Master 1 SVS – BIM *Algorithmique pour la biologie*
Examen du 8 décembre 2016 Documents autorisés.

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

Exercice 1 : (Programmation dynamique – [15 points])
Nous avons à notre disposition un grand nombre N de films sous forme numérique. Les films sont sous des formats différents (avi, mp4, mkv, flv...) avec des qualités différentes, ce qui fait que la taille des fichiers n'est pas représentative de la longueur des films. D'autre part, nous avons 2 clefs USB de taille T_1 et T_2 (exprimées en méga octet) et nous cherchons à stocker un grand nombre de films sur ces clefs de telle sorte que la somme des durées des films stockés sur les clefs soit maximale. Le choix des films à copier sur les clefs n'est pas défini *a priori*. Plusieurs politiques de choix sont donc possibles.

Notations. Pour faciliter l'écriture des algorithmes, nous supposons que les durées (non nulles) des N films sont exprimées en minutes (elles sont donc des entiers) et que les tailles des fichiers sont exprimées en méga-octets. δ_i représente la durée du film i (pour $1 \leq i \leq N$) et θ_i représente la taille du fichier du film i.

1. **[6 points]** Dans un premier temps, nous cherchons à maximiser la durée totale des films enregistrées sur les deux clefs, sachant que l'on interdit de couper un film : un film est copié en entier sur l'une des 2 clefs ou n'est pas copié du tout.
   (a) [3 points] Relation de récurrence. On note $durée(i, t_1, t_2)$ la durée maximale des films que l'on peut copier sur les clefs, sachant que l'on peut mettre au maximum t_1 méga-octets sur la première clef et t_2 sur la deuxième, et sachant que l'on ne copie que des films parmi les i premiers. $durée(N, T_1, T_2)$ nous donnera donc la solution de notre problème.
   Proposez une formule de récurrence définissant $durée(i, t_1, t_2)$.
   Indication : dans une solution optimale, soit le $i^{ème}$ film est copié sur la première clé, soit il est copié sur la deuxième, soit il n'est copié sur aucune des deux clefs.
   (b) [1 point] Quelles sont les conditions initiales de cette relation de récurrences ? (Dans quels cas la solution est évidente sans passer par la relation de récurrence ?)
   (c) [1 point] Proposez un algorithme récursif en python pour résoudre le problème (la récurrence établie en 1.(a) est à utiliser).
   (d) [1 point] Expliquez à l'aide d'un exemple que le même sous problème peut être calculé plusieurs fois lors d'un appel à la fonction précédente.
2. **[4 points]** Résolution par programmation dynamique « pure ».
   Proposez un algorithme basé sur le principe de la programmation dynamique et résolvant notre problème. Cet algorithme utilisera bien évidemment la relation de récurrence précédemment établie.
3. **[4 points]** Algorithme itératif
   (a) [3 points] Proposez un algorithme itératif pour résoudre le problème.
   (b) [1 point] Quelle est la complexité de cet algorithme ?
4. **[1 point]** Recherche des films à copier sur chacune des deux clefs.
   Expliquez, sans donner d'algorithme, quelle serait la stratégie pour donner deux listes de films à copier (une pour chacune des deux clefs) qui mènent à une durée d'enregistrement maximale. On supposera que toutes les solutions intermédiaires sont stockées.

Exercice 2 : (Pattern matching – [5 points])

1. **[2 points]** Construire et dessiner l'automate de recherche de toutes les occurrences du motif `atgggat`.
2. **[3 points]** Supposons maintenant que l'on recherche un motif parmi une famille de motif : {atgggat, atgggtt, atggggt, accgggat, accgggtt, accggggt}. En remarquant la similitude entre les différents motifs de la famille, proposez un automate déterministe qui permette de rechercher toutes les occurences de ces motifs. *Indication :* tous les mots partagent la $1^{ère}$ lettre, les trois `g` d'affilée et la dernière lettre.