EPU GB 4ème année BIMB — *Algorithmique pour la biologie*
Examen du 23 avril 2024 — Documents autorisés.

Inscrivez **lisiblement** vos Nom et Prénom en tête de vos copies.

Exercice 1 : (**Complexité** – [**3 points**])
Quelle est la complexité temporelle (en notation $O(.)$) de ces quatre algorithmes ? Justifiez vos quatre réponses.

```
Algorithm complexity(n)
  t=0
  for i in range(int(n*log(n))):
     t=t+i
  return t
```

1a) $O(log(n))$ 1b) $O(n)$ 1c) $O(n \times log(n))$
1d) $O(n^2)$ 1e) $O(n^3)$ 1f) $O(2^n)$

```
Algorithm GoodMorning(n)
  t=0
  for i in range (n):
     for j in range(n,0,-1):
         t = t + i + j
  return t
```

2a) $O(log(n))$ 2b) $O(n)$ 2c) $O(n \times log(n))$
2d) $O(n^2)$ 2e) $O(n^3)$ 2f) $O(2^n)$

```
Algorithm whoAmI(n)
  t=0
  i=1
  while log2(i) < n :
     t = t + 1
     i = i + 1
  return t
```

3a) $O(log(n))$ 3b) $O(n)$ 3c) $O(n \times log(n))$
3d) $O(n^2)$ 3e) $O(n^3)$ 3f) $O(2^n)$

```
Algorithm GoodLuck(n)
  i = 1; j = 1;
  while i <= n :
     j = j + 3;
     i = i * 2;
  return j-i
```

4a) $O(log(n))$ 4b) $O(n)$ 4c) $O(n \times log(n))$
4d) $O(n^2)$ 4e) $O(n^3)$ 4f) $O(2^n)$

Exercice 2 : (**Jobs d'étudiants** [**17 points**])

Un étudiant désire travailler en dehors de ses heures de cours pour gagner de l'argent de poche. Afin de ne pas compromettre ses études, il décide de consacrer un maximum de T heures par semaine à ses activités rémunératrices. Après de minutieuses recherches, il a trouvé n emplois possibles. Les salaires qui lui sont offerts ne sont pas proportionnels aux nombres d'heures de travail. Le tableau ci-dessous donne le salaire $s(j, i)$ de l'emploi i si on travaille j heures. T et n sont pris égaux à 4.

	Emploi I	Emploi II	Emploi III	Emploi IV
0h de travail	0	0	0	0
1h de travail	26	23	16	19
2h de travail	39	36	32	36
3h de travail	48	44	48	47
4h de travail	54	49	64	56

On s'intéresse maintenant à optimiser le revenu obtenu pour un nombre d'heures travaillées égal à 4. On considérera que l'étudiant peut travailler dans plusieurs entreprises différentes en parallèle, du moment que le nombre d'heures travaillées ne dépasse pas 4.

1. [**3.5 points**] Définissez la famille de problèmes à laquelle appartient cette recherche de revenu optimal. Donnez les cas de base (cas où on peut donner le résultat en temps constant) ainsi que la récurrence (relation entre différentes solutions associées à des problèmes de la famille).
2. [**2.5 points**] En déduire un algorithme naïf récursif.
3. [**2.5 points**] En déduire un algorithme de programmation dynamique, version mémoïsation (appel récursif, mais en passant en argument un tableau qui permet de ne pas recalculer une valeur déjà calculée).
4. [**2.5 points**] Ecrire enfin la version itérative de l'algorithme de programmation dynamique.
5. [**2 points**] Utilisez cet algorithme sur l'exemple.
6. [**2 points**] Quelles sont la complexité des 2 derniers algorithmes ?
7. [**2 points**] On souhaite, en plus de déterminer la rémunération optimale, savoir combien d'heures l'étudiant doit effectuer dans chaque emploi. Expliquer comment faire, autrement dit, décrire l'algorithme sans pour autant donner le pseudo-code.