**Đệ quy là gì?**

Quá trình trong đó một hàm gọi chính nó một cách trực tiếp hoặc gián tiếp được gọi là đệ quy và hàm tương ứng được gọi là hàm đệ quy. Sử dụng thuật toán đệ quy, một số vấn đề có thể được giải quyết khá dễ dàng. Ví dụ về các bài toán như vậy là Tháp Hà Nội (TOH), Trình duyệt cây theo thứ tự/đặt hàng trước/sau theo thứ tự, DFS của đồ thị, v.v. Một hàm đệ quy giải quyết một vấn đề cụ thể bằng cách gọi một bản sao của chính nó và giải các bài toán con nhỏ hơn của các bài toán ban đầu. Nhiều lời gọi đệ quy hơn có thể được tạo ra khi cần thiết. Điều cần thiết là phải biết rằng chúng ta nên cung cấp một trường hợp nhất định để chấm dứt quá trình đệ quy này. Vì vậy, chúng ta có thể nói rằng mỗi khi hàm gọi chính nó với một phiên bản đơn giản hơn của vấn đề ban đầu.

**Sự cần thiết đệ quy**

Đệ quy là một kỹ thuật tuyệt vời với sự trợ giúp của nó, chúng ta có thể giảm độ dài của mã và làm cho nó dễ đọc và viết hơn. Nó có những ưu điểm nhất định so với kỹ thuật lặp sẽ được thảo luận sau. Một nhiệm vụ có thể được xác định với nhiệm vụ con tương tự của nó, đệ quy là một trong những giải pháp tốt nhất cho nó. Ví dụ; Giai thừa của một số.

**Thuộc tính của đệ quy:**

* Thực hiện các thao tác giống nhau nhiều lần với các đầu vào khác nhau.
* Trong mỗi bước, chúng tôi thử đầu vào nhỏ hơn để làm cho vấn đề nhỏ hơn.
* Điều kiện cơ sở là cần thiết để dừng đệ quy nếu không vòng lặp vô hạn sẽ xảy ra.

**Thuật toán: Các bước** Các bước thuật toán để thực hiện đệ quy trong một hàm như sau:

Bước 1 - Xác định trường hợp cơ sở: Xác định trường hợp đơn giản nhất mà giải pháp đã biết hoặc tầm thường. Đây là điều kiện dừng cho đệ quy, vì nó ngăn không cho hàm gọi chính nó vô tận.

Bước 2 - Xác định trường hợp đệ quy: Xác định vấn đề theo các bài toán con nhỏ hơn. Chia vấn đề thành các phiên bản nhỏ hơn của chính nó và gọi hàm theo cách đệ quy để giải quyết từng vấn đề con.

Bước 3 - Đảm bảo kết thúc đệ quy: Đảm bảo rằng hàm đệ quy cuối cùng cũng đạt đến trường hợp cơ sở và không đi vào vòng lặp vô hạn.

Bước 4 - Kết hợp các giải pháp: Kết hợp các giải pháp của các bài toán con để giải bài toán ban đầu.

**Các hàm đệ quy được lưu trữ trong bộ nhớ như thế nào?**

Đệ quy sử dụng nhiều bộ nhớ hơn, bởi vì hàm đệ quy thêm vào ngăn xếp với mỗi lệnh gọi đệ quy và giữ các giá trị ở đó cho đến khi lệnh gọi kết thúc. Hàm đệ quy sử dụng Cấu trúc LIFO (LAST IN FIRST OUT) giống như cấu trúc dữ liệu ngăn xếp.

**Điều kiện cơ bản trong đệ quy là gì?**

Trong chương trình đệ quy, giải pháp cho trường hợp cơ sở được cung cấp và giải pháp cho vấn đề lớn hơn được thể hiện dưới dạng các vấn đề nhỏ hơn.

**Recursion VS Iteration**

|  |  |  |
| --- | --- | --- |
| **SR No.** | **Recursion** | **Iteration** |
| 1) | Terminates when the base case becomes true. | Terminates when the condition becomes false. |
| 2) | Used with functions. | Used with loops. |
| 3) | Every recursive call needs extra space in the stack memory. | Every iteration does not require any extra space. |
| 4) | Smaller code size. | Larger code size. |

**Phân tích độ phức tạp của các thuật toán không hồi quy**

**Hướng chung để phân tích độ phức tạp thời gian**

1. Xác định một hay nhiều tham số biểu thị kích thước của input
2. Xác định basic operation của thuật toán
3. Kiểm tra xem số lần thuật toán thực thi basic operation có chỉ phụ thuộc vào kích thước input hay không. Nếu nó còn phụ thuộc vào một vài yếu tố khác thì ta cần phải khảo sát riêng từng trường hợp xấu nhất, trung bình và trường hợp tốt nhất nếu cần thiết.
4. Thiết lập hệ thức truy hồi, với điều kiện khởi tạo thích hơp cho số lần basic operation được thực thi
5. Giải hệ thức truy hồi hoặc ít nhất xác định được cấp độ tăng của nó