STRUCTURES DE DONNÉES ET ALGORITHMES · CHAPITRE 5

Retirer le dernier élément ajouté

Appliquez le contrat LIFO, tracez empilement et dépilement et protégez le cas vide sans imposer de programmation orientée objet.

Par Ali ZAINOUL · Cours gratuit

Ce que vous allez apprendre

- Appliquer le contrat LIFO et protéger le dépilement à vide.
- LIFO retire en premier le dernier élément ajouté.
- Le sommet définit la partie active du stockage ; une ancienne valeur au-delà du sommet n'appartient plus à la pile.

Prérequis. Avoir étudié les chapitres DSA précédents ; savoir accéder à une collection et suivre des indices.

Partir d'un besoin

Pour annuler des actions, quelle action doit-on retirer d'abord ? La plus récente. Une pile exprime cette règle, indépendamment de son implantation.

Définir la structure et les conventions

Une **pile** suit la règle **LIFO**, Last In, First Out : dernier entré, premier sorti. Son **sommet** est l'extrémité où l'on ajoute et retire. Empiler ajoute au sommet ; dépiler retire et renvoie le sommet ; consulter le sommet ne retire rien.

Nous représentons la pile Python par une liste, avec le sommet à droite. `append` empile ; `pop()` dépile. MATLAB utilise un tableau de capacité fixée et une variable top qui compte les cases occupées ; top vaut 0 quand la pile est vide. On ne lit jamais une case située au-delà de

top.

Le contrat doit préciser les échecs. Ici, retirer d'une pile vide est une erreur. Le tableau MATLAB possède deux places ; ajouter au-delà de cette capacité demanderait soit de signaler une erreur, soit d'agrandir la représentation. L'exemple réalise seulement deux ajouts, puis un retrait.

Suivre les états

Trace LIFO

Opération	Pile du bas vers le sommet	Valeur retirée
Empiler 10	[10]	Aucune
Empiler 20	[10, 20]	Aucune
Dépiler	[10]	20

Comparer les réalisations

Le retrait donne 20 ; 10 reste dans la pile. Dans le tableau MATLAB, une ancienne valeur peut encore occuper une case au-delà de top. Elle n'appartient plus à la pile logique : le compteur définit la partie utilisée.

Pseudo-code

```
// example.pseudo
// use_stack
// Stack operations: PUSH adds, POP removes and returns the top.
// Empty POP is an error.
// PUSH 10; PUSH 20; POP returns 20, leaving 10.
```

Python

```python
# use_stack.py
stack = []
stack.append(10)
stack.append(20)
if len(stack) == 0:
    raise IndexError("Empty stack")
removed = stack.pop()
print(removed, stack)
```

MATLAB

```matlab
% use_stack.m
storage = zeros(1, 2);
top = 0;
top = top + 1;
storage(top) = 10;
top = top + 1;
storage(top) = 20;
if top == 0
    error('Empty stack');
end
removed = storage(top);
top = top - 1;
disp(removed);
disp(storage(1:top));
```

Justifier et compter

Les ajouts et retraits en fin de liste Python ont un coût O(1) amorti ; une réallocation peut coûter davantage lors d'un ajout. Les opérations du tableau préalloué MATLAB sont Θ(1) tant que la capacité suffit. Stocker n valeurs utilise Θ(n) mémoire.

Prévoir puis expérimenter

Prédisez les deux retraits après avoir empilé 10 puis 20 : 20, puis 10. Un troisième retrait doit être signalé comme une erreur. Une fonction de dépilement doit vérifier le vide avant de lire le sommet.

Exercice corrigé

▼ Une pile convient-elle pour servir les clients dans leur ordre d'arrivée ?

Non : le dernier arrivé passerait en premier. Une file réalise l'ordre d'arrivée ; c'est le chapitre suivant.

À retenir

- LIFO retire en premier le dernier élément ajouté.
- Le sommet définit la partie active du stockage ; une ancienne valeur au-delà du sommet n'appartient plus à la pile.
- Le vide et le dépassement de capacité font partie du contrat à préciser.

Version web : https://www.clearcode.fr/courses/programming/dsa/linear/stacks