**Nhóm 3:**

Mai Nhất Tùng - 21520523

Hà Trọng Tài - 21520436

Nguyễn Ngọc Nhung – 21521248

BÀI LÀM

Text

Description automatically generated

Xét bài toán với trên với n cái đĩa. Ta gọi Un là số bước tối thiểu để chuyển n cái đĩa từ cột này sang cột khác. Ta có:

* Để chuyển n đĩa từ cột 1 sang cột 2 ta cần chuyển đĩa lớn nhất sang cột 2 vậy phải chuyển n-1 đĩa trên cùng sang cột 3 tốn ít nhất lần chuyển.
* Tiếp theo chuyển đĩa lớn nhất sang cột 2 tốn 1 lần chuyển.
* Cuối cùng chuyển n-1 đĩa từ cột 3 về cột 2 tốn ít nhất lần chuyển.

Vậy ta có công thức truy hồi như sau = 2 + 1 với = 1, ta có:

+ 1 = 2( + 1) = … = ( + 1) =

=

Vậy để chuyển được 64 đĩa từ cọc nguồn đến cọc dích phải mất ít nhất là – 1 phút tương đương với 35100 tỷ năm.

Mã giả:

|  |
| --- |
| def move(n, source, dest, inter):  if n == 1:  move disk from source to dest  else:  move(n-1, source, inter, dest)  move nth disk from source to dest  move(n-1, inter, dest, source) |

Số lần di chuyển các đĩa.

* Ta nhận thấy theo thuật toán trên thì đĩa lớn nhất chỉ bị di chuyển đúng một lần.
* Dựa vào thuật toán trên ta dễ dàng thấy được move(n-1) luôn được tuy hồi gấp đôi move(n)
* Vậy số lần di chuyển của các đĩa từ lớn tới nhỏ là: 1, 2, 4, 8, …,

Ta có thể sử dụng thuật toán ngắn gọn sau:

|  |
| --- |
| col = ["source", "inter", "dest"]  if n%2==0:  col[1], col[2] = col[2], col[1]  for i in range(1,2\*\*n - 1):  print(f"Move a disk from {col[(i & i-1) % 3]} to {col[(i | i-1) + 1) % 3]}") |

Text

Description automatically generated

Độ phức tạp.

* T(1) = T(0) = 1, T(n) = T(k) + T(n-k-1) + n-1
* Worst case: trường hợp mà dãy số sắp xếp từ lớn tới bé khi đó mỗi lần chia theo pivot không có tác dụng do đó ta có:

T(n) = T(n-1) + T(0) + n-1 = T(n-1) + n = … = 1 + 2 + … + n

* Average case: trường hợp pivot ở vị trí ngẫu nhiên (random pivot)

* (1)

(2)

Lấy (1) – (2), ta được:

⬄

Chia 2 vế cho n\*(n+1)

…

**⬄ T(n)**

* Best case: trường hợp tốt nhất là pivot luôn nằm ở trung vị của dãy. Khi đó ta có:

T(n) = 2T(n/2) + n-1

**T(n) *n*log(n)**

Text

Description automatically generated

* Ý tưởng: Ta có 2n = 2n-1 + 2n-1. Xây dựng hàm đệ quy trả về powerOfTwo(n-1) + powerOfTwo(n-1). Nếu n = 0 thì trả về 1.

Mã giả:

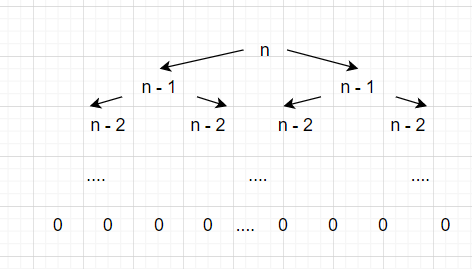
|  |
| --- |
| Function powerOfTwo(int n)  {  if ( n == 0 ) return 1  return powerOfTwo(n-1) + powerOfTwo(n-1)  } |

b)

|  |
| --- |
| T(n) = T(n-1) + T(n-1) + 1 = 2 \* T(n-1) + 1 (với T(0) = 0) |

|  |
| --- |
| T(n) = 2 \* T(n-1) + 1  = 22 \* T(n-2) + 2 + 1  …  = 2n \* T(0) + 2n-1 + … + 1  = 2n - 1 |

c)



* Tổng số lần gọi hàm là 2n + 1, trong đó gọi đệ quy 2n và 1 lần gọi đầu

d)

* Với số lần gọi hàm đệ quy là 2n + 1 và độ phức tạp O(2n). Thuật toán sẽ không phù hợp với giá trị n quá lớn
* Thay vào đó chúng ta có thể thực hiện dời bits cho bài toán tính giá trị cho 2n này, 2n = 0 << (n+1) (dời bit sang trái n+1 đơn vị).