Alchimie

Épreuve pratique d'algorithmique et de programmation
Concours commun des Écoles normales supérieures

Durée de l'épreuve : 3 heures 30 minutes

Juin 2025

ATTENTION !

N'oubliez en aucun cas de recopier votre u_0
à l'emplacement prévu sur votre fiche réponse

Important.

Il vous a été donné un numéro u_0 qui identifie les fichiers d'entrée pour vos codes. Ceux-ci se trouvent dans un répertoire `data/u0/`, où `u0` est remplacé par votre numéro u_0. Les réponses attendues sont généralement courtes et doivent être données sur la fiche réponse fournie à la fin du sujet. À la fin du sujet, vous trouverez en fait deux fiches réponses. La première est un exemple des réponses attendues pour le numéro particulier $\widetilde{u_0}$. Cette fiche est destinée à vous aider à vérifier le résultat de vos programmes. Vous indiquerez vos réponses (correspondant à votre u_0) sur la seconde et vous la remettrez à l'examinateur à la fin de l'épreuve.

En ce qui concerne la partie orale de l'examen, lorsque la description d'un algorithme est demandée, vous devez présenter son fonctionnement de façon schématique, courte et précise. Vous ne devez en aucun cas recopier le code de vos procédures !

Quand on demande la complexité en temps ou en mémoire d'un algorithme en fonction d'un paramètre n, on demande l'ordre de grandeur en fonction du paramètre, par exemple : $O(n^2)$, $O(n \log n)$, etc. La lecture de l'instance par le code qui vous a été fourni ne sera pas prise en compte dans la complexité.

Il est recommandé de commencer par lancer vos programmes sur de petites valeurs des paramètres et de ***tester vos programmes sur des petits exemples que vous aurez résolus préalablement à la main ou bien à l'aide de la fiche réponse type fournie en annexe.***

Enfin, il est recommandé de lire l'intégralité du sujet avant de commencer afin d'effectuer les bons choix de structures de données dès le début.

Les Parties 2 et 4 doivent être implémentée en C (à l'aide du fichier `base.c`), et la Partie 3 doit être implémentée en OCaml (à l'aide du fichier `base.ml`). Ces parties sont indépendantes.
Les deux fichiers susmentionnés vous sont fournis dans le dossier `base`. Au même niveau se trouve dossier `data/` qui contient les jeux de tests. Il est recommandé de garder une sauvegarde de tous les fichiers fournis, au cas où ceux-ci seraient modifiés par erreur.
Il est demandé de nous fournir sur votre clef USB vos fichiers sources. Ces consignes doivent impérativement être suivies.
Les instances sont dans des fichiers nommés `u0/entree-Q.x.txt` où u_0 est le numéro qui vous a été attribué, Q est le numéro de la question et x la lettre de la sous-question. Les instances pour $\widetilde{u}_0$ et $x =$ `a` sont les mêmes pour toutes les questions et illustrées en Figure 1. Nous l'appelons l'instance de test. Ainsi, vous pourrez vous servir de cette figure pour comprendre la sortie de vos programmes et les déboguer.

1 Préliminaires

Depuis l'Antiquité circule la théorie que le monde est construit à partir de quelques éléments d'origine. Leur combinaison peut former d'autres éléments, qui à leur tour peuvent être combinés de manière répétée pour enfin obtenir le monde tel que nous le connaissons. Aujourd'hui nous allons travailler avec ce modèle.
Des codes sources `base.c` et `base.ml` vous permettent de lire les fichiers d'entrée.

- En C, la fonction `read_from_file` prend en argument le nom du fichier et écrit dans les variables globales décrites plus bas.
- En OCaml, la fonction `read_from_file` prend en argument le nom du fichier et renvoie une valeur du type `univers` défini par :

```
type univers = int * int array * int array array
```

  Un univers `u` est un triplet `(n, produit, ingredients)` ayant le même sens que les variables C correspondantes (voir ci-dessous).

n le nombre d'éléments dans l'univers. Ces éléments sont numérotés de 0 à $n-1$.

m (Seulement en C) le nombre de recettes. Ces recettes sont numérotées de 0 à $m-1$. En OCaml cette valeur est `Array.length produit`.

produit un tableau de taille m avec le numéro du produit de la recette.

degre (Seulement en C) un tableau de taille m avec le nombre d'ingrédients par recette. En OCaml le degré de la r-ème recette est `Array.length ingredients.(r)`.

ingredients un tableau à deux dimensions, avec les ingrédients de chaque recette. Pour tout $0 \leq r < m$ et $0 \leq k <$ degre$[r]$ le numéro du k-ème ingrédient de la r-ème recette est `ingredients[r][k]` en C et `ingredients.(r).(k)` en OCaml. Tous les ingrédients d'une même recette sont distincts.

Pour l'analyse de complexité on considère également un paramètre $D = \sum_{r=1}^{m} d_r$, qui est défini comme la somme des degrés des recettes.
Le fichier `base.c` fournit également la constante **MAX=5000**, qui est une borne supérieure sur n, sur m et sur les degrés. Cependant, le degré maximum des instances données est seulement 10.

Notez que pour tout élément de l'univers, il n'y a pas forcément exactement une recette qui le produit, il peut y en avoir aucune voire plusieurs.

Un monde est ensemble d'éléments de l'univers. Chaque élément est soit absent du monde, soit existe en quantité arbitrairement grande. Une recette est active, si son produit n'est pas encore dans le monde, et si tous ses ingrédients sont dans le monde. Le déclenchement d'une recette active consiste à ajouter son produit dans le monde. Ses ingrédients restent dans le monde, car ils existent en quantité arbitrairement grande.
À titre d'exemple, en Figure 1, la recette 16 indique que les éléments 0 (temps) et 1 (chaos) produisent l'élément 8 (ordre).

2 Partie en C : monde point fixe

2.1 Recettes actives

Dans le monde initial, l'univers est exempt de tout élément. Puis arrive le Big Bang, qui rajoute au monde tous les éléments produit par des recettes sans ingrédient (de degré 0).

Question 1 *Implémentez en C un programme qui identifie les recettes actives immédiatement après le Big Bang. Il devra calculer la somme des indices de ces recettes.*

a) `u0/entree-1.a.txt` ***b)*** `u0/entree-1.b.txt`

Dans l'instance de test, les éléments $0, 1$ et 5 sont crées par le Big Bang, et seules les recettes 16 et 21 sont actives. La réponse est donc $16 + 21 = 37$.

Question à développer pendant l'oral 1 *Décrivez votre algorithme et les structures de données que vous avez choisi.*

2.2 Le monde point fixe

Quand une recette r est déclenchée, certaines recettes peuvent devenir inactives, c'est le cas des recettes produisant le même élément que r. De même certaines recettes peuvent devenir actives, si leur dernier ingrédient manquant est produit par r. Considérez le processus de déclenchement de toutes les recettes actives dans un ordre arbitraire, jusqu'à ce que toutes les recettes soient inactives. La dynamique termine en un point fixe, qui ne dépend pas de l'ordre de déclenchement des recettes. Nous appelons monde point fixe l'ensemble des éléments actifs à ce moment là.

Question 2 *Implémentez un programme en C qui identifie les éléments composant le monde point fixe, et calcule la somme de leurs numéros.*

a) `u0/entree-2.a.txt` ***b)*** `u0/entree-2.b.txt`

Sur l'instance de test, le monde point fixe est composé des éléments 0 à 10 ainsi que 13, 14, 17, 18, 21, 22, 23 et 25. La réponse est leur somme, à savoir 208.

Question à développer pendant l'oral 2 *Pourquoi est-ce que le monde point fixe ne dépend pas de l'ordre de déclenchement des recettes ? Quelle est la complexité de votre algorithme en n, m, D, sans compter la partie de lecture de fichier ? Un algorithme de complexité optimisée n'est pas nécessaire pour la taille des instances données, mais quel pourrait être un tel algorithme ?*

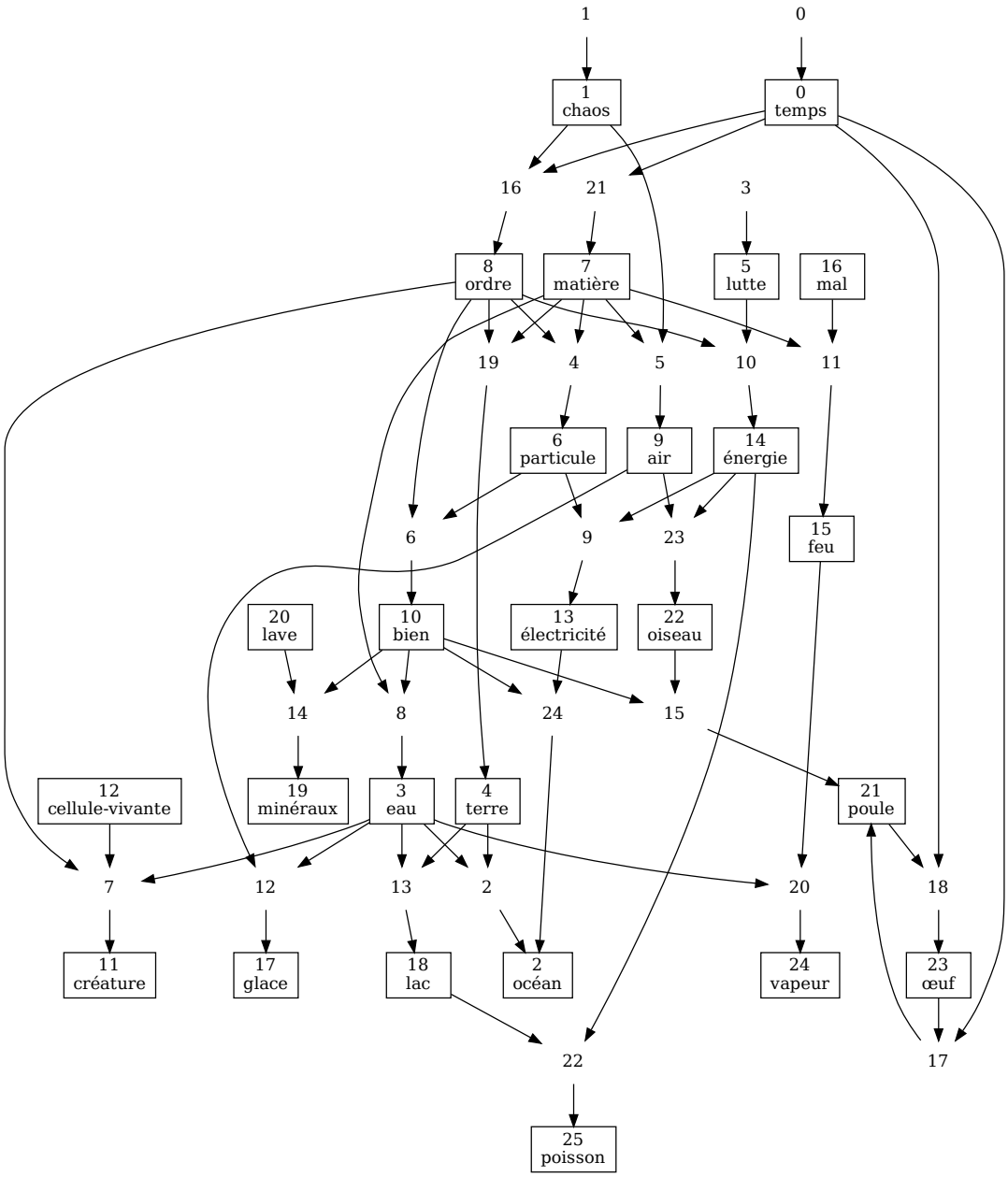

FIGURE 1 – L'instance de test pour $\widetilde{u_0}$ et les sous-questions 'a'. Les éléments sont illustrés par des sommets rectangulaires et étiquetés par leur indice et nom, alors que les recettes sont illustrées par des sommets sans entourage et étiquetés par leur indice. Les arcs entrants indiquent les ingrédients, et l'unique arc sortant le produit de la recette. Les noms des éléments sont données uniquement pour l'illustration et ne font pas partie de l'instance.

3 Partie en OCaml : production

3.1 Dépendances

Les recettes induisent une notion de dépendance entre les éléments. On dit que l'élément t dépend de l'élément s si il existe une séquence non vide de recettes $r_1, \ldots, r_k$ tel que :

- s est un ingrédient de la recette r_1,
- pour tout $i = 1, \ldots, k-1$, le produit de la recette r_i est un ingrédient de la recette r_{i+1},
- et t est le produit de la recette r_k.

Une telle séquence est la preuve que t dépend de s. Notez que la définition exclut de la séquence les recettes sans ingrédient.
Par exemple dans l'instance de test la séquence de recettes 16, 6, 8, 13, 22 prouve que l'élément 25 dépend de l'élément 0. Par contre 0 ne dépend pas de 25, d'ailleurs aucun élément ne dépend de 25. Curiosité, l'élément 23 dépend de lui-même par les recettes 17, 18.

Question 3 *Implémentez un programme en OCaml qui compte le nombre de paires d'éléments (s, t) telles que t dépend de s.*

a) `u0/entree-3.a.txt` ***b)*** `u0/entree-3.b.txt`

Question à développer pendant l'oral 3 *Quelle est la complexité de votre algorithme en n, m, D ? Pour (s, t) tel que t dépend de s, comment pourrait-on trouver une séquence de recettes prouvant la dépendance, qui soit de longueur minimale ?*

3.2 Transformer du plomb en or

Une question ouverte en alchimie était de trouver la manière de créer de l'or à partir de plomb. Pour cela, il faudrait sûrement d'autres ingrédients, et une séquence de recettes particulières.
Formellement, un processus de création d'un élément t à partir d'un élément s consiste en un ensemble initial S_0 et d'une séquence de recettes $R = (r_1, \ldots, r_k)$ satisfaisant les propriétés suivantes, où on note p_i l'élément produit par la recette r_i et I_i l'ensemble de ses ingrédients :

- $s \in S_0$
- $t = p_k$
- pour tout i, après avoir déclenché les recettes $r_1, \ldots, r_{i-1}$, la recette r_i est active (en particulier, r_1 est active au début) ; formellement, on demande que l'ensemble I_i soit inclus dans $S_0 \cup \{p_1, \ldots, p_{i-1}\}$ et que p_i n'appartienne pas à $S_0 \cup \{p_1, \ldots, p_{i-1}\}$.
- l'ensemble S_0 est minimal par inclusion, dans le sens où chaque ingrédient de S_0 est nécessaire pour au moins une recette de cette séquence de recettes.

Par difficulté d'une recette on entend tout simplement son numéro. Par extension, la difficulté d'un processus de création est la difficulté maximale de ses recettes.
S'il existe plusieurs processus de création pour deux éléments s, t donnés, alors on en privilégie un de difficulté minimale. Plus précisément, la difficulté pour créer t à partir de s est la difficulté minimale sur tous les processus de création de t à partir de s. S'il est impossible de créer t à partir de s, alors cette difficulté est -1 par définition.
À titre d'exemple voici deux processus de création pour $s = 0, t = 2$ dans l'instance de test :

- $S_0 = \{8, 10, 14\}$, $R = (21, 4, 9, 24)$, difficulté 24
- $S_0 = \{1, 4, 6, 7\}$, $R = (16, 6, 8, 2)$, difficulté 16

D'autres processus de création existent, mais ils ont tous une difficulté d'au moins 16.

Question 4 *Implémentez un programme en OCaml qui, pour deux éléments s, t donnés, calcule la difficulté de créer t à partir de s.*

a) *s=0, t=2,* `u0/entree-4.a.txt` ***b)*** *s=0, t=3,* `u0/entree-4.b.txt`

c) *s=1, t=2,* `u0/entree-4.c.txt` ***d)*** *s=1, t=3,* `u0/entree-4.d.txt`

Question à développer pendant l'oral 4 *Quelle est la complexité de votre algorithme en n, m, D, k ? Comment est-ce qu'il pourrait servir à calculer la difficulté de créer non pas un élément cible t, mais un ensemble T d'éléments cible ? (Il faut alors que T soit inclus dans $S_0 \cup \{p_1, \ldots, p_k\}$, mais pas dans $S_0 \cup \{p_1, \ldots, p_{k-1}\}$.)*

3.3 Produire à coût minimal

Cette question ajoute la notion de coût à la question 2.2. Nous disons qu'une recette est potentiellement active, si tous ses ingrédients appartiennent au monde point fixe.
Désormais les produits du monde point fixe ont un coût, ainsi que les recettes potentiellement actives.
Le coût d'une recette potentiellement active est 1 € plus la somme des coûts de ses ingrédients, avec un plafond à 10 000 € : si la somme excède ce plafond, alors le coût est ramené à 10 000 €. En particulier, les recettes sans ingrédient coûtent 1 €.
Le coût d'un élément du monde point fixe est le coût minimum sur toutes les recettes potentiellement actives qui ont pour produit cet élément.
Notez que les dépendances circulaires ajoutent une difficulté intéressante à ces définitions de coût.
Ainsi, pour l'instance de test, les éléments 0, 1, 5 coûtent chacun 1 €, l'élément 7 coûte 2 €, l'élément 8 coûte 3 €, l'élément 9 coûte 4 € et ainsi de suite. La somme des coûts des éléments du monde point fixe est 202 € (dans la fiche réponse type, on a noté simplement 202 sans unité).

Question 5 *Écrivez un programme en OCaml, qui calcule la somme des coûts de tous les éléments du monde point fixe.*

a) `u0/entree-5.a.txt` ***b)*** `u0/entree-5.b.txt`

Question à développer pendant l'oral 5 *Quelle est la complexité de votre algorithme en n, m et D ? Imaginons que le coût d'une recette r potentiellement active soit désormais c_r plus le coût total de ses ingrédients pour des coûts entiers strictement positifs $c_0, \ldots, c_{m-1}$ donnés ; comment pourrait-on adapter votre algorithme à cette généralisation ?*

4 Partie en C : optimiser

4.1 Un monde meilleur

Qui n'a pas rêvé d'un monde sans guerre ni chewing-gum collé sous les tables ? Si seulement il était possible de supprimer certains éléments pour atteindre cet état de grâce.
Concrètement nous cherchons à éliminer des éléments au moment du Big Bang.
Soit G l'ensemble des éléments produits par des recettes de degré zéro. Pour tout ensemble $S \subseteq G$, soit $F(S)$ le monde point fixe obtenu à partir de S en appliquant de manière répétée

toutes les recettes actives qui ont un degré non nul. En particulier $F(G)$ est le monde point fixe que vous avez calculé en question 2.2, et $F(\varnothing) = \varnothing$.
Étant donné un élément i, on cherche un sous-ensemble $S \subseteq G$, tel que $i \notin F(S)$. Pour rendre la solution unique, nous introduisons un ordre sur les ensembles.
Soient deux ensembles $S \subseteq G$ et $T \subseteq G$. On dit que S est lexicographiquement plus grand que T, noté $S \succ T$, lorsqu'une des conditions suivantes est satisfaite.

- Si S n'est pas est vide, mais T l'est, alors $S \succ T$.
- Si S n'est pas vide ni T, soient $s = \min(S)$ et $t = \min(T)$. Si $s < t$, alors $S \succ T$.
- Par contre si $s = t$, alors $S \succ T$ si et seulement si $S \setminus \{s\} \succ T \setminus \{t\}$.

Par exemple :

$$\{1\} \succ \{2, 3, 4\}$$
$$\{1, 2, 3\} \succ \{1, 2\}$$
$$\{1, 2, 3\} \prec \{1, 3\}$$

Pour un élément i, nous voulons calculer l'ensemble lexicographiquement maximal $A_i \subseteq G$ tel que $i \notin F(A_i)$. Nous notons a_i l'entier défini par $a_i = \sum_{j \in A_i} j$.
Notez que si i n'appartient pas à $F(G)$, alors $A_i = G$.
À titre d'exemple pour l'instance de test, nous avons

- $F(\varnothing) = \varnothing$
- $F(\{5\}) = \{5\}$
- $F(\{1\}) = \{1\}$
- $F(\{1, 5\}) = \{1, 5\}$
- $F(\{0\}) = \{0, 7\}$
- $F(\{0, 5\}) = \{0, 5, 7\}$
- $F(\{0, 1\}) = \{0, 1, 2, 3, 4, 6, 8, 9, 10, 17, 18\}$
- $F(\{0, 1, 5\}) = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 14, 17, 18, 21, 22, 23, 25\}$

L'ensemble lexicographiquement maximal $S \subseteq \{0, 1, 5\}$ tel que $2 \notin F(S)$ est $S = \{0, 5\}$. Nous avons donc $a_2 = 0 + 5$.

Question 6 *Implémentez en C un programme qui calcule a_i pour un élément i donné.*

a) a_2, `u0/entree-6.a.txt` ***b)*** a_3, `u0/entree-6.b.txt`

c) a_4, `u0/entree-6.c.txt` ***d)*** a_5, `u0/entree-6.d.txt`

Question à développer pendant l'oral 6 *Expliquer votre algorithme. Quelle est sa complexité en $n, m, D, |G|$?*

4.2 Production unitaire

Désormais on considère le temps dans le processus de création du monde point fixe. Au temps 0 il y a le monde vide. Au temps 1 arrive le Big Bang, qui crée tous les éléments qui peuvent être produits par des recettes sans ingrédient. Puis pour tout instant $t = 2, 3, \ldots$, sont déclenchées de manière simultanée toutes les recettes actives au temps $t - 1$. La date de naissance d'un élément du monde point fixe est le temps où l'élément sera produit.
À titre d'exemple sur l'instance de test, les dates de naissance des éléments du monde point fixe sont les suivants. Leur somme est 71.

i	0	1	2	3	4	5	6	7	8	9	10	13	14	17	18	21	22	23	25
naissance de i	1	1	5	5	3	1	3	2	2	3	4	4	3	6	6	5	4	6	7

Question 7 *Implémentez en C un programme qui calcule la somme des dates de naissance de tous les éléments du monde point fixe.*

a) `u0/entree-7.a.txt` ***b)*** `u0/entree-7.b.txt`

Question à développer pendant l'oral 7 *Pourquoi est-ce que votre algorithme est correct ? Quelle est sa complexité en n, m, D ?*

4.3 Production lente

Désormais les recettes prennent un certain temps pour être déclenchées, une seule unité de temps pour les recettes avec un seul ou aucun ingrédient, et deux unités de temps pour les recettes à plusieurs ingrédients. Formellement, dès qu'une recette devient active, disons au temps t, elle est alors immédiatement déclenchée et produira son élément au temps $t + \max\{1, \min\{2, d\}\}$, où d est le degré de la recette. Évidement il se pourra qu'à ce moment là, l'élément ait déjà été produit préalablement par une autre recette.
Par exemple sur l'instance de test, les dates de naissance des éléments du monde point fixe sont les suivants. Leur somme est 121.

i	0	1	2	3	4	5	6	7	8	9	10	13	14	17	18	21	22	23	25
naissance de i	1	1	9	9	5	1	5	2	3	4	7	7	5	11	11	9	7	11	13

Question 8 *Implémentez en C un programme qui calcule la somme des dates de naissance de tous les éléments du monde point fixe.*

a) `u0/entree-8.a.txt` ***b)*** `u0/entree-8.b.txt`

Question à développer pendant l'oral 8 *Pourquoi est-ce que votre algorithme est correct ? Quelle est sa complexité en n, m, D ? Quelle structure de données avez vous utilisé ? Comment généraliser l'algorithme pour permettre des durées des recettes arbitraires données ?*

Fiche réponse type : Alchimie

$\widetilde{u_0}$: 0

Question 1

a) 37

b) 104 924

Question 2

a) 208

b) 472 790

Question 3

a) 132

b) 483 222

Question 4

a) 16

b) 1 422

c) 3 010

d) 3 010

Question 5

a) 202

b) 222 925

Question 6

a) 5

b) 9 967

c) 23 358

d) 23 358

Question 7

a) 71

b) 12 403

Question 8

a) 121

b) 14 343

Fiche réponse : Alchimie

Nom, prénom, u_0 : .....

Question 1

a)

b)

Question 2

a)

b)

Question 3

a)

b)

Question 4

a)

b)

c)

d)

Question 5

a)

b)

Question 6

a)

b)

c)

d)

Question 7

a)

b)

Question 8

a)

b)