Partiel

CPES2: Algorithmique et Applications

Pierre SENELLART

4 novembre 2022

Les seuls documents autorisés sont deux feuilles A4, recto-verso (4 pages) avec le contenu de votre choix. Cet examen dure deux heures, comporte deux exercices et 15 questions et est noté sur 20 points.

A. Buffer circulaire (11 points)

Un buffer circulaire est une structure de données pouvant implémenter une file à deux bouts. Il est constitué d'un tableau (que l'on considérera pour simplifier de taille fixe, la *capacité* du buffer circulaire) auquel on associe une *taille* (le nombre d'éléments effectivement stocké dans le buffer circulaire) et la position du *premier élément* du tableau. L'ensemble des éléments est stocké à la suite du premier élément, dans l'ordre. L'élément de position 0 dans le tableau est considéré comme faisant suite à l'élément de position $(\textit{capacité} - 1)$.

Il est possible dans un buffer circulaire d'ajouter des éléments après le dernier (opération *pushr*), ou avant le premier (opération *pushl*), ou d'enlever des éléments en partant du dernier (opération *popr*) ou en partant du premier (opération *popl*).

Le but de cet exercice est d'implémenter un tel buffer circulaire en Python, pour stocker une file à deux bouts d'entiers.

1. (1 point) On considère la séquence d'opérations suivantes dans un tableau circulaire de capacité 6, initialement vide :

   a) $\textit{pushl}(2)$
   b) $\textit{pushl}(42)$
   c) $\textit{pushr}(7)$
   d) $\textit{pushl}(4)$
   e) $\textit{popr}()$
   f) $\textit{pushr}(12)$
   g) $\textit{pushr}(27)$
   h) $\textit{popl}()$
   i) $\textit{pushr}(14)$
   j) $\textit{pushl}(13)$

   Quel est la liste des éléments de la file à deux bouts implémentée par un tel buffer circulaire après cette séquence d'opération ?

2. (1 point) Définir une classe Python `circ_buffer` et son constructeur prenant en entrée un paramètre `capacity` et créant un buffer circulaire de la capacité donnée. Une instance de cette classe contiendra les propriétés (protégées) suivantes :

   `_array` le tableau de taille fixe ; on utilisera un objet de la classe `numpy.array`, qu'on pourra créer avec `numpy.empty(`n`, dtype=int)` où n est la taille du tableau à créer ;

   `_capacity` la capacité du buffer circulaire ;

   `_start` la position du premier élément (s'il existe) ;

   `_size` le nombre d'éléments effectivement contenu dans le buffer circulaire.

3. (1 point) Définir une méthode protégée `_end` ne prenant pas d'argument (en dehors de `self`) et qui calcule la position du dernier élément du buffer circulaire (s'il existe).
4. (1.5 points) Définir une méthode publique `pushr` prenant en argument un entier `x` et qui insère cet entier après le dernier élément du buffer circulaire. On supposera que cette méthode n'est appelée que si le buffer circulaire n'est pas plein.
5. (1.5 points) Définir une méthode publique `pushl` prenant en argument un entier `x` et qui insère cet entier avant le premier élément du buffer circulaire. On supposera que cette méthode n'est appelée que si le buffer circulaire n'est pas plein.
6. (1.5 points) Définir une méthode publique `popr` ne prenant pas d'argument (en dehors de `self`) et qui supprime le dernier élément du buffer circulaire et le renvoie. On supposera que cette méthode n'est appelée que si le buffer circulaire n'est pas vide.
7. (1.5 points) Définir une méthode publique `popl` ne prenant pas d'argument (en dehors de `self`) et qui supprime le premier élément du buffer circulaire et le renvoie. On supposera que cette méthode n'est appelée que si le buffer circulaire n'est pas vide.
8. (1 point) Définir la méthode magique `__str__` renvoyant une représentation de la liste des entiers contenus dans le buffer circulaire, du premier au dernier, sous la forme d'une chaîne de caractères séparée d'espaces.
9. (1 point) Quelle est la complexité de chacune des méthodes de cette classe ? Justifier.

B. Découper des morceaux de bois (9 points)

Un magasin de bricolage vend des pièces de bois ayant toutes la même largeur et la même épaisseur mais des longueurs variables $l_1, \ldots, l_n$ où n est le nombre de longueurs possibles. Le prix d'une pièce dépend de sa valeur et est fonction de la demande pour ce type précis de pièce. La valeur v_l d'une pièce de longueur l est supposée fixée, et donnée par un tableau comme le suivant :

Longueur (m)	1	2	3	5
Prix (€)	2	3	8	9

La propriétaire du magasin reçoit chaque jour un long morceau de bois, de même épaisseur et largeur que les pièces vendues, qu'elle souhaite débiter en pièces qu'elle peut vendre, en maximisant le prix total qui peut être retiré des pièces découpées. Par exemple, un morceau de 4 m peut être découpé en deux pièces de 2 m, rapportant 2×3 €, mais ce n'est peut-être pas optimal.

10. (1 point) Quelle est la découpe optimale d'un morceau de 4 m en supposant le tableau de prix ci-dessus ? Prouver votre affirmation.

    On suppose pour le reste de l'exercice qu'un tableau de prix $(l_i, v_{l_i})_{1 \leqslant i \leqslant n}$ est fourni, pas nécessairement celui de l'exemple.

11. (2 points) Proposer une formule de récurrence permettant de relier la valeur optimale que l'on peut obtenir en coupant un morceau de bois de longueur L à un sous-problème plus simple. Justifier.
12. (2 points) Proposer (sous forme de pseudo-code) une solution récursive la plus simple à écrire possible pour déterminer la manière optimale de découper un morceau de bois de longueur L. Que pouvez-vous dire de la complexité de cet algorithme ?
13. (2 points) Proposer (sous forme de pseudo-code) une solution plus efficace en utilisant la programmation dynamique. Quelle est la complexité de cet algorithme ?
14. (1 point) Il est aussi possible de résoudre ce problème avec de la mémoïsation. Discuter des avantages et inconvénients de cette solution par rapport à la précédente.
15. (1 point) L'algorithme par programmation dynamique obtenu à la question 13 est-il un algorithme en temps polynomial en la taille des entrées ? Justifier votre réponse.