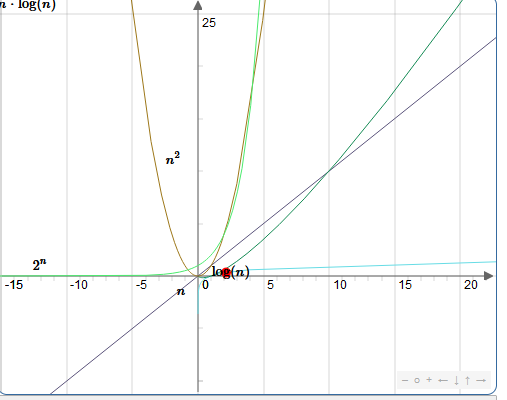
**Algo J1**

**Exercice 1: Courbe de complexité**

Donner sur le même dessin les courbes des fonctions f(n)= log(n), n, n log(n), n² et 2n



**Exercice 2 : comparaison de complexité**

Si une instruction est traitée en 10-7 seconde, quelle est la taille maximum que peut traiter un algorithme en log(n), n, n log(n), n², 2n si on dispose d'une seconde, une minute, une heure, une journée

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Log(n) | n | N\*log(n) | N^2 | 2^n |
| 1 | 12589254 | 1\*10^7 |  | 3162 | 23 |
| 60 | 1\*10^13 | 6\* 10^8 |  | 24494 | 29 |
| 3600 |  | 3,6\* 10^10 |  | 189736 | 35 |
| 86400 |  | 8.64\*10^11 |  | 929516 | 39 |

**Exercice 3 : Algorithmes de complexités différentes**

Rechercher un algorithme qui teste si un tableau de taille n représente bien une permutation (i.e tous les éléments sont distincts/différents et compris entre 1 et n)

* Donner un premier algorithme simple mais quadratique

Declare Algo1 (nombre tab[]) :

Int i,j

Pour i allant de 1 à len(tab[])-1 :

Pour j allant de 1 à (len(tab[])-1 :

Si tab[i] == tab[j]

Renvoyer faux

Renvoyer vrai

* Donner un second algorithme linéaire utilisant un deuxième tableau

Declare Algo2 (nombre tab[]) :

Int i

Verif Tab[] = tab[]

Pour i allant de 1 à len(tab[])-1 :

Si tab[i] != Verif[i] :

Renvoyer faux

Renvoyer vrai

**Exercice 4 : Ordres de grandeur**

Donner l'ordre de grandeur des expressions suivantes

* (n2 – 3n -1 ) / (n+1) **linéaire**
* (n log(n) +n² + log(n)²)/ (n+1) **logarithmique**