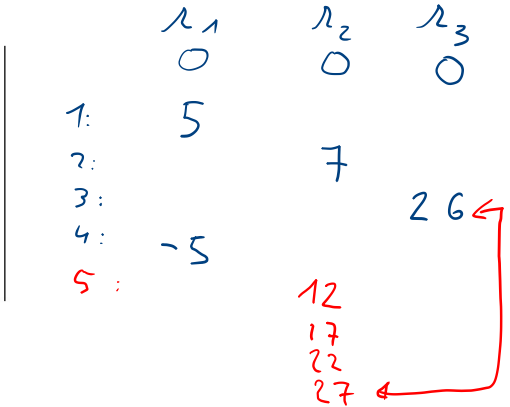
Leçon III.1 – Examen final 2016 exercice 2

En supposant que tous les registres sont à zéro au départ, que contient le registre `r2` à la fin du programme ci-dessous.

```
1: soustr  r1,  5, r2
2: soustr  r2,  7, r3
3: soustr  r3, 26, r3
4: soustr  r1,  0, r1
5: soustr  r2, r2, r1
6: cont_pp r2, r3, 5
```



Leçon III.1 – Examen final 2016 exercice 2

En supposant que tous les registres sont à zéro au départ, que contient le registre `r2` à la fin du programme ci-dessous.

```
1: soustr  r1,  5, r2
2: soustr  r2,  7, r3
3: soustr  r3, 26, r3
4: soustr  r1,  0, r1
5: soustr  r2, r2, r1
6: cont_pp r2, r3, 5
```

ligne	r1	r2	r3
1	$5-0=5$	0	0
2	5	$7-0=7$	0
3	5	7	$26-0=26$
4	$0-5=-5$	7	26
5	-5	$7-(-5)=12$	26
5	-5	$12-(-5)=17$	26
5	-5	$17-(-5)=22$	26
5	-5	$22-(-5)=27$	26

Leçon III.1 – Examen final 2015 question 11

L'instruction « `charge_adr r1, r2` » met dans le registre `r1` la valeur stockée en mémoire à l'adresse contenue dans `r2` (pointeur). Par exemple si `r2` contient la valeur 123, cette instruction mettra dans `r1` la valeur stockée en mémoire à l'adresse 123.

L'instruction « `ecrit_adr r1, r2` » écrit en mémoire à l'adresse contenue dans `r2` la valeur stockée dans `r1`. Par exemple si `r2` contient la valeur 123 et `r1` la valeur 45, cette instruction écrira la valeur 45 à l'adresse 123 en mémoire.

Si l'on suppose que l'on exécute toujours ce programme avec une valeur de `r1` strictement supérieure à celle de `r0`, et si l'on note *n* la différence entre la valeur contenue dans `r1` et celle de `r0`, **quelle est** alors en fonction de *n* **la complexité** du programme assembleur suivant :

```
 1: charge r2, r0
 2: somme r3, r2, 1
 3: charge_adr r4, r2
 4: charge_adr r5, r3
 5: continue_ppe r4, r5, 9
 6: ecrit_adr r4, r3
 7: ecrit_adr r5, r2
 8: continue 1
 9: charge r2, r3
10: continue_ppe r2, r1, 2
```

Charge adr r_1 r_2 : $r_1 = *r_2$;

Leçon III.1 – Examen final 2015 question 11

Entrées: r_1 r_0

```
1: charge r2, r0
2: somme r3, r2, 1
3: charge_adr r4, r2
4: charge_adr r5, r3
5: continue_ppe r4, r5, 9
6: ecrit_adr r4, r3
7: ecrit_adr r5, r2
8: continue 1
9: charge r2, r3
10: continue_ppe r2, r1, 2
```

$r_4 \leq r_3$

$r_4 > r_5$

$r_2 \leq r_1$

$r_2 \leftarrow r_0$

do

$r_3 \leftarrow r_2 + 1$

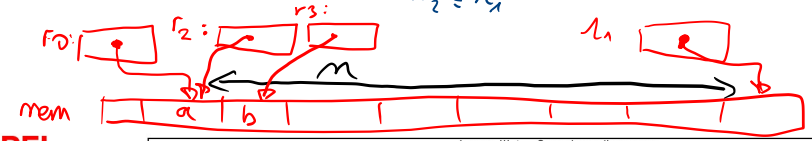
$r_4 \leftarrow *r_2$

$r_5 \leftarrow *r_3$

if $(r_4 > r_5)$ { swap $*r_3 = r_4$; $*r_2 = r_5$; recommence }

$r_2 \leftarrow r_3$

tant que $r_2 \leq r_1$



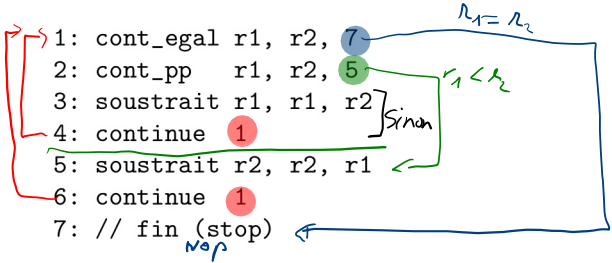
Leçon III.1 – Examen final 2015 question 11

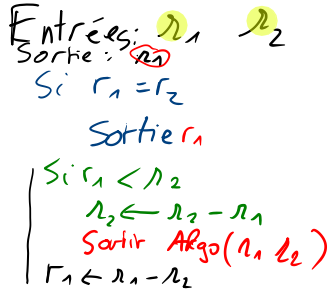
```
 1: charge r2, r0
 2: somme r3, r2, 1
 3: charge_adr r4, r2
 4: charge_adr r5, r3
 5: continue_ppe r4, r5, 9
 6: ecrit_adr r4, r3
 7: ecrit_adr r5, r2
 8: continue 1
 9: charge r2, r3
10: continue_ppe r2, r1, 2
```

☞ $\Theta(n^2)$

Leçon III.1 – Examen final 2016 exercice 1

Considérons le code assembleur suivant :





- Que vaut `r1` lorsque le programme ci-dessus se termine, si au départ `r1` contenait `01000110` et `r2` `00101010` ?
- Ecrivez l'algorithme correspondant au programme ci-dessus.
- En une phrase, que fait cet algorithme (mathématiquement) ?
- Quelle est la complexité (temporelle pire cas) de cet algorithme ?

Leçon III.1 – Examen final 2016 exercice 1

Considérons le code assembleur suivant :

```
1: cont_egal r1, r2, 7
2: cont_pp   r1, r2, 5
3: soustrait r1, r1, r2
4: continue  1
5: soustrait r2, r2, r1
6: continue  1
7: // fin (stop)
```

- Que vaut `r1` lorsque le programme ci-dessus se termine, si au départ `r1` contenait `01000110` et `r2` `00101010` ?
  14 (c.-à.d. `00001110`) : $70(\texttt{r1}) - 42(\texttt{r2}) = 28 ; 42 - 28 = 14 ; 28 - 14 = 14$.
- Ecrivez l'algorithme correspondant au programme ci-dessus.
- En une phrase, que fait cet algorithme (mathématiquement) ?
- Quelle est la complexité (temporelle pire cas) de cet algorithme ?

ou

PGCD

entrée : *Deux nombres entiers (strictement positifs)* $r1$ *et* $r2$
sortie : *PGCD(*$r1$*,* $r2$*)*

Si $\mathtt{r1} = \mathtt{r2}$
  Sortir : $\mathtt{r1}$
Sinon, si $\mathtt{r1} > \mathtt{r2}$
  Sortir : PGCD$(\mathtt{r1} - \mathtt{r2}, \mathtt{r2})$
Sortir : PGCD$(\mathtt{r1}, \mathtt{r2} - \mathtt{r1})$

PGCD

entrée : *Deux nombres entiers (strictement positifs)* $r1$ *et* $r2$
sortie : *PGCD(*$r1$*,* $r2$*)*

Tant que $\mathtt{r1} \neq \mathtt{r2}$
  Si $\mathtt{r1} > \mathtt{r2}$
    $\mathtt{r1} \longleftarrow \mathtt{r1} - \mathtt{r2}$
  Sinon
    $\mathtt{r2} \longleftarrow \mathtt{r2} - \mathtt{r1}$
Sortir : $\mathtt{r1}$

Le pire cas est quand l'un des deux vaut 1 et l'autre n.
L'algorithme fait alors $n-1$ soustractions (et tests).

Si l'on néglige la complexité de la soustraction, alors la complexité est en $\Theta(\texttt{max}(\texttt{r1},\texttt{r2}))$.

Leçon III.2 (stockage & réseaux) – Points clés

- besoin de **structure** (métadonnées)
  (même « histoire » que la **convention** pour la représentation des nombres)

  ☞ compromis facilité de stockage/facilité d'exploitation
- différencier *identification*, *localisation* et *accès* (3 besoins)
  moyens : noms, adresses, routes
- structure d'un disque (métadonnées = catalogue des adresses de blocs)
- 5 couches de l'Internet (physique, lien, IP (réseau), TCP (transport), application)
- routage par commutation de paquets :
  1. TCP : A envoie, B aquitte (A renvoie si nécessaire), B remet dans l'ordre
  2. IP : routage de proche en proche

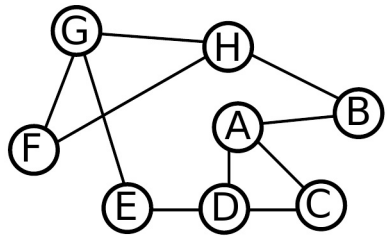
1. général

2. disque

3. réseau

Leçon III.2 (stockage & réseaux) – Etude de cas

Laquelle des lignes suivantes fait partie de la table de routage de A pour le réseau suivant (en exprimant les distances en nombre d'arcs) :



A]

dest.	dir.	dist.
F	B	3

min ? oui

B]

dest.	dir.	dist.
H	D	4

pas min (B 2) mieux

C]

dest.	dir.	dist.
D	G	2

pas voisin

D]

dest.	dir.	dist.
A	F	3

Leçon III.2 (stockage & réseaux) – Etude de cas

Dans un réseau TCP/IP, si un paquet est perdu lors de l'échange d'emails :

A] Ce n'est pas grave car les emails sont redondants.

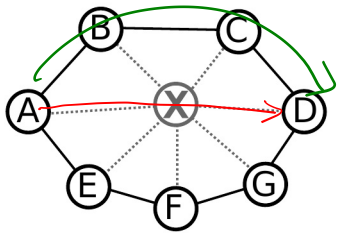
B] Le paquet est renvoyé par l'ordinateur émetteur.

C] Le paquet est renvoyé par le dernier routeur.

D] C'est qu'il a été perdu par la couche TCP car la couche IP garantit aucune perte de paquet.

Leçon III.2 (stockage & réseaux) – Etude de cas

On considère le réseau TCP/IP suivant :



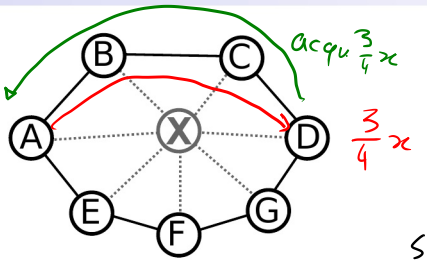
1. Donnez la ligne de la table de routage de A correspondant au nœud D lorsque le nœud X est présent, puis lorsqu'il est absent.

dst clio
D X 2

D B 3

Leçon III.2 (stockage & réseaux) – Etude de cas

On considère le réseau TCP/IP suivant :



2. On envoie un message de A vers D (X absent). Ce message est décomposé en 50 paquets. Sachant que
   - chaque nœud met 10 ms à transmettre un paquet au nœud suivant ;
   - on négligera tout autre temps ;
   - en moyenne 1 paquet sur 4 est malheureusement perdu ;

   quel est (en moyenne) le temps total de transmission de ce message, lorsque X est absent ?

$t = 60 + 10(x-1)$

Leçon III.2 (stockage & réseaux) – Etude de cas

À chaque fois que A envoie x paquets à D :

- 1/4 de ces paquets est perdu, ce dont A se rend compte par le fait que D n'a pas envoyé de paquet d'acquittement ;
- **mais aussi** 1/4 des $(1-1/4)\cdot x$ paquets d'acquittement renvoyés par D dont perdus, donc A considère les paquets de départ correspondants aussi comme perdus.

Donc au final pour x paquets, A doit renvoyer

$$x/4 + x(1-1/4)/4 = (2\times 4-1)x/4^2 = 7x/16$$

paquets.

Et cela tant que A doit renvoyer des paquets, donc un nombre $n+1$ de fois tel que $(7/16)^n \times 50 = 1$, soit $\log 50/\log(7/16)+1 = 5+1$ fois (pas demandé).

Leçon III.2 (stockage & réseaux) – Etude de cas

Du point de vue du temps, comme il y a 3 nœuds de transmission, le premier paquet met 3×10 ms à arriver à D et son acquittement 3×10 ms à revenir à A (**latence**). Ensuite (**débit**), un paquet (mais aussi son acquittement) arrive toutes les 10 ms. Donc globalement pour un total de x paquets, la communication dure $6 \times 10 + (x-1) \times 10 = (x+5) \times 10$ ms.

Deux remarques :

1. en réalité A attendrait certainement plus longtemps, mais cela n'est pas du tout considéré ici puisque nous avons clairement dit que tout autre temps était négligé ;
2. on pourrait ne pas considérer le temps d'acquittement des tous derniers paquets de la communication globale, mais c'est un détail ici.

Et donc, globalement, la transmission de 50 paquets prend :

$$10 \times \Big(50 + 50 \times 7/16 + 50 \times (7/16)^2 + \ldots + 1 + 5\Big) = 10 \times \Big(\overbrace{50 + 22 + 10 + 4 + 2 + 1 = 89}^{} \; 50 \times \sum_{i=0}^{5} (7/16)^i \; + 5\Big)$$

(ce qui, avec une calculette, mais ce n'était évidemment pas attendu en examen, donne au final 940 ms).

Une réponse moins précise, mais acceptée en examen car elle garde les aspects essentiels des communications TCP/IP (aller-retour, donc facteur 2 pour simplifier (donc $2 \times \frac{1}{4} = \frac{8}{16}$ au lieu de $\frac{7}{16}$) ; et pertes aussi sur la seconde, troisième, ..., communication), pourrait être (en considérant les communications de chaque paquet comme séquentielles nœud après nœud, c.-à-d. en ignorant le parallélisme des nœuds) :

$$3 \times 10 \times 50 \times 2 \times \left(1 + 1/4 + (1/4)^2 + ...\right)$$

qui donnerait :

$$\simeq 6 \times 10 \times (50 + 12.5 + 3 + 1) \simeq 4000 \text{ ms}$$