Question 1

1. $5n^4 + 10n^2 + n - 100 = O(n^4)$
2. $3\log_4(n^5) + 6 = O(\ln n)$
3. $n(\log_{10} n + 10) = O(n \ln n)$
4. $\frac{3n}{n+1} = O(1)$
5. $\sqrt{n} + \left(\frac{1000}{1001}\right)^n = O(\sqrt{n})$
6. $n^10 + \left(\frac{1001}{1000}\right)^n = O(\left(\frac{1001}{1000}\right)^n)$
7. $(4n^3 + 2n^2 + 15)(25n^4 + 10n + 1) = O(n^7)$
8. $2n^2 + 2^{n+3} = O(2^n)$
9. $\frac{2^n}{3^n} = O(\left(\frac{2}{3}\right)^n) = O(1)$
10. $\frac{3^n}{2^n} = O(\left(\frac{3}{2}\right)^n)$
11. $2^{\log_2 n} + \log_2(2^n) = 0(n)$

Question 2 Tri par ordre croissant sur l'ordre induit par O

1. n^100
2. $(2^n)^2$
3. $n!$
4. n^n

5. (2^{n^2})
6. $A(n,n)$ définie par $\begin{cases} A(0,n) = n+1 \\ A(m,0) = A(m-1,1) \\ A(m,n) = A(m-1, A(m,n-1)) \end{cases}$

Question 3

1. $T(n) = 2T(\frac{n}{2}) + n^2$
2. $T(n) = 4T(\frac{n}{4}) + \sqrt{(}n)$

On va devoir utiliser le théorème :
Complexité des algos "diviser pour régner" : Cas $T(n) = aT(\frac{n}{b}) + f(n)$ avec $a \geq 1$, $b > 1$ et $T(0) = \theta(1)(= 1)$ (on peut remplacer $\frac{n}{b}$ par sa partie entière inférieure ou supérieure)

Propriété :

- Si $f(n) = O(n^{\log_b a - \varepsilon})$ $(\varepsilon > 0)$ alors $T(n) \in \theta(n^{\log_b a})$
- Si $f(n) = \theta(n^{\log_b a})$ alors $T(n) = \theta(n^{\log_b a} \ln n)$
- Si $f(n) = \Omega(n^{\log_b a + \varepsilon})$ $(\varepsilon > 0)$ et $af(\frac{n}{b}) \geq cf(n)$ pour n assez grand et c une constante < 1. Alors $T(n) \in \theta(f(n))$

J'applique ce théorème sur les deux cas :

1. $f(n) = n^2$, $n^{\log_b a - \varepsilon} = n^{\log_2 2 - \varepsilon} = n^{-\varepsilon}$. C'est le 3eme cas donc $T(n) = O(n^2)$
2. $T(n) = 4T(\frac{n}{2}) + \sqrt{(}n)$, $n^{\log_b a - \varepsilon} = n^{\log_2 4 - \varepsilon} = n^{2-\varepsilon}$. C'est le 2eme cas donc $T(n) = O(n^{\log_2 4} \ln n) = O(n^2 \ln n)$

Le premier cas est préférable.

Question 4

1. $T(n) = T(\frac{n}{2}) + \Omega(\sqrt{n})$. $a = 1, b = 2, \log_a b = \ln 2 > \frac{1}{2}$
2. $T(n) =$

Question 5

1. $O(n)$
2. Multiplication d'un chiffre par un nombre : temps constant (en binaire), puis toutes les additions font $O(n^2)$
3. Il faut calculer ac, $ad+bc$ et bd puis additionner les résultats. On divise en temps constant en 4 sous problèmes de taille $\frac{n}{2}$, puis combiner est 3 additions donc en $O(n)$ donc $T(n) = 4T(\frac{n}{2}) + O(n)$.
4. $\log_b a = \log_2 4 = 2 > 1$ donc $f(n) = O(n) << n^2$ donc on est dans le premier cas donc $O(n^2)$
5. Karatsuba propose une autre solution : $u = ac$, $v = bd$, $w = (a+b)(c+d)$, on transforme le coef 4 en coef 3. Donc on passe en complexité $O(n^{\log_2 3})$ et $\log_2 3 \approx 1,58$

Fonctions de complexité :

1. $T(n) = 2T(\frac{n}{2}) + O(1)$, $\log_b a = \log_2 2 = 1$, $1 \in O(n^{1-\varepsilon})$ donc $T(n) \in O(n)$
2. $T(n) = T(\frac{n}{2}) + O(1)$, $\log_b a = \log_2 1 = 0$, $1 \in O(n^0)$ donc $T(n) \in O(\log n)$
3. $T(n) = 2T(\frac{n}{2}) + O(n)$, $\log_b a = \log_2 2 = 1$, $n \in \theta(n^1)$ donc $T(n) \in O(n \log n)$
4. $T(n) = T(\frac{n}{k}) + O(1)$, $\log_b a = \log_k 1 = 0$, $1 \in O(n^(1 - \varepsilon))$ donc $T(n) \in O(\log n)$

Algo : On veut trouver les 2 points les plus proches dans un ensemble de points du plan.

- Algo naïf(T: Tableau de points) :

```
d = ∞
n = taille de T
Pour i de 0 à n-2 Faire
        Pour j de i+1 à n-1 Faire
                Si Dist(T[i],T[j]) < d Alors d = Dist(T[i],T[j])
        FinPour
FinPour
Retourne d
```

- Idée diviser pour mieux régner : on fait la partie gauche du plan (d_g) et la partie droite (d_d), on appelle d le min des deux. Il faut regarder dans une bande de largeur $2d$ autour de la frontière. En triant les points selon y et en observant une boule de rayon d autour de chaque point, on peut se rendre compte que dans cette bande, lorsqu'on étudie un point on a pas besoin d'aller chercher plus loin que 7 points de plus pour en trouver un plus près que d du point de départ. C'est donc en temps constant pour chaque point.

- Voir algo sur fiche. $T(n) = 2T(\frac{n}{2}) + O(n \log n)$

- $\log_b a = \log_2 2 = 1$, comparer $n \log n$ et n^1, $n \log n \in \Omega(n^{1+\varepsilon})$ donc $T(n) \in O(n \log n)$

Voyageur de commerce :

- E : Trouver une tournée la moins longue possible
- O : Quelle est la distance minimale d'une tournée ?
- D : Existe-t-il une tournée dont la distance est $\leq k$?

Montrons l'équivalence entre ces 3 versions du problèmes :

- E $\Rightarrow$ O et D : trivial
- D $\Rightarrow$ O : recherche de la distance minimale par dichotomie
- O $\Rightarrow$ E : admis

Cycles hamiltonnien :

- E : Trouver un cycle hamiltonnien dans G
- D : Existe-t-il un cycle hamiltonnien dans G ?

Montrons l'équivalence entre ces 2 versions du problème :

- E $\Rightarrow$ D : trivial
- D $\Rightarrow$ E : si le graphe n'est pas hamiltonnien, on répond impossible, sinon on enlève les arêtes tant que le graphe reste hamiltonnien.

Algos polynomialement équivalents :

- Clique : Existe-t-il une clique de taille k dans G ?
- Indé : Existe-t-il dans G un sous ensemble de sommets de taille k sans arêtes communes ?
- Couv : Existe-t-il dans G un sous ensemble de sommets de taille k tel que toute arête est adjacente à un de ces sommets ?

Montrons l'équivalence polynomiale entre ces 3 problèmes :

- Indé $\leq_p$ Clique : $R : G, k \to \bar{G}, k$
- Clique $\leq_p$ Indé : $R : G, k \to \bar{G}, k$
- Couv $\leq_p$ Indé : $R : G, k \to G, n - k$
- Indé $\leq_p$ Couv : $R : G, k \to G, n - k$

Voyageur de commerce et cycle hamiltonnien :

On part d'un graphe G, on met un poids de 1 sur toutes les arêtes. On ajoute les arêtes manquantes pour le rendre complet, avec un poids de 2.
On pose $k =$ nombre de sommets de G.
L'algo R ainsi défini transforme toute instance de Ham en une instance de Voyageur ayant la même réponse.