M1.L5 : Série d'exercices sur la théorie du calcul [Solutions]

1 Dénombrabilité

a) Chaque client déjà présent se déplace dans la chambre portant un numéro supérieur d'une unité à celle qu'ils occupent actuellement. cela libère la chambre 1 pour le nouveau client.

b) Chaque client actuel se déplace la chambre portant le numéro qui est le double de celui de sa chambre actuelle. On peut ainsi utiliser l'infinité des numéros impairs pour les passagers du bus.

c) Plus difficile, il faut s'inspirer du dénombrement de paires d'entiers pour généraliser la partie 2. Si on imagine que les bus sont alignés sur un parking (un par ligne) on peut imaginer que les passagers de chaque bus apparaissent dans des colonnes (un par colonne). Même avec un nombre infini de lignes et de colonnes on peut construire un parcours de cette grille en logeant les paires (bus, passager) de même somme car il y a toujours un nombre fini de paires pour une somme donnée. On commence par la somme valant 2 (bus 1, passager 1), puis valant 3 (bus 1, passager 2) et (bus 2, passager 1) etc (qu'on loge dans les chambres de numéros impairs comme dans l'exercice précédent).

2 Classes de problèmes P vs NP

Seules l'affirmation a) est vraie.

3 Classe de problems P vs NP *(in english)*

The correct answer is #4.

1)We already know an O(N log N) algorithm for sorting a list, thus finding or using an exponential algorithm to do so, is a waste of resources. Also, from a theoretical perspective, any O(N log N) algorithm is also an exponential one, but we usually use the stricter bound to name the complexity of an algorithm. In either case, it would not change the status of P vs NP.

2)Equivalently to 1, using an exponential algorithm instead of a polynomial one is a waste of time.

3)Solving a specific instance of a problem (for example, coloring a specific graph, or finding a Traveling Salesman solution for a specific set of cities/distances and maximum distance) in general does not provide any information about the difficulty of the problem.

4)The Traveling Salesman is an NP problems that is at least as hard as any other NP problem (within a polynomial factor of complexity). This implies that if a polynomial solution exists for this problem, then all the NP problems are solvable in polynomial time.

5)Again, equivalently to 1, using a polynomial algorithm instead of a logarithmic one is a waste of resources.

4 Expression de l'ordre de complexité avec plusieurs paramètres

Chaque boucle **Pour** est caractérisée par un nombre de passage différent associé à un paramètre du problème, cela se reflète dans l'expression de l'ordre de complexité qui est alors en O(N*M).

5 Manipulation de grands nombres et ordre de complexité.

a) La complexité de l'addition de deux grands nombres de N chiffres est en O(N)

b) La complexité de la multiplication de deux grands nombres de N chiffres est en $O(N^2)$ car cela revient à effectuer N additions.

c) comme pour la détermination de primalité d'un nombre nous pouvons effectuer une recherche jusqu'à la racine carrée de Z car P et Q sont forcément inférieurs (ou égaux) à cette racine. On ne dispose pas de moyens particuliers pour parcourir l'ensemble des nombres premiers ; on doit donc parcourir les entiers à partir de 2 et vérifier s'ils divisent Z en produisant un reste nul dans la division entière. Le premier entier trouvé est l'un des deux opérandes P ou Q et ou obtient l'autre en divisant Z par le nombre trouvé.

Dans le pire des cas on parcourt une quantité de nombres de l'ordre de racine de Z. Or Z contient N chiffres en base 10, ce qui veut dire que la plus grande valeur de Z est proche de 10^N et celle de la racine de Z est donc proche de $10^{N/2}$. Cet ordre de grandeur exprimé en fonction du nombre N de chiffres de Z en fait une **complexité exponentielle** en $\mathbf{O(10^{N/2})}$ sans même considérer le coût d'un passage dans la boucle.

Si on analyse ce coût supplémentaire il faut **multiplier** le nombre de passage en $O(\mathbf{10^{N/2}})$ par un coût en $O(N^2)$ car le calcul du modulo en $O(N^2)$ est dominant par rapport aux comparaisons, en O(N), et à l'incrémentation qui n'est autre qu'une addition, aussi en O(N). Le bilan complet est de l'ordre de $O(\mathbf{N^2 10^{N/2}})$.

d) Mais cela veut-il dire que le problème est non-polynomial? En fait, la complexité d'un problème est la complexité du meilleur algorithme résolvant ce problème et l'algorithme de détermination de primalité exploité dans cet exercice n'est pas le meilleur. Le 6 aout 2002, 3 chercheurs indiens ont publié l'algorithme AKS (d'après Agrawal-Kayal-Saxena) dont la complexité est polynomiale; le problème est donc dans P (mais reste très couteux).

6 Coloriage de graphes

a) Pour le graphe de gauche, la réponse est oui; pour celui de droite, la réponse est non.

b) L'algorithme est le suivant : on part d'un sommet (n'importe lequel), qu'on colorie d'une couleur(disons vert). Après cela, on colorie tous ses voisins de l'autre couleur (disons rouge), puis on essaye de colorier tous les voisins des voisins à nouveau en rouge, etc. Si en procédant ainsi, on se retrouve un moment donné avec deux sommets voisins de même couleur, alors on sait qu'un coloriage de tout le graphe avec seulement deux couleurs n'est pas possible. Si par contre, on arrive jusqu'au bout sans problème, alors on a trouvé une solution au problème.

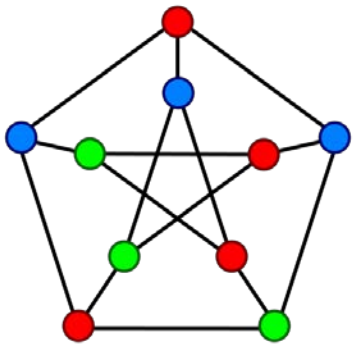
c) Le nombre d'opérations effectuer pour exécuter l'algorithme ci-dessus est le suivant : au pire, on doit parcourir tous les n sommets du graphe, et chaque sommet, on doit vérifier qu'aucun des voisins n'a déjà été colorié dans la même couleur que celle qu'on est en train d'utiliser pour colorier le sommet. Comme chaque sommet peut avoir jusqu'à $n-1$ voisins (ce sera le cas pour certains sommets dans certains graphes), on risque au pire d'effectuer $O(n^2)$ opérations en tout (mais bien sûr, il se peut qu'on puisse conclure beaucoup plus vite que a dans certains cas).

Quoi qu'il en soit, le fait qu'on n'ait besoin au pire que de $O(n^2)$ opérations pour résoudre le problème implique que le problème du coloriage d'un graphe avec deux couleurs est dans la classe P (c'est un problème soluble en un temps polynomial en le nombre de variables).

d) Pour vérifier si un coloriage donné avec k couleurs fonctionne, on parcourt simplement tous les sommets du graphe, et pour chacun des sommets, on vérifie qu'aucun de ses voisins n'est colorié de la même couleur que le sommet lui-même. Comme il y a n sommets et que chacun des sommets peut avoir au pire $n-1$ voisins, le nombre total d'opérations nécessaires pour cette vérification est nouveau $O(n^2)$. Le problème du coloriage d'un graphe avec k couleurs est donc dans la classe NP (c'est un problème tel que si on nous donne une solution du problème, alors il est possible de vérifier en un temps polynomial en le nombre de variables si cette solution est correcte ou non).

e*) Sans indication de départ, trouver un coloriage avec trois couleurs qui fonctionne pour un graphe donné est a priori (beaucoup) plus compliqué. Bien sûr, s'il se trouve qu'un coloriage avec deux couleurs fonctionne, comme dans le cas du graphe de gauche, alors automatiquement on sait dans ce cas qu'un coloriage avec trois couleurs fonctionne également (il suffit de remplacer la couleur d'un des sommets avec la troisième couleur, par exemple); en fait, beaucoup de coloriages avec trois couleurs fonctionnent dans ce cas.

Par contre, si aucun coloriage du graphe avec deux couleurs ne fonctionne (comme dans le cas du graphe de droite), alors tout se complique pour le cas de trois couleurs. Il se trouve que pour le graphe de droite, un tel coloriage existe :



Comment le trouver? (comment l'avez-vous trouvé vous-même?) On peut bien sûr essayer en tâtonnant et espérer avoir de la chance... On peut aussi essayer toutes les possibilités, comme suggéré dans l'énoncé de l'exercice, mais le nombre d'opérations nécessaires sera alors de 3^n, donc exponentiel en le nombre de variables... Est-il possible de faire mieux? A l'heure actuelle, la réponse cette question est : on ne sait pas s'il existe un algorithme capable de résoudre ce problème en un temps polynomial en n (ceci reviendrait prouver que P=NP).