1. Audrey
2. Audrey
3. Le problème est divisé en 2 parties. Tout d’abord, il faut générer le Graph à partir des informations qui sont stockées dans le fichier texte. Par la suite, à partir de se Graph, il faut trouver le chemin le plus optimale pour faire le meilleur ratio de points par distance.

Afin de générer le Graph, les arcs lus dans le fichier sont d’abord triés en ordre de distance. On prend ensuite l’arc le plus court et on sait qu’il y a un arc entre ses deux extrémités (Nodes).  
Par la suite, on prend le second plus petit d’arc. On considère la première extrémité comme le point A et la seconde comme le point B. Le programme essaie de trouver, a partir des arcs déjà présents sur la graphique, un chemin du point A au point B avec comme distance même que l’arc présentement traité. Cette recherche de distance est faite méthodiquement entre tous les arcs afin d’être certain de ne pas en laisser de côté.  
Cet algorithme a une complexité exponentielle mais on gère avec un petit nombre de Nodes donc l’algorithme est raisonnable pour nos besoins. Il s’agit d’un algorithme récursif.  
Cet algorithme est répété pour chacun des arcs lus dans le fichier en ordre de distance.

Une fois le graph crée, il faut trouver le chemin optimal selon les requis de l’utilisateur. Que ce dernier veule un maximum de points pour une certaine distance ou la plus petite distance pour un certain nombre de points, l’algorithme est le même. Un chemin (Path) est créé à partir du départ. Le Path a comme attribut une distance totale parcourue et un nombre de point accumulé, ce qui permet de calculer le ratio distance/points. Le Path a aussi une Map des Nodes désactivés. Il a finalement une list des Nodes précédemment visités  
A partir de ce Path, *n* nouveaux Path sont créés où *n* est le nombre de Nodes reliés au Node actuel du Path traité. Chacun de ses Path ont un ratio distance/points différent au premier. L’algorithme retire le Path traité, ajoute les nouveaux Path a sa liste de Paths potentiels, puis les trie en ordre de ratio.  
Le Path avec le meilleur ratio est ensuite traité comme l’a été le chemin de l’itération précédente.  
Une fois qu’un Path atteint l’objectif désiré, il est mis en mémoire par l’algorithme et l’algorithme passe à la prochaine itération. Une fois que l’algorithme a trouvé 500 Path complets, il les trie en fonction de ce que l’utilisateur a demandé et renvoie le meilleur.  
Le nombre de Path potentiels gérés par l’algorithme grandit très rapidement. Afin d’éviter les erreurs OutOfMemory et accélérer le processus, seulement 500 Paths potentiels sont gardés. Cela n’a pas vraiment d’incidences sur la qualité du chemin trouvé car les Path qui sont continués sont ceux avec les meilleurs ratios et ceux qui sont enlevés sont ceux avec les pires.