TRẦN THỊ THANH THƯ_DHKL17A2HN_23174600085_CA CHIỀU

Bài 1:

B 1. gọi hàm sum_of_number , hàm đệ quy được gọi với đối số n = 7.

b2. Vì n khác 1, không thỏa mãn điều kiện đầu tiên trong hàm sum_of_number nên chúng ta chuyển sang khối else.

b3. Hàm sẽ trả về n + sum_of_number (n+1) . Với n = 7, điều này trở thành 7 + sum_of_number (6).

b4. Tiếp tục quá trình đệ quy, sum_of_number (6) sẽ trả về 6 + sum_of_number (5).

b5. Quá trình này lặp lại cho đến khi chúng ta đến sum_of_number (1) , khi đó n = 1 và hàm trả về 1.

b6. Sau đó, kết quả của mỗi lần đệ quy được tính toán và cộng dồn lại với nhau: 7 + 6 + 5 + 4 + 3 + 2 1 .

b7. Kết quả cuối cùng là tổng của các số từ 1 đến 7, là 28.

Bài 2:

1. Khi gọi fibonacci(8) , hàm đệ quy được gọi với đối số n = 8.

2. Vì n lớn hơn 1, chúng ta đi vào khối else của hàm.

3. Hàm sẽ trả về tổng của hai lần gọi đệ quy: fibonacci(7) + fibonacci(6) .

4. Tiếp tục quá trình đệ quy, fibonacci(7) sẽ trả về fibonacci(6) + fibonacci(5) .

5. Quá trình này tiếp tục cho đến khi chúng ta đạt được các trường hợp cơ bản (n <= 1) với fibonacci(1) và fibonacci(0) , chúng trả về 1 và 0 tương ứng.

6. Khi đạt được các trường hợp cơ bản, các giá trị này được tính toán và cộng dồn lại từ dưới lên: fibonacci(8) = fibonacci(7) + fibonacci(6) = (fibonacci(6) + fibonacci(5)) + (fibonacci(5) + fibonacci(4)).

7. Quá trình này tiếp tục cho đến khi chúng ta đạt được kết quả cuối cùng cho fibonacci(8)

8. Kết quả cuối cùng là số Fibonacci thứ 8, là 21

Bài 3

1. Ban đầu, hàm power(2, 6) được gọi.

2. Bởi vì n không bằng 0, và n không chẵn, nên chúng ta sẽ vào trường hợp else .

3. Trong trường hợp này, chúng ta nhân x (tức là 2) với kết quả của power(x, n//2) (cụ thể là power(2, 3) * power(2, 3)).

4. Hàm power(2, 3) được gọi hai lần. Một lần cho mỗi lần nhân.

5. Lần này, với n = 3, chúng ta vẫn không ở trường hợp nào là n bằng 0 hoặc n chia hết cho 2, nên chúng ta tiếp tục đệ quy.

6. Chúng ta lại nhân x với kết quả của power(x, n//2) (cụ thể là power(2, 1) * power(2, 1)).

7. Bây giờ, với n = 1, chúng ta vào trường hợp n = 0 và trả về 1.

8. Kết quả là 2 * 2 * 2 * 2 * 2 * 2, tức là 64.

Bài 4:

1. Bước 1: Gọi thap_ha_noi(3, 'A', 'B', 'C') (bài toán con) để chuyển 3 đĩa từ cọc A sang cọc trung gian C, với cọc đích là B.

2. Bước 2: Chuyển đĩa thứ 4 từ cọc A sang cọc B.

3. Bước 3: Gọi thap_ha_noi(3, 'C', 'A', 'B') (bài toán con) để chuyển 3 đĩa từ cọc trung gian C sang cọc B, với cọc ban đầu là A.

4. Bước 4: Lặp lại quy trình cho các bài toán con, tiếp tục giảm số đĩa cho đến khi chỉ còn 1 đĩa, sau đó chuyển trực tiếp từ cọc ban đầu sang cọc đích.

Quá trình này tiếp tục lặp lại cho đến khi tất cả các đĩa được chuyển từ cọc ban đầu sang cọc đích theo quy tắc của bài toán Tháp Hà Nội.

Bài 5:

1. Trong đệ quy, ta cần có một điều kiện để dừng đệ quy. Ở đây, nếu tong_dong_vat (tổng số động vật) bằng 0, nghĩa là không còn động vật nào để xét, ta trả về (0, 0) tức là không có gà và chó.

2. Đầu tiên, chúng ta giảm số lượng động vật đi một đơn vị và giảm tổng số chân đi 2 (vì mỗi con gà có 2 chân).
Sau đó, ta gọi đệ quy cho phần còn lại của bài toán với số động vật và số chân đã giảm.

3. *Kiểm tra số lượng chó có thể có*: Sau khi gọi đệ quy, ta cần kiểm tra xem có thể thêm một con chó vào số lượng đã tính được không.
Điều này kiểm tra bằng cách xem phần dư của tổng số chân sau khi đã trừ đi số chân của tất cả gà. Nếu phần dư chia hết cho 4 (vì mỗi con chó có 4 chân), ta có thể thêm một con chó.

4. **Trả về kết quả**: Cuối cùng, ta trả về số lượng gà và chó đã tính được.

Hàm cho_ga này tiếp tục gọi đệ quy cho đến khi gặp trường hợp cơ bản, sau đó trả về kết quả cuối cùng.