$a \geq b$

TD1

Exercice 1

Q1) Si $b = 0$, ça se termine,
sinon,

invariant de boucle
init : $i \leftarrow b$ donc $i \geq pgcd(a, b)$ car $a \geq b$
boucle : on suppose condition vraie en début de boucle $i \geq pgcd(a,b)$
montrer que : $i - 1 \geq pgcd(a,b)$
qq entre dans la boucle.
i ne divise pas a
i ne divise pas b
donc $i > pgcd(a,b)$
d'où $i - 1 \geq pgcd(a,b)$.
sortie boucle : i divise a et i divise b.

Q2) $a = b$ est le meilleur cas. 0 itération de la boucle.
↳ 6 opérations primitives, 2 opérations si $b = 0$.

Q3) a et b premiers entre eux, il y aura $b - 1$ itération de la boucle.

Exercice 2

TD2

1.c $O(n^2)$ n_0 en 1min. $\Rightarrow n_1^2 = 100 n_0^2 \Rightarrow n_1 = 10 n_0$

Sur l'autre ordinateur, $10 n_0$ en 1min

2.c $O(2^n)$ n_0 en 1min. $\Rightarrow 2^{n_1} = 100 \times 2^{n_0} \Rightarrow n_1 = \log_2(100) + n_0$

sur l'autre ordinateur, $\log_2(100) + n_0 \simeq 6{,}64 + n_0$

$f(n) \in O(g(n)) \Leftrightarrow f(n) \leq c \times g(n)$ $c \in \mathbb{R}_{+*}^+$

$f(n) \in \Omega(g(n)) \Leftrightarrow g(n) \leq c \times f(n)$ $c \in \mathbb{R}_{+*}^+$

$f(n) \in \Theta(g(n)) \Leftrightarrow c_1 g(n) \leq f(n) \leq c_2 g(n)$ $c_1, c_2 \in \mathbb{R}_+^*$

4

1. $1000 \in O(1)$
2. $n^2 \notin \Omega(n^3)$
3. $n^3 \in \Omega(n^2)$
4. $2^{n+1} \in O(2^n)$ car $2^{n+1} = 2^n \times 2 \leq 4 \cdot 2^n = 2^{n+2}$.
5. $(n+1)^2 \in \Theta(n^2)$
6. $n^3 + 12 \in O(2^n)$
7. $\sqrt{n} \notin \Theta(n)$
8. $\pi^{32} \in O(1)$
9. $(n-3)(n+1) \in O(n^3)$
10. $n^3 m^2 \notin O(m^6)$
11. $2^{(n-12)} \in \Omega(2^n)$
12. $3^n \notin O(2^n)$
13. $n^{3,14} \notin \Theta(n^3)$
14. $n^{3,14} \notin O(n^3)$
15. $n^{3,14} \in \Omega(n^3)$
16. $(n+1)\left(\frac{n}{2} - 1\right) \in \Theta(n^2)$
17. $n^3 + 3n^2 + n + 2017 \in O(n^3)$
18. $n^5 \notin O(n^3)$
19. $\frac{1}{2} n^2 - 3n \in \Theta(n^2)$
20. $\frac{1}{4} n^2 - \Omega(n) \notin \Omega(n^3)$
21. $\log_2(2n) \in \Theta(\log n)$ vrai
22. $\log_2(n^2) \in \Theta(\log n)$ vrai
23. $(\log_2(n))^2 \notin O(\log n)$
24. $3^n \notin O(2^n)$
25. $2^{n+3} \in O(2^n)$
26. $2^{3n} \notin O(2^n)$
27.
28.
29.
30.

TD 2

① Pour $n = 0$, 0 itération de la boucle.
Dans le pire des cas, n grand :

$\sum$

TD2

⑤ ⓐ $p(n) \in \Omega(g(n)) \Rightarrow g(n) \in O(p(n))$

$\exists c, n_0 \quad \forall n \geq n_0 \quad p(n) \geq c \cdot g(n)$

$\Rightarrow \frac{1}{c} p(n) \geq g(n)$ d'où $g(n) \in O(p(n))$

ⓑ $g(n) \in O(p(n)) \Rightarrow p(n) \in \Omega(g(n))$

$\exists c, n_0 \quad \forall n \geq n_0 \quad g(n) \leq c \cdot p(n)$

$\Rightarrow \frac{1}{c} g(n) \leq p(n)$ d'où $p(n) \in \Omega(g(n))$

⑥ $\left.\begin{array}{l} p(n) \geq 0 \\ g(n) \geq 0 \end{array}\right\} \Rightarrow O(\max(p(n), g(n))) = \Theta(p(n) + g(n))$

Soit $h(n), n \in \mathbb{N}$.

$h(n) \in O(\max(p(n), g(n)))$

$\Leftrightarrow h(n) \leq c \cdot \max(p(n), g(n))$, $c \in \mathbb{R}^+$ ($\exists n_0 \in \mathbb{N}, \forall n \geq n_0$)

$\Rightarrow h(n) \leq c \cdot (p(n) + g(n))$

$\Rightarrow h(n) \in O(p(n) + g(n))$

TD3

Exercice 1 :

$$rec_1(x) = \begin{cases} 1 & \text{si } x \leq 0 \\ 2 rec_1(x-1) & \text{sinon} \end{cases}$$

$$rec_2(x) = \begin{cases} 1 & \text{si } x \leq 0 \\ rec_2(x-1) & \text{sinon} \end{cases}$$

$$T_1(n) = \begin{cases} \Theta(1) & \text{si } x \leq 0 \\ T_1(n-1) + \Theta(1) & \text{sinon} \end{cases} \Rightarrow T_1(n) \in \Theta(n)$$

(a) $T_{rec_1}(n) \in \Omega(n)$ $\quad T_{rec_1}(x) \geq C_1 x$

* vraie au rang 1 $\quad T_{rec_1}(1) = 1 \geq C_1$

$\Rightarrow$ on peut trouver $C_1 \in \mathbb{R}_+^*$

* hyp : vraie du rang 1 à $n-1$: $\forall x \in [1, n-1], T_{rec_1}(x) \geq C_1 x$

on cherche au rang n

$$T_{rec_1}(n) = T_{rec}(n-1) + \Theta(1) \geq C_1(n-1) + \Theta(1)$$
$$\geq C_1 n - C_1 + 1$$
$$\geq C_1 n$$

* CCL : Vraie $\forall n \in \mathbb{N}^*$

(b) $T_{rec_1}(n) \in O(n)$ $\quad T_{rec_1}(x) \geq$

* vraie au rang 1 $\quad T_{rec_1}(1) = 1$

Question 2

x codé en binaire

↳ Longueur du mot binaire en fonction de la valeur

$p = \log_2(x)$

TD3

2ème algo : $T(x) = \begin{cases} \Theta(1) & \text{si } x = 1 \\ T(x/2) + \Theta(1) & \text{si } x > 1 \end{cases}$

↳ Méthode par substitution

On veut montrer $T(x) \in \Theta(\log_2(x))$ (on intuite)

$T(x) \in O(\log_2(x))$

$T(1) \not\leq c \cdot \log_2(1)$

$1 \not\leq 0$

$T(2) \leq c \times \log_2(x)$

$1 + 1 \leq c \cdot \log_2(2)$

$\boxed{2 \leq c}$

} Cas initial (x puissance de 2).

On suppose que $T(i) \leq c(\log(i)) \quad \forall i \in [2, x-1]$

$T(i) \leq c \cdot \log_2(i)$ ↓ on substitue

$T(x) = T\left(\frac{x}{2}\right) + \Theta(1)$

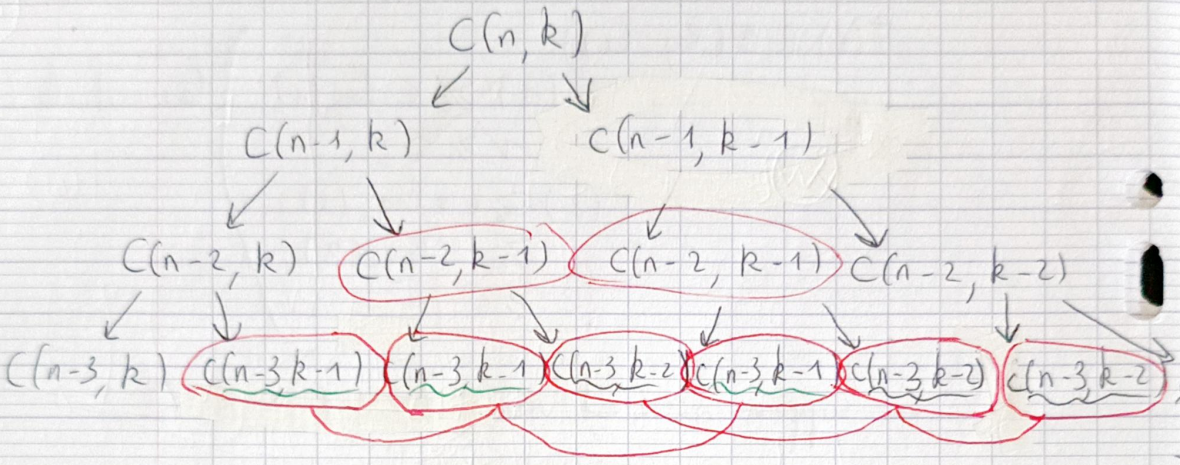
$\leq c \cdot \log_2\left(\frac{x}{2}\right) + 1$

$\leq c \cdot \log_2\left(\frac{x}{2}\right) + \underbrace{\log_2(2)}_{\leq c \log_2(2)}$ car $\boxed{c \geq 2}$

$\leq c \log_2\left(\frac{x}{2}\right) + c \log_2(2)$

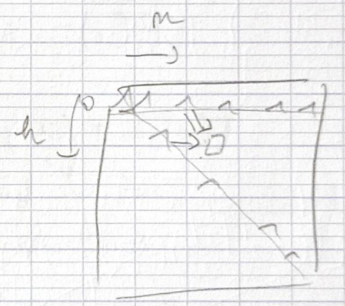
$\leq c \log_2(x)$ CQFD.

- Formule récursive de $\binom{n}{k}$
  - ↳ $C(n,k) = C(n-1,k) + C(n-1,k-1) \quad \forall k \in [1, n[$
  - ↳ $C(n,0) = 1$
  - ↳ $C(n,n) = 1$



C[][].

```
fonction bin_c(n, k)
  si k=0 OR n=k
    return 1
  sinon
    si C[n][k] existe
      return C[n][k]
    sinon
      C[n][k] = bin_c(n-1, k) + bin(n-1, k-1)
      return C[n][k].
```



TD4

Exercice 2 :

1a

- 1 seul sac à dos
- Pas de restriction de volume.
- 10kg de poids.

n objets : $(O_1, O_2, \dots O_n) \in (\mathbb{N}^{\infty})^n$

W : poid total.

x_i : O_i est-il sélectionné ?

w_i poid

v_i valeur.

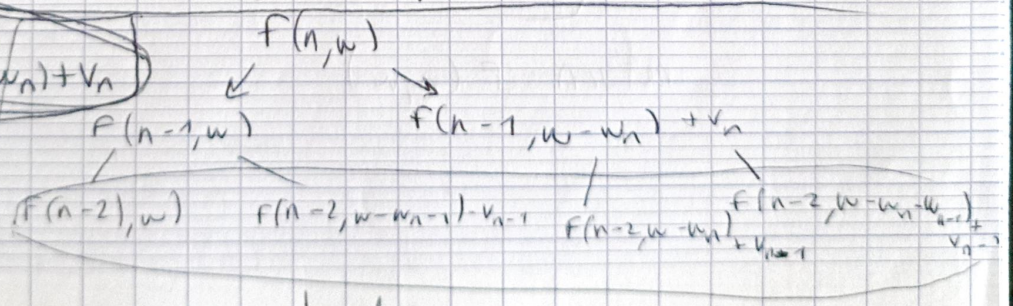
OPTIMAL : 0 objet.

F(n, W)

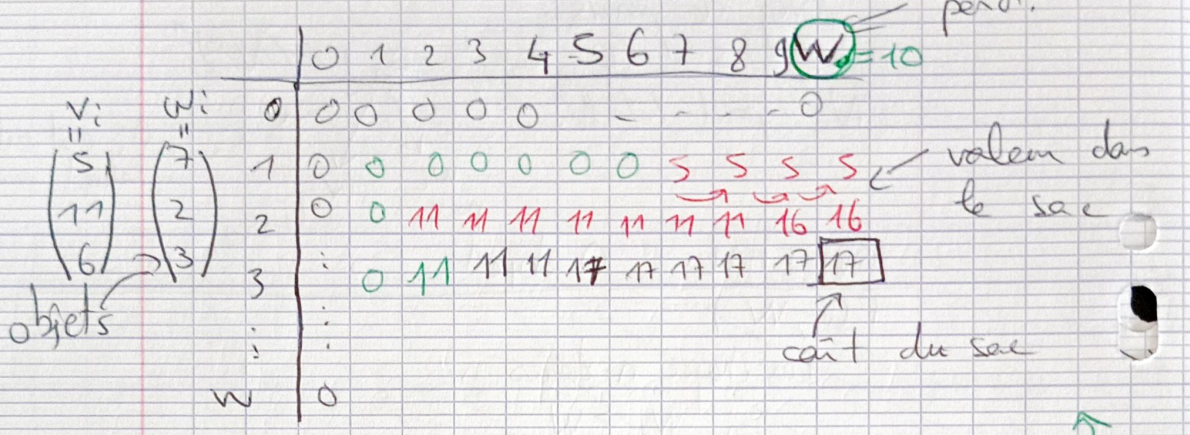
Pour chaque n_i dans n :
  Si $w_i > W$
    retirer n_i de n
  sinon
    $ratio_i = v_i / w_i$
Si $n \neq$ NULL :
  ~~return [max~~
  maxratio = n[index(max(ratio_i))]
  retirer n[maxratio]
  return [maxratio, F(n, W - (w_maxratio))]
else
  return NULL

ça sert à rien de calculer le ratio de rentabilité d'objets trop lourd

$f(n,w) = \max \{ f(n-1,w) \; ; \; f(n-1, w-w_n) + v_n \}$



Prog dynamique pour le sac à dos
⇒ utilise une matrice $F(n,w)$ dans laquelle on va construire progressivement la solution.



$n=3$; $w=10$, $W_i=(7,2,3)$; $V_i=(5,11,6)$.

- Init Matrice : pour j de 0 à w
  - $F(0,j)=0$
- Parcourir les objets et remplir la matrice
  - pour i de 1 à n :
    - pour j de 0 à w
      - si $w_j > j$ (pas assez de place).
        - $F(i,j) = F(i-1,j)$
      - sinon
        - $F(i,j) = \max(F(i-1,j),\ F(i-1, j-w_i)+V_i)$

return $F(n,w)$

TD 5

$E = \{x_1, \dots x_n\}$ et d une fonct° distance.
↳ $d(x_i, x_j) \in \mathbb{R}^+ \quad \forall (i,j) \in [1,n]^2$
↳ $d(x_i, x_i) = 0 \quad \forall i \in [1,n]$

$c(E) = \max \{ d(x_i, x_j), (x_i, x_j) \in [1,n] \}$

- Calculer distance entre ∀ les points (a,b)
- Sélectionner la distance max.
- Affecter $\begin{cases} a \to E_1 \\ b \to E_2 \end{cases}$ on divise

↳ on refait ça selon le nombre de division qu'on souhaite. (sélectionner les paires par distance max)

(x,y) la paire sélectionnée.

affectations déjà connues
| si $x \in E_1$ et $y \notin E_1 \Rightarrow y \in E_2$
| si $x \in E_2$ et $y \notin E_2 \Rightarrow y \in E_1$
| si $y \in E_1$ et $x \notin E_1 \Rightarrow x \in E_2$
| si $y \in E_2$ et $x \notin E_2 \Rightarrow x \in E_1$

affectations inconnues
| choisir le cas le moins pire $\begin{cases} x \text{ dans } E_1, y \text{ dans } E_2 \\ x \text{ dans } E_2, y \text{ dans } E_1 \end{cases}$

Complexité
1ère étape
| calcul distance : $k = n \cdot \frac{n-1}{2}$
| + Trie : $k \log_2(k)$

Itérations :
k fois
calcul distance entre 1er (pts) et ∀ les autres.
$\Rightarrow k \times n$.
$\Rightarrow k + k \times \log_2(k) + k \times n$.
avec $k = \frac{n(n-1)}{2}$ donc $\mathcal{O}(n^3)$

Exercice 2 :

F = ∅

L ← Trier E par priorité ↓

pour i de 1 à n faire

  si F ∪ {i} valide alors

    L Ajouter i dans F

fin.

return F

ex : E = {a1, a2, a3, a4}

[2 1 1 3] date

[3 1 2 3] prio

Liste triée : (a1, a4, a3, a2)

i = 1 F = [a1]

2 F = [a1 | a4]

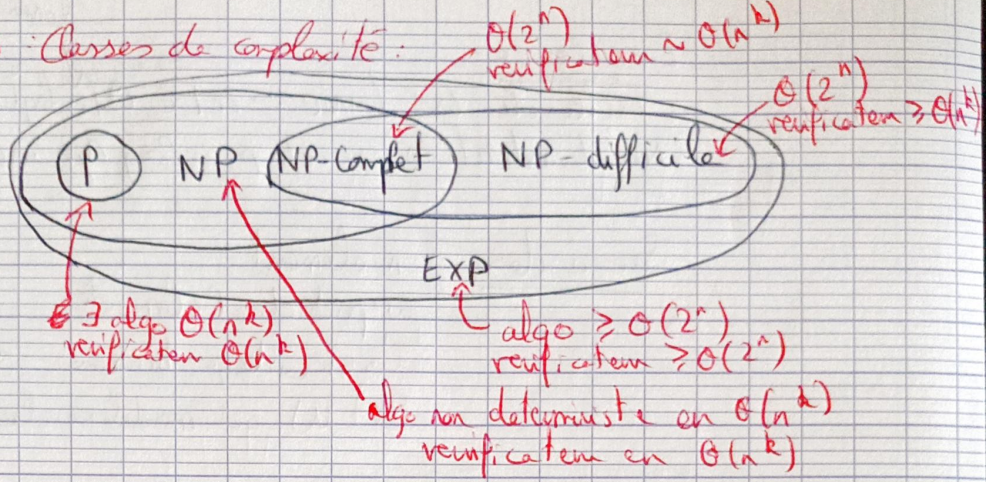
3 F = [a3 | a1 | a4]

4 F = [a3 | a2 | a1 | a4]

$a_2 \notin$ solution.

solution : F = [a3 | a1 | a4] cout = 3+3+2 = 8.

TD 8 : Classes de complexité :



Exercice 1 :

- $P \subset NP$ ✓
- $P \subset NP$ on ne sait pas (problème ouvert : $P=NP$ ou $P \neq NP$ pas de preuve)
- Si $P=NP$, alors $P=NP=$ NP-complet ✗
- Si $P \subset NP$, alors NP - NP-complet $= P$ ✗
- Si $P \subset NP$, alors NP - P = NP-complet ✗
- Si $P \subset NP$, alors $P \subset$ NP-complet. ✗
- $P \subset EXP$ ✓
- $NP \subset EXP$ ✓
- $NP \subset$ NP-complet ✗
- Si $P=NP$, alors NP-complet $- P = \emptyset$ ✓
- $NP = EXP$ ✗
- Si on a un algo $A \in \Theta(2^n)$ pour pb, alors $P_b \in EXP$ ✓
- $P_b \in NP$ ON NE SAIT PAS
- $P_b \in P$ ON NE SAIT PAS

~~Exercice 2 :~~

~~Sur à dos~~

Exercice 2 :

élément, taille encodage des poids

Taille de la donnée : $n \ell \in O(n \log(w))$

Nb d'instruction executé par ProgDyn : $O(nW)$

Sac à dos : n élément

w : poids de l'élément

↳ encodage binaire $O(\log(w))$ de bits.

Soit W qui modélise le volume.

TD7

Exercice 1 :

Q1

n, W

$(w_i)_{i \in [1,n]}$

$(v_i)_{i \in [1,n]}$

Sac-à-dos

Si on a Obj l'ensemble des n objets et W poids max, un espace de recherche peut-être.

$\mathcal{P}(Obj) \times [\![0, N]\!]$

parties de l'objet

(A, k), A partie de Obj $k \in [\![0, W]\!]$.

Taille : $2^N (W+1)$.

On rendrait $(W+1)(N+1)$.

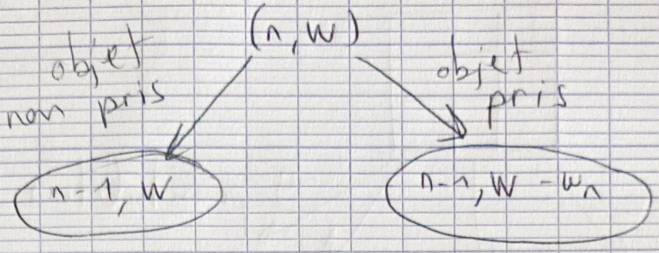
Deuxième espace plus restreint :

$[\![0, n]\!] \times [\![0, W]\!]$

Ensemble des couples (k, w) avec w poids restant.

Objet de 1 à k restants à considérer.

107



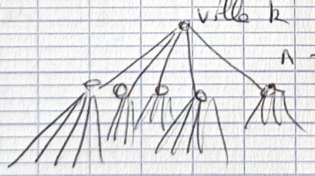
Feuille : une des deux valeurs est nulle.

E ensemble des villes restantes
$(v_1, \dots, v_k)$ villes vues dans cet ordre
C poid du chemin parcouru

Voyageur de commerce

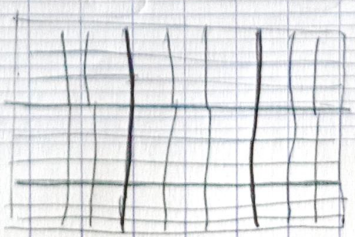
E \ {v}
$(v_1, \dots, v_k, v)$
$C + D(N_k, v)$

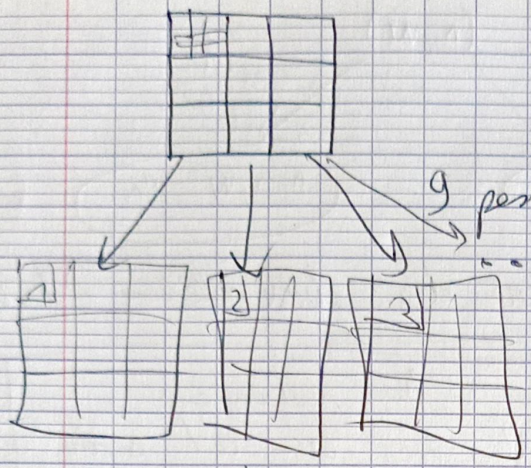
Si on prend juste E ensemble de villes restantes et C coût du chemin parcouru.
Il y a 2^n ensemble E possibles.
Si W_{max} = distance max. Il y a $[\![0, nW_{max}]\!]$.



$$= \sum \frac{(n-1)!}{k!} \sim e \cdot (n-1)!$$

Sudoku





9^{81}.

1 feuille : tout est rempli.
élagage : grand arbre à 2 chiffres identiques.

L'espace de recherche : toutes les grilles.

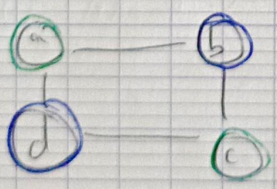
(Q2) | n, W, V
(w_i)
(v_i)

Existe-t-il un remplissage de poids $\leq W$ et de valeur $\geq V$

| avec un sac de grenaille
existe-t-il un trajet de moins de 100 km
par exemple

Exercice 2 :

(Q3) Il n'est pas 3 coloriable. En revanche.
↳ Il est complet donc nb couleurs = nb sommet.



celui ci est 2 coloriable.

TD7

Q4 borne sup : nombre d'états du graphe
borne inf :

$$\omega(G) \leq \chi(G) \leq \underbrace{\Delta(G)}_{\text{max de connexion}} + 1$$

3.A : min 2 (en réalité 4).

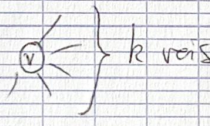
Q5 Ce graphe n'est pas 3 coloriable.

Q6 $\chi(G) = \min\left[\chi(G+(u,w)), \chi(G/(v,v))\right]$

tkt c'est quantique !!!

Q7 une feuille est le graphe obtenu lorsque ce n'est plus possible de contracter ou séparer des états (quand on a une clique).
L'espace de recherche est tous les graphes (intermédiaire compris) obtenus.

Q8



On suppose que v k-dégénéré et qu'on peut k+1 colorier G\v.
v a au plus k voisins qui utilisent au plus k couleurs. Il en reste donc au moins une parmis les k+1 pour colorier v.