, nœud_suivant, n)
  Sinon
  n <- (-1)
  Sortir n
Sinon
  Ligne <- départ[nœud_suivant]
  Nœud_suivant <- nœud_suivant + 1
  Propagation(départ, arrivé, stock, degré, degré_max, ligne, nœud_suivant, n)

Algo_Propagation
Entrée : *Info_tâche3 **Sortie** : propagatio contenant dans chacune de ses colonnes les nœuds appartenant au même degré de séparation

Pour j allant de 0 à n
  Si j <n-1
  Si Mat[ligne][j] = 1
  Si visited[j] = 1
  Continuer
  Sinon
  Arrivé[degré] <- j
  Visited[j] <- 1
  Propagatio[degré][degré_max] <- j
  Degré <- degré +1
  Sinon
  Continuer
  Sinon
  Si Mat[ligne][j] = 1
  Si visited[j] = 1
  critère_arrêt (départ, arrivé, stock, degré, degré_max, ligne, nœud_suivant, n)
  Si n = (-1)
  Fin
  Sinon
  Arrivé[degré] <- j
  Visited[j] <- 1
  Propagatio[degré][degré_max] <- j
  Degré <- degré +1
  critère_arrêt (départ, arrivé, stock, degré, degré_max, ligne, nœud_suivant, n),
  Si n = (-1)
  Fin
  Sinon
  critère_arrêt (départ, arrivé, stock, degré, degré_max, ligne, nœud_suivant, n)
  Si n == -1
  Fin

b)

** Info_tâche3 : Matrice d'adjacence Mat, de taille n*n, un vecteur de taille n contenant des booléen initialisées à 0 sauf pour le rang 1 nommé visited, une matrice vide de taille n*n nommé propagatio, *Info_critère_arrêt : deux vecteurs respectivement nommés arrivé et départ de taille n et initialisés à (-1) partout, un entier désignant le degré de séparation en cours nommé degré initialisés à 0, le degré de séparation maximum du graphe nommé degré_max initialisés à 0, un vecteur de taille n nommé stock retenant les degrés de séparation, un entier retenant l'emplacement du prochain nœud dans départ ou arrivé nommé nœud_suivant, l'entier indice de ligne nommé ligne initialisé à 0

c) En admettant que l'algorithme récursif qui détermine les (n-1) degrés de séparation d'un seul nœud de départ à une complexité de O(N^2), on peut estimer l'ordre de complexité de la tâche 4 à O(N^3) car on répète N fois l'algorithme récursif afin d'obtenir le degré de séparation à partir de tous les nœuds du graphe.
d) Avec un temps d'exécution « user » des fichiers ; t07, t08, t09 et t10, respectivement ; 0.822, 5.866, 43.286 et 358.535, secondes, on observe un type de croissance presque linéaire, puisque 5.866/0.822 environ égale à 43.286/5.866 environ égale à 358.535/43.286 environ égale à 4. Ce n'est pas cohérent avec la question précédente, mais cela peut peut-être s'expliquer par le fait que l'algorithme récursif n'est pas d'ordre O(N^2) dans mon programme.

a) Mon programme est composé de 4 fonction principale servant chacune une tâche précise du projet, elles-mêmes regroupées dans une fonction globale permettant à la fonction main d'être complétement dénuée de variables. La première tâche composée d'une part de la détection éventuelle d'erreur dans l'input et d'autre part d'une fonction permettant la transformation de l'image pbm en matrice d'adjacence. La seconde tâche quant à elle, inclut une fonction qui génère les nœuds visité de la matrice sous forme de vecteur permettant à la tâche de déterminé si le graphe associé est connexe ou non. Les tâches 3 et 4 utilisent une fonction commune qui donne le degré de séparation des nœuds, au sein même de laquelle se trouve une fonction déterminant le critère d'arrêt à plusieurs endroits de la fonction principale. Cela permet, à la tâche 3 de donner l'ordre de passage des nœuds depuis le nœud zéro, et à la tâche 4 de calculer le degré moyen de séparation du graphe. Les tâches 3 et 4 utilisent donc une même fonction dans laquelle une fonction auxiliaire est répétée.