Semaine 1 : Série d'exercices introductive [Solutions]

1 Culture générale informatique

1.1 Quiz informatique

1. En algèbre booléenne, laquelle de ces propriétés est toujours vraie ?

   A. $a + 0 = 0$ B. $a + 0 = 1$ C. $a + 0 = a$ ✓ D. $a + 1 = a$

   Explication : Quelle que soit l'algèbre que vous utilisez, additionner 0 ne change pas la valeur (c'est même la définition de 0).

   Lien avec le cours : L'algèbre de Boole ne sera pas présentée en tant que telle (nous resterons au niveau de la logique elle-même), mais elle est utilisée en informatique pour réaliser des circuits électroniques qui *calculent* avec des propositions logiques.

2. Quelle est la meilleure complexité ?

   A. $\Theta(\log(n))$✓ B. $\Theta(n)$ C. $\Theta(n \cdot \log(n))$ D. $\Theta(n^2)$

   Explication : La meilleure complexité est la *plus petite* (lorsque n tend vers l'infini).

   Lien avec le cours : La notion de complexité algorithmique sera présentée en leçon I.1.

3. Qu'appelle-t-on le « sel » dans le cadre d'un système de mots de passe ?

   A. des caractères que l'on enlève au mot de passe
   B. des caractères aléatoires que l'on ajoute au mot de passe ✓
   C. l'empreinte d'un mot de passe
   D. la longueur de l'empreinte d'un mot de passe

   Explication : Cela permet d'augmenter la sécurité du stockage de leur forme cryptée (pour vérification ultérieure).

   Lien avec le cours : Les systèmes de mots de passe et leur qualité/complexité/sécurité seront présentés dans la dernière leçon.

4. Qu'est-ce qui caractérise un algorithme glouton ?

   A. Il est optimal
   B. Il est gourmand en mémoire
   C. Il est gourmand en temps
   D. Il ne remet jamais en cause un choix passé ✓

   Explication : Un algorithme glouton est un algorithme de recherche qui à chaque étape choisit la meilleure solution (à ce stade). Il n'est donc pas sûr du tout d'arriver à la meilleure solution globale à la fin ; un peu comme un « glouton » qui mangerait à chaque plat le maximum sans forcément pouvoir arriver à goûter au meilleur plat ou à finir le repas (il vaudrait peut être mieux manger un peu de tout).

   Lien avec le cours : Nous ne présenterons pas les algorithmes gloutons en tant que tels, mais plusieurs stratégies de constructions d'algorithmes seront présentées dans la leçon I.2.

5. Lequel de ces mots de passe est le MOINS robuste à une attaque par dictionnaire, suivi d'une attaque par force brute ?

   A. bkcpq B. AkQr C. A8Pk D. hirondelle ✓

   Explication : Ce n'est pas la longueur d'un mot de passe qui assure sa robustesse, mais son *entropie*, qui mesure, d'une certaine façon, la difficulté à deviner le mot. Dans une attaque au dictionnaire, l'entropie d'un mot présent dans le dictionnaire est 0 (on est sûr de le trouver).

   Lien avec le cours : L'entropie d'un message sera présentée, pour un modèle simple, en leçon II.3. Les systèmes de mots de passe et leur qualité/complexité/sécurité seront présentés dans la dernière leçon (sécurité).

6. Dans la base hexadécimale, de combien de symboles a-t-on besoin pour écrire les nombres ?

   A. 6 B. 10 C. 12 D. 16 ✓

Explication : *hexa*-décimal veut dire 16. On peut compter en n'importe quelle base (sauf 0 et 1), y compris dans des bases plus grandes que 10.

Lien avec le cours : Les représentations des nombres, en particulier en binaire, seront abordées lors de la leçon I.4.

7. Combien d'octets contient un téraoctet ?

   A. 10^4 B. 10^6 C. 10^9 D. 10^{12} ✓

   Explication : 10^6, c'est « méga » et 10^9, « giga ».

   Lien avec le cours : Les différentes mesures de tailles seront présentées dans la leçon I.4.

8. L'architecture de von Neumann se compose de 4 parties distinctes : l'unité arithmétique et logique, l'unité de contrôle, les entrées-sorties et...

   A. La mémoire ✓ B. Le programme C. L'écran D. La carte-mère

   Explication / Lien avec le cours : Voir la leçon III.1 qui présentera l'architecture de Von Neuman, schéma de base de tous les ordinateurs.

9. Combien de mots de passe différents peut-on composer avec 7 lettres majuscules ?

   A. 26^7 ✓ B. 7^{26} C. 52^7 D. 7^{52}

   Explication : Vous avez 26 choix pour chacune des 7 lettres, donc au total $26 \times 26 \times 26 \times 26 \times 26 \times 26 \times 26$.

   Lien avec le cours : Les systèmes de mots de passe et leur qualité/complexité/sécurité seront présentés dans la dernière leçon.

10. Comment s'appelle le système mis en place pour différencier un ordinateur d'un humain ?

    A. captcha ✓ B. firewall C. botnet D. phishing

    Lien avec le cours : cf. leçon III.4.

11. Un algorithme qui contient un appel à lui-même est dit...

    A. récursif ✓ B. autochtone C. auto-référent D. itératif

    Lien avec le cours : Les algorithmes récursifs seront présentés dans la leçon I.2.

12. Que vaut ce nombre écrit en base 2 : 1010110 ?

    A. 86 ✓ B. 84 C. 80 D. 76

    Explication : 1010110 en binaire se décompose comme suit : $1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 = 64 + 16 + 4 + 2 = 86$.

    Lien avec le cours : L'écriture en binaire sera présentée dans la leçon I.4.

1.2 Qu'est-ce qui est difficile ?

Problème a) Déterminer si un nombre donné x est premier est a priori difficile : une façon directe de faire est de chercher à savoir si le contraire est vrai, c.-à-d. si x est divisible par un nombre premier plus petit que lui. Par exemple, pour savoir si 29 est premier, on essaye de le diviser par 2, puis 3, puis 5, puis 7 (et c'est tout : en effet, on n'a pas besoin de tester tous les nombre premiers jusqu'à 29, mais seulement ceux jusqu'au nombre premier venant juste après $\sqrt{29}$; vous devinez pourquoi ?). Bref, si le nombre x est composé lui-même de 20 chiffres, on va devoir a priori tester tous les nombres premiers plus petits que $\sqrt{x}$, qui est lui composé de 10 chiffres environ, et ça fait du monde (de l'ordre de 10^{10} possibilités, donc un nombre exponentiel en la taille du nombre) !
Notez cependant qu'il existe aujourd'hui des algorithmes sensiblement plus performants que ça qui résolvent le problème en un temps dit « polynomial » en la taille du nombre (algorithme de Lenstra–Pomerance (2005) en n^6, où n est le nombre de chiffres).

Problème b) Additionner deux nombres entiers x et y, chacun composé de 20 chiffres, est par contre facile et ne vous prendra que 20 opérations en moyenne (c.-à-d. de l'ordre de la taille du nombre).

Problème c) Multiplier deux nombres entiers x et y, chacun composé de 20 chiffres, est aussi facile, mais demande a priori 20x20=400 opérations (c.-à-d. de l'ordre du carré de la taille du nombre), si on suit l'algorithme simple que vous avez appris à l'école primaire. Là encore, aujourd'hui, il existe des algorithmes qui permettent de réduire le nombre d'opérations à effectuer (sans toutefois descendre plus bas que le

nombre d'opérations à effectuer pour une simple addition (problème b) : algorithme de Harvey et van der Hoeven (2021) en $n \log n$, où n est le nombre de chiffres).

Problème d) Il existe de nombreux algorithmes pour trier une liste de 20 nombres arbitraires dans l'ordre croissant. Les meilleurs nécessitent un peu plus de 20 opérations (plus précisement, si n est la longueur de la liste, alors de l'ordre de $n \cdot \log n$ opérations sont nécessaires). Vous reverrez cela en détail dans les cours qui suivent.

Problème e) Le meilleure manière pour trouver un nombre choisi au hasard dans une liste de 20 nombres est d'utiliser l'algorithme dit « de dichotomie » qui nécessite de poser 4 à 5 questions seulement (plus précisement, si n est la longueur de la liste, alors de l'ordre de $\log n$ opérations sont nécessaires). A nouveau, vous reverrez cela en détail dans les cours qui suivent.

Notez qu'ici on ne fait pas une recherche ordonnée, « alphabétique » ; il n'est donc pas nécessaire, ici, de trier la liste. On peut par exemple demander comme première question : « *est-ce que ton nombre est parmi* { 4, 5.5, -4, 10, 0, 1200, 47, 25, -707, 101 } ? », etc. [1]

Problème f). Déterminer le meilleur coup à jouer aux échecs pour obtenir la meilleure position possible 20 coups plus tard reste un problème très difficile. En supposant qu'on puisse jouer une dizaine de coups « raisonnables » à chaque fois, on reste avec 10^{20} possibilités à analyser ! C'est l'incroyable puissance de calcul des ordinateurs modernes, et bien sûr aussi toute une série d'innovations dans le monde des algorithmes, qui ont permis de réaliser des ordinateurs capables d'évaluer un grand nombre de possibilités, de manière à finalement battre les meilleurs joueurs d'échecs humains.

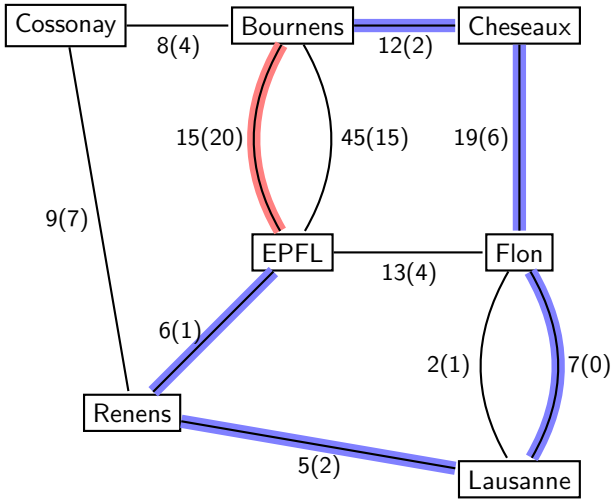
Donc le classement final est (du plus facile au plus difficile) : **e), b), d), c), a), f)**.

---

2 Histoires de chemins

2.1 Tous les chemins mènent à l'EPFL

Afin de mieux visualiser le problème, le plus simple est de le représenter sous forme d'un graphe. Les nombres sur les arrêtes représentent le temps de parcours et, entre parenthèses, le coût.



1. Ceci dit, c'est un peu plus subtil car cette façon de poser les questions est elle-même en fait plus complexe que la recherche : poser la première question nécessite d'écrire la moitié de la liste, poser la seconde le quart, etc. et ainsi de suite. Le fait même de poser ces questions a donc une complexité qui dépend de la taille de la liste, une complexité qui est de l'ordre de n. On peut alors plutôt poser les questions sous la forme : « *est-ce que ton nombre est dans la 1re moitié de la liste* ? », puis « *est-il dans la 1re moitié de* ...(ce qui reste)... ? », etc. De cette façon, le fait même de poser une question ne dépend pas de la taille de la liste et le tout est donc de l'ordre de $\log n$ opérations. Mais en procédant de la sorte, ce n'est en fait pas une valeur que l'on cherche, mais bien une position (dans l'ensemble de départ). Et l'ensemble des positions, lui, est par nature ordonné.
Tout cela pour dire que le calcul de la complexité peut être assez subtil et *surtout* qu'il dépend fortement de la définition *précise* de la tâche à accomplir.

Selon ces données, le moyen le plus rapide de venir à l'EPFL est donc de prendre la voiture directement de Bournens à l'EPFL (en rouge sur le graphe). Pour le chemin le plus économique, il faut passer par Cheseaux, Lausanne Flon, Lausanne Gare, Renens et EPFL, pour un total de 11.- CHF (en bleu sur le graphe).

transport	départ	arrivée	durée	coût
voiture	Bournens	EPFL	15 min	20 CHF
vélo	Bournens	EPFL	45 min	15 CHF
Bus	Bournens	Cossonay	8 min	4 CHF
Train	Cossonay	Renens	9 min	7 CHF
Bus	Bournens	Cheseaux	12 min	2 CHF
Train	Cheseaux	Lausanne Flon	19 min	6 CHF
M1	Renens	EPFL	6 min	1 CHF
M1	Lausanne Flon	EPFL	13 min	4 CHF
M2	Lausanne Flon	Lausanne Gare	2 min	1 CHF
train	Lausanne Gare	Renens	5 min	2 CHF
pieds	Lausanne Flon	Lausanne Gare	7min	0 CHF

2.2 Le marchand itinérant

1. Comptons le nombre d'itinéraires possibles. Le marchand choisit la première ville parmi les N villes disponibles. Pour la 2e ville de l'itinéraire, il peut choisir l'une des $N-1$ villes qu'il n'a pas choisie comme point de départ. Plus généralement, pour la n-ième ville de l'itinéraire, il pourra librement choisir parmi les $N-n$ villes qu'il n'a pas visitées précédemment. Le nombre total d'itinéraires est donc

$$N \cdot (N-1) \cdot (N-2) \cdots 3 \cdot 2 \cdot 1 = N!.$$

2. Il y a $N = 162$ villes que le marchand souhaite visiter. Le nombre d'itinéraires est donc de 162!. Essayons de comparer ceci au nombre d'atomes d'hydrogène dans l'univers ($\approx 10^{80}$) et de secondes depuis le Big Bang ($\approx 4 \times 10^{17}$.) Par exemple en utilisant la formule de Stirling :

$$N! \approx \left(\frac{N}{e}\right)^N \sqrt{2\pi N} \approx \left(\frac{162}{2.718}\right)^{162} \sqrt{2 \cdot 3.142 \cdot 162} \approx 59.596^{162} \cdot \sqrt{1017.876}$$

En utilisant le fait que $x^y = 10^{y \cdot \log_{10}(x)}$, on arrive à :

$$162! \approx \sqrt{1018} \cdot 10^{288} \approx 3 \cdot 10^{289}.$$

Si on ne connaît pas la formule de Stirling, on peut essayer de trouver une borne inférieure pour la factorielle, en bornant chaque facteur du produit par un nombre plus petit :

$$\begin{aligned} 162! &= 162 \cdot 161 \cdots 100 \cdot 99 \cdots 11 \cdot 10 \cdot 9 \cdots 2 \cdot 1 \\ &\geq \underbrace{100 \cdot 100 \cdots 100 \cdot 100}_{63 \text{ fois}} \cdot \underbrace{10 \cdot 10 \cdots 10 \cdot 10}_{90 \text{ fois}} \cdot \underbrace{1 \cdots 1 \cdot 1}_{9 \text{ fois}} = 10^{216}. \end{aligned}$$

Dans les deux cas, on se rend compte que le nombre d'itinéraires à considérer est bien plus grand que nos deux nombres de référence par des centaines de puissance de 10 d'ordres de grandeur :

— si l'on pouvait décrire un itinéraire avec un seul atome d'hydrogène, il faudrait la quantité d'hydrogène disponible dans environ 10^{209} univers ;

— si l'on pouvait calculer un milliard (10^9) d'itinéraires par seconde, il faudrait près de 10^{263} fois la durée qui s'est écoulée depuis le Big Bang.

Tout ça pour seulement 162 villes ! ! La stratégie du marchand est donc peu recommandée.

2.3 Les sept ponts de Königsberg

1. Sur la figure, on observe qu'il y quatre parties de la ville qui sont séparées par les eaux : la partie nord, la partie sud, l'île au centre, et la bande à l'est. Une façon d'abstraire le problème et de le rendre plus facile à analyser est de représenter chaque partie par un point, et de dessiner un trait entre deux points, comme sur la Figure 1. On a alors réduit les données le langage des *graphes* : un graphe contient des *noeuds* (les points) et des *arêtes* (les traits reliant les noeuds). Dans cette nouvelle formulation, le problème devient alors la recherche d'un parcours dans le graphe qui commence et se termine sur un même noeud, et passe exactment une fois sur chaque arrête.

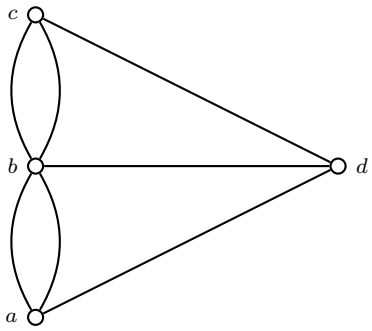


FIGURE 1 – Le problème des 7 ponts de Königsberg représenté sous forme de graphe.

On observe que chaque noeud est relié aux autres noeuds par un nombre impair d'arêtes : le noeud central est atteint par 5 arêtes, alors que les trois autres sont atteints chacun par 3 arêtes. On peut alors s'apercevoir que c'est impossible de trouver un parcours qui passe par chaque arête et revient au noeud de départ. Chaque fois que le parcours passe par le noeud de départ et le quitte, il faut qu'il puisse y rentrer à nouveau, ce qui n'est possible que si le noeud est atteint par un nombre *pair* d'arêtes.

Il est possible de faire cette observation sans passer par une représentation sous forme de graphe. Il suffit de remarquer que chaque partie de la ville est reliée aux autres par un nombre impair de ponts ; le même raisonnement s'applique ensuite.

2. En ajoutant de nouveau ponts, on peut facilement trouver une solution, comme le montre la figure 2. La solution indiquée sur la figure n'est pas la seule possible.

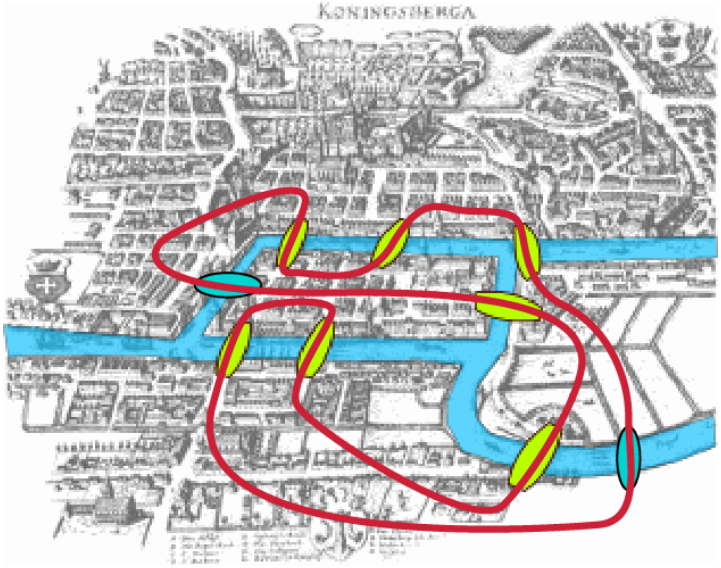


FIGURE 2 – En construisant deux ponts de plus, il est possible de résoudre le problème proposé par Euler.

Il s'avère qu'une condition à la fois nécessaire et suffisante pour trouver un tel parcours est que tous les noeuds du graphes soient atteints par un nombre pair d'arêtes. De façon équivalente, il est nécessaire et suffisant que chaque partie de la ville soit connectée aux autres parties avec un nombre pair de ponts [2].

2. Pour être tout à fait précis, il faut aussi qu'il soit possible d'aller de n'importe quelle partie de la ville à n'importe quelle autre — une propriété qui s'appelle *connectivité* dans le langage des graphes.