Développement d'Applications Web

PHP: la Récursivité et la Répétition (Boucles)

Présenté par :

Me Derkaoui Orkia

Université Docteur Moulay TAHAR de Saida

derkaouiorkia@gmail.com

Plan

- Les variables
- Les fonctions
- La récursivité
- La répétition: les boucles

Exercice

Comparez entre ces 2 codes sources :

```php
<html><body>
<?php
$str = "Hello World !";

function reverse_i($str)
{
 for($i=1;$i<=strlen($str);$i++)
 echo substr($str, -$i, 1); //affiche un caractère en partant de la fin
 return;
}
reverse_i($str);

?>
</body></html>
```

```php
<html><body>
<?php
$str = "Hello World !";
reverse_r($str);
function reverse_r($str)
{
 if(strlen($str) > 0)
//substr retourne le premier caractère
 reverse_r(substr($str, 1)); //appel récursif
 echo substr($str, 0, 1); //affiche un caractère
 return;
}
?>
</body></html>
```

Réponse

- **Il y’a 2 solutions:**

Laquelle est la plus performante ou la plus efficace?

Réponse

- Pour compare ces 2 programmes ;
- Il faut d'abord qu'il soient tous les 2 fonctionnels et donne les résultat attendus.
- On peut alors les comparer pour utiliser le plus performant et le plus efficace.

Réponse

- Pour juger les performance d'un programme, on s'intéresse à la quantité de ressources que le programme demande pour son exécution,
- Ces ressources sont :
  - Le temps et
  - L'espace

Réponse

- **Pour répondre à la questions, il faut calculer la complexité de chaque programme ou algorithme.**
- **Complexité spatiale**
- **Complexité en temps ou temporelle**

Complexité asymptotique ; indépendant des performances de l'ordinateur utilisé.

Réponse

Comparez entre ces 2 codes sources :

```
<html><body>
<?php
$str = "Hello World !";

function reverse_i($str)
{
 for($i=1;$i<=strlen($str);$i++)
 echo substr($str, -$i, 1); //affiche un caractère en partant de la fin
 return;
}
reverse_i($str);

?>
</body></html>
```

Complexité

=

O(n)

Réponse

Comparez entre ces 2 codes sources :

```
<html><body>
<?php
$str = "Hello World !";
reverse_r($str);
function reverse_r($str)
{
 if(strlen($str) > 0)
//substr retourne le premier caractère
 reverse_r(substr($str, 1)); //appel récursif
 echo substr($str, 0, 1); //affiche un caractère
 return;
}
?>
</body></html>
```

Complexité

=

O(n)

Calcul de la complexité

Pour voire comment on fait le calcul de la complexité, consulter les deux vidéos suivantes :

1. https://youtu.be/tFGSoK-0l2A
2. https://youtu.be/E4EiCdHIll4

Réponse

La comparaison proposée par l'etudiant :

- Mr Ahmed MOUFFOK
- **2 année LMD**
- **Département d'Informatique**
- **Faculté de Technologie**
- **Université Dr Moulay TAHAR de Saida**

Réponse

COMPARISON :

- Both programs are functions that take a string as input and return the reversed version of the string. However, they use different algorithms to achieve this.
- The first program uses a simple for loop to iterate over the string, and at each iteration, it extracts the last character of the string using the `substr` function and echoes it. The loop runs for the length of the string, so it has a time complexity of O(n), where n is the length of the string.

Réponse

- The second program uses a recursive function to reverse the string. The function first checks if the length of the string is greater than zero. If it is, it recursively calls itself with the substring starting from the second character of the original string. Once the recursive call returns, the function echoes the first character of the original string. This process continues until all characters of the original string have been echoed in reverse order. The time complexity of this algorithm is also O(n), where n is the length of the string.

Réponse

- Both programs have the same time complexity, but the second program has additional space complexity due to its recursive nature. Each recursive call adds a new stack frame to the call stack, which can potentially consume a lot of memory for very large strings. However, for practical purposes, this is unlikely to be a problem unless the string is extremely large or the recursion depth is very high.
- In summary, both programs have the same time complexity of O(n), but the second program has higher space complexity due to its recursive nature.

Cordialy ,

Ahmed MOUFFOK

Réponse

- COMPARAISON :Les deux programmes sont des fonctions qui prennent une chaîne en entrée et renvoient la version inversée de la chaîne. Cependant, ils utilisent différents algorithmes pour y parvenir.Le premier programme utilise une simple boucle for pour itérer sur la chaîne, et à chaque itération, il extrait le dernier caractère de la chaîne en utilisant la fonction `substr` et lui renvoie un écho. La boucle s'exécute sur la longueur de la chaîne, elle a donc une complexité temporelle de O(n), où n est la longueur de la chaîne.

Réponse

- Le deuxième programme utilise une fonction récursive pour inverser la chaîne. La fonction vérifie d'abord si la longueur de la chaîne est supérieure à zéro. Si c'est le cas, il s'appelle récursivement avec la sous-chaîne à partir du deuxième caractère de la chaîne d'origine. Une fois l'appel récursif renvoyé, la fonction renvoie le premier caractère de la chaîne d'origine. Ce processus se poursuit jusqu'à ce que tous les caractères de la chaîne d'origine aient été renvoyés en écho dans l'ordre inverse. La complexité temporelle de cet algorithme est également O(n), où n est la longueur de la chaîne.

Réponse

- Les deux programmes ont la même complexité temporelle, mais le second programme a une complexité spatiale supplémentaire en raison de sa nature récursive. Chaque appel récursif ajoute un nouveau cadre de pile à la pile d'appels, ce qui peut potentiellement consommer beaucoup de mémoire pour les très grandes chaînes. Cependant, pour des raisons pratiques, il est peu probable que cela pose un problème à moins que la chaîne ne soit extrêmement grande ou que la profondeur de récursivité soit très élevée.

Réponse

- En résumé, les deux programmes ont la même complexité temporelle de O(n), mais le second programme a une complexité spatiale plus élevée en raison de sa nature récursive.
- cordialement,
- Ahmed MOUFFOK

Références Bibliographiques

[1] http://icps.u-strasbg.fr/~genaud/courses/webtech/php/ch02.html

[2] http://ivmad.free.fr/pi/PHP/php-1.html

[3] https://www.php.net/manual/fr/indexes.functions.php

[4] https://www.ionos.fr/digitalguide/sites-internet/developpement-web/programmation-imperative/

https://www.invivoo.com/paradigme-de-programmation/

https://slideplayer.fr/slide/2736836/