REPORT_WEEK1_24120184

Exercise 1: Fibonacci Series:

Yêu cầu đề bài:

- Viết hàm đệ quy để tính số Fibonacci thứ n. Số Fibonacci thứ n được định nghĩa là số được tính bằng tổng của số Fibonacci thứ n – 1 và số Fibonacci thứ n – 2.
- In ra dãy các số Fibonacci bắt đầu từ F(0), Nhập n với n là số lượng số Fibonacci cần in.

Hướng giải quyết:

- Vì số Fibonacci thứ n bằng tổng của số Fibonacci thứ n – 1 và số Fibonacci thứ n – 2, nên ta có thể sử dụng đệ quy để tính lần lượt số Fibonacci thứ n – 1 và số Fibonacci thứ n – 2, sau đó tính tổng của chúng.
- Nhưng ta cần định nghĩa 2 số đầu của dãy vì 2 số đầu không có các số trước đó để tính được, trường hợp n = 0 thì F(0) = 0, n = 1 thì F(1) = 1.
- Hàm đệ quy:
  -Điều kiện cơ bản – dừng đệ quy: n = 0 trả về 0 , n = 1 trả về 1
  -Khi n > 1: thực hiện đệ quy gọi lại chính nó trả về F(n – 1) + F(n – 2)
- Sử dụng vòng lặp để in ra output thõa mãn.

Độ phức tạp của hàm:

- Vì gọi đệ quy 2 lần , và mỗi lần đệ quy lại gọi thêm 2 lần đệ quy => Tạo ra cây đệ quy nhị phân mà cây đệ quy có số nút tăng theo cấp số nhân (2)

  =>Dẫn đến hàm có độ phức tạp $O(2^n)$.

Exercise 2: Factorial of a Number:

Yêu cầu đề bài:

- Viết hàm đệ quy tính giai thừa của n cho trước.Giai thừa được định nghĩa:

$$n! = (n-1) \times (n-2) \times \dots \times 2 \times 1 \text{ và } 0! = 1$$

Hướng giải quyết:

- Ta thấy rằng n! = n × (n – 1)! , nên ta có thể sử dụng đệ quy để tính n! dựa vào (n – 1)!.
- Định nghĩa 0! = 1.
- Hàm đệ quy:

  -Trường hợp cơ bản-dừng đệ quy: n = 0 trả về 1.

  -Khi n >= 1 thì trả về n × (n – 1)!

Độ phức tạp:

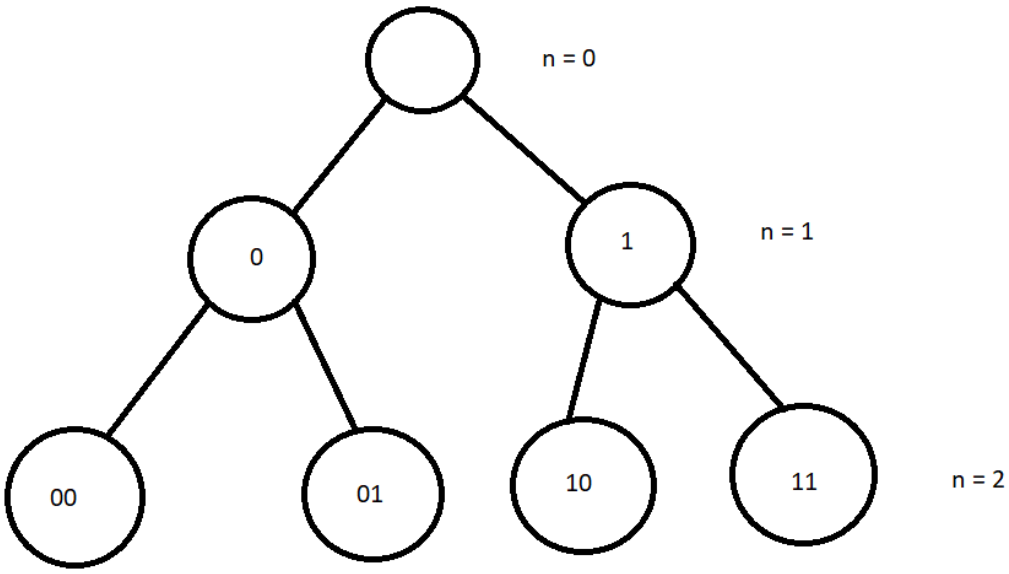
- Mỗi lần gọi hàm chỉ gọi lại 1 lần, hàm chạy từ n đến 0 => O(n).

Exercise 3: Generate All Binary Strings:

Yêu cầu đề bài:

- Viết hàm đệ quy tạo ra tất cả chuỗi nhị phân có độ dài n. Chuỗi nhị phân bao gồm ‘0’ và ‘1’.

Hướng giải quyết:



- Ta thấy với n = 1: bằng cách ta thêm 0 và 1 vào cuối chuỗi rỗng (tức là n = 0)
- Tương tự với n = 2: bằng cách ta thêm 0 và 1 vào cuối chuỗi có độ dài n = 1
- Từ đó ta có thể sử dụng đệ quy để tạo chuỗi bit có độ dài n dựa vào chuỗi bit có độ dài n – 1
- Hàm đệ quy:

  -Điều kiện dừng khi đệ quy đến n = 0 và xuất giá trị của chuỗi.

  Với n > 0:

  -Đệ quy trả về chuỗi có độ dài n – 1 thêm ‘0’ vào cuối chuỗi.

  -Đệ quy trả về chuỗi có độ dài n – 1 thêm ‘1’ vào cuối chuỗi.

Độ phức tạp:

- Hàm tạo chuỗi con gọi đệ quy 2 lần , là 2 hàm con => Tạo ra cây đệ quy nhị phân mà cây đệ quy có số nút tăng theo cấp số nhân (2)

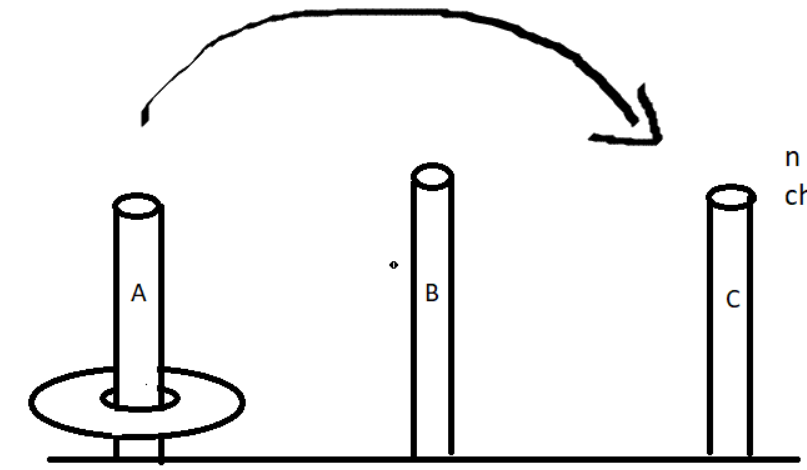
  =>Dẫn đến độ phức tạp của hàm là $O(2^n)$.

Exercise 4: Towers of Hanoi puzzle

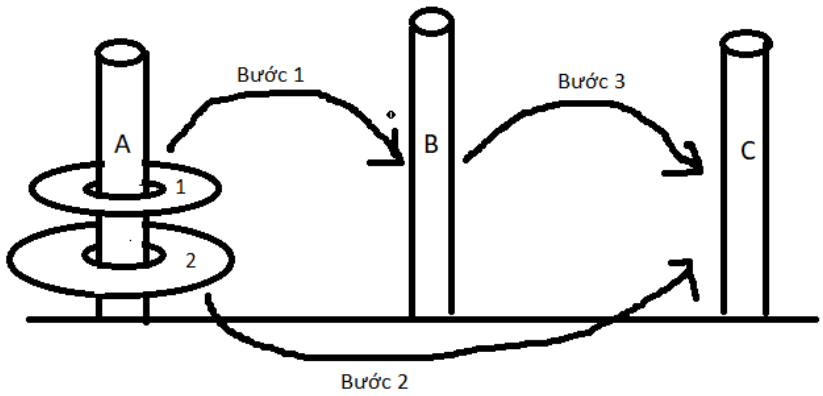
Yêu cầu đề bài:

- Viết hàm đệ quy tính số cách cần để di chuyển n cái đĩa từ cộc A sang cộc C thông qua cộc trung gian B

Hướng giải quyết:



n = 1 ; Có 1 cách chuyển



n = 2;
Đầu tiên di chuyển (n - 1) đĩa tức là 1 đĩa từ cộc A sang B.
Tiếp di chuyển đĩa cuối đĩa (2) từ A sang C
Và di chuyển (n - 1) đĩa từ cộc B sang C.
Số cách:
số cách di chuyển (n - 1) đĩa × 2 + 1.
2 × 1 + 1 = 3 cách.

- Ta sử dụng đệ quy để tính số cách di chuyển n đĩa = số cách di chuyển n đĩa × 2 + 1.
- Hàm đệ quy:

  -Trưởng hợp cơ bản - Dừng đệ quy: n = 1 trả về 1 cách , n = 0 trả về 0 cách.

  -Với n > 1: Đệ quy trả về 2 × số cách di chuyển (n – 1) đĩa + 1.

Độ phức tạp:

- Mỗi lần gọi hàm chỉ gọi lại 1 lần, hàm chạy từ n đến 1 => O(n).

Exercise 5: Given an array , check whether the array is in sorted order with recursion.

Yêu cầu đề bài:

- Cho 1 mảng, kiểm tra xem mảng đã cho đã sắp xếp hay không với đệ quy.

Hướng giải quyết:

- Sử dụng đệ quy kiểm tra xem mảng có n phần tử đã sắp xếp hay không bằng cách kiểm tra mảng có n – 1 phần tử (đồng thời so sánh phần từ thứ n có lớn hơn phần tử thứ n – 1).
- Hàm đệ quy:

  -Điều kiện cơ bản – dừng đệ quy: Còn lại 0 hoặc 1 phần tử trả về true (đã sắp xếp).

  -Nếu phần tử thứ n >= phần tử thứ n – 1 thì trả về true or false của (n – 1) phần tử còn lại.

  -Nếu phần tử thứ n < phần tử thứ n – 1 trả về false.

Độ phức tạp:

- Mỗi lần gọi hàm chỉ gọi lại 1 lần, hàm chạy từ n đến 1 => O(n).

Exercise 6: N-Queens problem:

Yêu cầu đề bài:

- Tìm số cách sắp xếp N quân hậu vào bàn cờ có kích thước N x N đảm bảo rằng mọi quân hậu đều ở vị trí an toàn.
- Nước đi của quân hậu: có thể đi theo cả 2 đường chéo, ngang và dọc.

Hướng giải quyết:

- Vì quân hậu trong ô cờ có thể ăn theo 2 đường chéo, hàng ngang và dọc nên xác định vị trí 1 con hậu an toàn dựa vào các nước mà các quân hậu trước có thể ăn được.
- Vì hậu có thể ăn theo hàng ngang => Đặt N quân hậu theo N hàng(mỗi hàng gồm 1 quân hậu) , tương ứng nếu đặt được quân hậu => di chuyển đến hàng tiếp theo duyệt qua từng cột và kiểm tra quân hậu có an toàn hay không. Nếu duyệt qua tất cả các cột không tìm được vị trí thõa mãn thì quay về bước đặt hậu trước đó đặt chỗ mới và lặp lại cho đến khi tìm được vị trí thích hợp co tất cả quân hậu.
- Cần có hàm kiểm tra sự an toàn của vị trí đặt hậu (đặt theo hàng nên chỉ quan tâm đến trên cột của vị trí đặt đã có quân hậu hay chưa), kiểm tra cả 2 đường chéo của vị trí được xét.
- Hàm đệ quy:

  -Điều kiện dừng đệ quy khi duyệt qua tất cả hàng và đặt hậu thành công (số bước + 1) hoặc duyệt qua tất cả các cột của hàng nhưng không tìm thấy vị trí thõa mãn

  -Duyệt qua từng cột để tìm kiếm vị trí:

-Nếu vị trí an toàn cập nhật vị trí đã đặt quân hậu , đệ quy đi đến hàng tiếp theo tìm kiếm vị trí đã đặt quân hậu mới , nếu kiểm tra không có vị trí phù hợp trả về bước đặt hậu phía trước đó và tìm vị trí mới.

Độ phức tạp:

- Ta thấy ở hàng đầu tiên ta sẽ có N lựa chọn để đặt quân hậu , nhưng sang hàng tiếp theo ta chỉ có N – 1 lựa chọn đặt quân hậu, lặp lại cho đến khi đến hàng thứ N.
  ⇨ $N \times (N - 1) \times (N - 2) \times \ldots \times 2 \times 1 = N!$

Vậy độ phức tạp của thuật toán là O(N!).

Post trên Github: