# COMPOSITION D'INFORMATIQUE B (XELC), FILIÈRE PC

## 1 Remarques générales

Le sujet était découpé en 4 parties, dont une partie d'énumération récursive de chemins dans un graphe, et une partie de codage et de programmation dynamique. L'énumération (une question semblable était présente dans le sujet de l'année dernière) a posé de grandes difficultés aux candidats. Les questions de codage, souvent simples, ont eu du succès mais la lecture de code de programmation dynamique ainsi que la question finale d'utilisation de l'algorithme ont été très peu réussies.

Les consignes de l'épreuve ont été parfois négligées : les opérations autorisées dans l'épreuve étaient indiquées en début de document, en particulier la syntaxe pour la copie de liste en passage par valeur ou le parcours de dictionnaire. Dans ce dernier cas, une erreur de syntaxe s'est glissée dans le sujet mais n'a pas été pénalisée.

Peu de questions de complexité cette année. Hormis la première très largement traitée avec succès, elles ont été peu traitées.

Un manque d'attention au typage des fonctions (en entrée ou en sortie) génère des réponses hors-sujet et des erreurs largement évitables. En particulier, la liste `sauts` contenait des couples et le dictionnaire `bonus` avait pour clés des couples et pour valeurs des listes.

Les problèmes de compréhension des boucles de type `for` et `while` sont moins présents que ces dernières années, mais certaines confusions existent. Par exemple, certains candidats confondent

- `for i in l:`,
- `for i in range(l):` et
- `for i in range(len(l)):`.

Quelques erreurs fréquentes à éviter :

1. `[1,2].append([3,4])` produit `[1,2,[3,4]]` et non `[1,2,3,4]`
2. `(1,2)+(3,4)` produit `(1,2,3,4)` et non `(4,6)`
3. `if b==True:` ou `if b==False:` au lieu de `if b:` ou `if not b:` sont corrects mais à éviter.
4. De même, éviter les structures comme :

```
n=0
if b:
  n=n+1
if n==1:
  return True
```

## 2 Partie I

### Question 1

Une première question simple toujours traitée avec très peu d'erreurs (y compris pour la complexité), essentiellement dues à une incompréhension de la question ou à de l'inattention.

### Question 2

Question plutôt réussie. Il était attendu de ne pas modifier les entrées `i` et `j`.

### Question 3

Encore une question très accessible et bien réussie dans l'ensemble. Cependant beaucoup de candidats n'ont pas eu l'ensemble des points :

- soit pour avoir modifié l'entrée `sauts`.
- soit pour avoir utilisé `bonus` comme un dictionnaire dont les valeurs sont des couples et non des listes.
- soit pour une erreur algorithmique comme l'ajout des sauts bonus à l'ensemble de sauts possibles au mauvais moment (trop tard en général).

### Question 4

Dernière question de cette première partie accessible et là aussi réussie. Un grand nombre de candidats a perdu des points en ne prêtant pas assez attention au typage de bonus (c'est un dictionnaire et pas une liste donc `for x in bonus` parcoure les clés et les valeurs sont des listes de couples et pas des couples). La négation de la formule logique n'a pas toujours été correcte non plus.

## 3 Partie II

### Question 5

Début des questions moins réussies. Malgré la question 4 qui demandait d'étudier la condition d'acceptabilité des sauts, la grand majorité des étudiants a restreint l'ensemble des sauts possibles ou mal interprété la condition. En particulier, beaucoup ont "oublié" qu'un saut pouvait avoir une coordonnée négative ou de valeur absolue différente de 0 ou 1. Et extrêmement peu d'étudiants ont pensé à proposer un saut dépendant de la taille du plateau.

### Question 6

Plusieurs difficultés algorithmiques et techniques ici autour de la recherche récursive exhaustive de chemins. À la fois le principe de descente récursive dans le DAG et la mise en oeuvre ont permis de distinguer les meilleures copies. En particulier :

- faire le calcul en remontant de la fin des chemins;
- construire les solutions optimales le long de chaque chemin;
- utiliser correctement chaque variable (créer une copie plutôt que modifier les variables qui seront réutilisées, utiliser les primitives adaptées aux types donnés).

La question de complexité a été très peu traitée.

### Question 7

Le mot et l'idée générale de "mémoïsation" sont bien intégrés. Les candidats n'ont en revanche presque jamais pensé à détailler en donnant la liste des paramètres, il fallait penser à ajouter `sauts` et `sauts_max`.

# 4 Partie III

### Question 8

Beaucoup d'erreurs dans l'application de l'algorithme. Il semble que les candidats suivent souvent une intuition plutôt que d'appliquer les étapes de manière systématique.

Sur la question ouverte, il fallait évidemment donner une justification (un contre-exemple en l'occurrence).

### Question 9

Nouvelle question algorithmiquement plus riche. Comme pour la question 6, beaucoup de difficultés pour bien comprendre la question et avoir les bonnes idées algorithmiques. La mise en oeuvre technique s'est révélée particulièrement compliquée pour les candidats.

# 5 Partie IV

### Question 10

Question très accessible, souvent correctement traitée. Des erreurs techniques regrettables tout de même :

- calcul du nombre de 1 dans une suite de 0 et de 1, au lieu du calcul du nombre représenté en binaire;
- calcul en base 10 au lieu de 2.

### Question 11

Très peu de candidats ont eu tous les points, il manquait des détails, en particulier le test de validité de la case `(i_s,j_s)`.

### Question 12

Peu de tentatives de réponses sur cette question de génération conditionnelle de sous-ensembles, l'algorithme était présenté dans le sujet. Une proportion non négligeable d'entre elles était correcte ou approchait une solution correcte.

### Question 13

Le code demandé dans cette question était plutôt simple, la difficulté venait de la multiplicité des objets à manipuler dans cette fin de sujet et de la compréhension de l'objectif poursuivi. Très peu de tentatives de réponses à nouveau, mais souvent bonnes.

### Question 14

Extrêmement peu de points attribués sur cette question de code à trous. Les quelques candidats ayant compris l'algorithme implémenté et pris le temps de travailler cette question ont souvent gagné des points.

### Question 15

Dernière question de nouveau très peu traitée. Il s'agissait de parcourir un tableau de solutions donné par l'algorithme de programmation dynamique de la question précédente. Pas de difficulté dans le parcours, mais il fallait là encore avoir bien compris la démarche. D'autant que contrairement au cas le plus fréquent en programmation dynamique, le calcul du tableau se faisait depuis `(i,j)=(N-1,N-1)`, donc le parcours du tableau se fait depuis `(i,j)=(0,0)`.