Devoir libre : Piles, Files

CPGE Oujda

February 5, 2025

Partie 1 : Piles

Exercice 1: Notation polonaise inversée (NPI) et évaluation d'une expression postfixe

Vous recevrez une expression arithmétique sous forme de chaîne de caractères en notation polonaise inversée (postfixe). Par exemple, l'expression suivante :

$$3\ 4\ +\ 2\ *\ 7\ /$$

doit être évaluée pour donner un résultat.

Étapes d'évaluation

L'évaluation de l'expression postfixe se fait de la manière suivante :

- Si l'élément est un **opérande** (un nombre), vous l'empilez dans la pile.
- Si l'élément est un **opérateur** ($+$, $-$, $*$, $/$), vous dépilez les deux derniers éléments de la pile, effectuez l'opération correspondante, puis empilez le résultat.

À la fin de l'évaluation, le seul élément restant dans la pile est le résultat de l'expression.

Exemple d'exécution

Prenons l'expression suivante :

"3 4 + 2 * 7 /"

L'évaluation se fait comme suit :
1. $3\ 4\ + \Rightarrow 7$ 2. $7\ 2\ * \Rightarrow 14$ 3. $14\ 7\ / \Rightarrow 2.0$
Ainsi, le résultat de l'expression est 2.0.

Détails de l'exercice

1. Lire l'expression postfixe en entrée.
2. Utiliser une pile pour évaluer l'expression :
   - Lorsqu'un opérande est rencontré, empilez-le.
   - Lorsqu'un opérateur est rencontré, dépilez deux éléments de la pile, effectuez l'opération, puis empilez le résultat.

3. À la fin de l'évaluation, le résultat final sera l'élément restant dans la pile.
4. Affichez ce résultat à l'écran.

Entrée et Sortie

Entrée : Une expression postfixe, par exemple :

"3 4 + 2 * 7 /"

Sortie : Le résultat de l'évaluation de l'expression. Par exemple :

2.0

Bonus

- Ajoutez une gestion d'erreur pour les cas où l'expression est invalide, par exemple si le nombre d'opérandes et d'opérateurs est incohérent.
- Implémentez des opérateurs supplémentaires comme (puissance) ou % (modulo).

Exercice 2 : Conversion infixe → postfixe (4 pts)

Écris une fonction `infixe_vers_postfixe(expression: str) -> str` pour convertir une expression mathématique infixe en notation postfixée en utilisant une pile.

- Utilise une **pile** pour gérer les opérateurs `+, -, *, /`.
- Respecte la **priorité des opérateurs** : `*` et `/` sont plus prioritaires que `+` et `-`.
- Traite les parenthèses correctement : `(` indique une sous-expression à évaluer en premier.

Exemple :

```
infixe_vers_postfixe("3 + 5 * (2 - 8)")  # Résultat : "3 5 2 8 - * +"
```

Partie 2 : Algorithmes avancés

Exercice 3 : Plus grand rectangle dans un histogramme (5 pts)

On vous donne un histogramme sous forme d'un tableau où chaque élément représente la hauteur d'une barre.

- Implémente une fonction `max_rectangle(histogramme: List[int]) -> int` qui retourne l'aire du plus grand rectangle possible dans cet histogramme.
- Utilise une **pile** pour stocker les indices des barres et déterminer efficacement la plus grande aire possible en $O(n)$.

Exemple :

```
max_rectangle([2,1,5,6,2,3])  # Résultat : 10
```

Exercice 4 : Algorithme du problème de l'eau piégée (Trapping Rain Water) (5 pts)

Introduction

Le problème de l'eau piégée, connu sous le nom de "Trapping Rain Water" en anglais, est un problème classique en algorithmique qui consiste à calculer la quantité d'eau qui peut être piégée entre des barres de hauteurs différentes après une pluie. Ce problème est souvent utilisé pour illustrer des concepts liés aux tableaux, aux pointeurs et à l'optimisation de l'espace et du temps.

Description du problème

On vous donne un tableau d'entiers non négatifs représentant les hauteurs de barres verticales alignées sur un axe horizontal. Chaque barre a une largeur de 1 unité. L'objectif est de déterminer combien d'unités d'eau peuvent être piégées entre ces barres après une pluie. L'eau est piégée dans les creux formés par les barres, et la quantité d'eau à un emplacement donné dépend des hauteurs des barres environnantes.

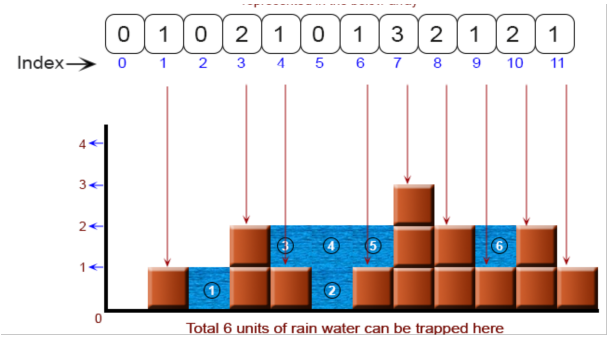

Figure 1: Exemple 1

On vous donne un tableau représentant des hauteurs de bâtiments, où chaque élément est la hauteur d'un bloc. L'eau est piégée dans les creux entre les bâtiments.

- Implémente une fonction `eau_piegee(hauteurs: List[int]) -> int` qui retourne la quantité d'eau piégée.
- Utilise une **pile** pour stocker les indices des bâtiments.

Exemple 1:

```
eau_piegee([0,1,0,2,1,0,1,3,2,1,2,1])  # Résultat : 6
```

Exemple 2:

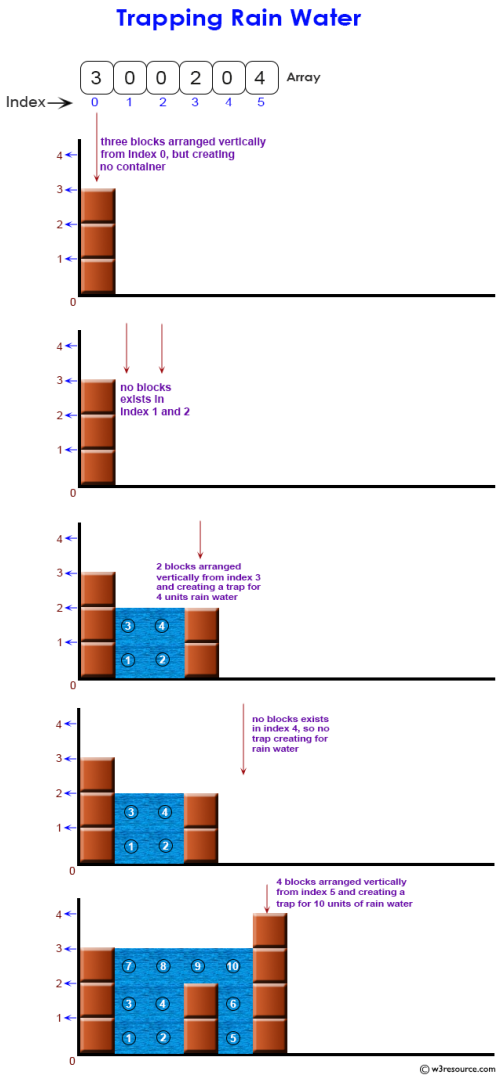

Figure 2: Exemple 2

Barème

- Partie 1 (Piles) : 10 pts
- Partie 2 (Algorithmes avancés) : 10 pts

Total : 20 points