Bài tập về nhà của nhóm 17 - Chủ đề 2: Phân tích độ phức tạp thời gian của đệ quy

Bài toán 1 Với thiết lập của bài toán, ta dễ thấy số ô được loan tới có khoảng cách manhattan với ô bắt đầu luôn **bé hơn hoặc bằng** n.

Gọi $F(n)$ là số ô được loan màu sau n đơn vị thời gian. $F(0) = 1$.

Từ nhận xét 1, $F(n)$ có thể được xây dựng bằng công thức truy hồi sau:

$$F(n) = F(n-1) + 4 * n$$

Do đó ta có thể giải quyết bài toán bằng thuật toán đệ quy với đoạn code python sau:

```python
def F(n):
    # Trường hợp cơ sở
    if n == 0:
        return 1

    # Công thức truy hồi
    return F(n-1) + 4 * n
```

Gọi $T(n)$ là độ phức tạp thời gian của hàm $F(n)$. Ta có công thức sau:

- $T(0) = 0$
- $T(n) = T(n-1) + 1$ với $n > 0$

Ta sẽ giải công thức truy hồi bằng phương pháp thế:

$T(n) = T(n-1) + 1$

$= [T(n-2) + 1] + 1 = T(n-2) + 2$

$= [T(n-3) + 1] + 2 = T(n-3) + 3$

$= ...$

$= T(n-n) + n = T(0) + n = O(n)$.

Như vậy độ phức tạp thuật toán là $O(n)$.

Bài toán 2

a. Xây dựng và giải hệ thức truy hồi Từ dữ kiện về thuật toán được cho, với $T(n)$ là tổng thời gian để nấu n chiếc hamburger, ta có hệ thức - $T(1) = 2, T(2) = 2, T(n) = T(n-2) + 2$ với $n > 2$

Xét $n = 2k$ với k là số nguyên:

$$T(n) = T(n-2) + 2$$

$$T(2k) = T(2k-2) + 2$$
$$T(2(k)) = T(2(k-1)) + 2$$

Ta giải bằng phương pháp thế tương tự với Bài toán 1. Ta có hệ thức sau:

$$T(2(k)) = T(2[k-(k-1)]) + 2(k-1)$$
$$T(2k) = T(2) + 2(k-1)$$
$$T(2k) = 2k$$

→ Ta kết luận với n có dạng $n = 2k$ thì $T(n) = n$.

Xét $n = 2k+1$ với k là số nguyên:

$$T(n) = T(n-2) + 2$$
$$T(2k+1) = T(2k+1-2) + 2$$
$$T(2(k)+1) = T(2(k-1)+1) + 2$$

Tương tự sử dụng phương pháp thế, ta có hệ thức:

$$T(2(k)+1) = T(2(k-k)+1) + 2k$$

$$T(2k+1) = T(1) + 2k$$

$$T(2k+1) = 2k + 2$$

→ Ta kết luận với n có dạng $n = 2k+1$ thì $T(n) = n+1$. #### b. Phân tích tính tối ưu của thuật toán Lý do thuật toán đệ quy trên không phải là chiến lược tối ưu để giảm thiểu tổng thời gian nấu là vì với số chiếc hamburger n lẻ tức có dạng $2k+1$, ta phải mất 2 giây cuối để nấu một chiếc bánh cuối cùng.

Cụ thể: Theo thuật toán đã cho, ta sẽ mất $T(3) = 3+1$ (giây) để nấu $n = 3$ chiếc hamburger.

Trong khi đó, ta có thể nấu 3 chiếc hamburger trong 3 giây với ý tưởng nấu sau: - Giây thứ nhất: Nấu mặt trên của chiếc 1, và mặt trên của chiếc 2. - Giây thứ hai: Nấu mặt dưới của chiếc 1, và mặt trên của chiếc 3. - Giây thứ ba: Nấu mặt dưới của chiếc 2, và mặt dưới của chiếc 3.

NOTE: Ta quy định một chiếc bánh hamburger có 2 mặt là mặt trên và mặt dưới.

c. Đề xuất thuật toán tối ưu Từ ví dụ của câu b, ta có thể thêm trường hợp cơ sở của bài toán $T(3) = 3$.

Thuật toán cụ thể như sau: + Trường hợp cơ sở ($n \leq 3$): + $T(1) = T(2) = 2$ + $T(3) = 3$ + Hệ thức truy hồi: + $T(n) = T(n-2) + 2$.

Hệ thức sau khi giải $T(n) = n$. Chứng minh: + Ta cũng xét chẵn lẻ nhưng câu b. + $T(n) = n$ với $n = 2k$. + Với $n = 2k + 1$: + Tại hệ thức 2, sử dụng phương pháp thế:

$$T(2(k) + 1) = T(2(k - (k - 1) + 1) + 2(k - 1)$$

$$T(2k + 1) = T(3) + 2k - 2$$

$$T(2k + 1) = 2k - 2 + 3$$

$$T(2k + 1) = 2k + 1$$

$\rightarrow T(n) = n$ với $n = 2k + 1$.

Với mọi n ta mất đúng n phút để nấu n chiếc hamburger do đó đây là thời gian ít nhất để nấu n chiếc hamburger.

Bài toán 3

- Tham số biểu thị kích thước đầu vào ở đây là n.
- Phép toán cơ bản của thuật toán ở đây là phép tính cộng và phép nhân.
- Kiểm tra số lần phép toán cơ bản được thực hiện không có thay đổi với input.
- Gọi $T(n)$ là số lượng thao tác thực hiện các phép toán cơ bản của hàm `countPairs(n)`. Ta có hệ thức sau:

$$T(0) = T(1) = 0$$

$$T(n) = \sum_{i=0}^{n-1} (T(i) + T(n - 1 - i)) + n$$

$T(n)$ có thể được viết lại:

$$T(n) = 2 \sum_{i=0}^{n-1} T(i) + n$$

Gọi $S_i = \sum_{i=0}^{n} T(i)$. Ta có:

$$T(n) = 2S_{n-1} + n$$

Mà

$$S_n = S_{n-1} + T_n$$

nên ta có:

$$S_n = S_{n-1} + 2S_{n-1} + n$$

$$S_n = 3S_{n-1} + n$$

Sử dụng phương pháp tìm nghiệm tổng quát, ta khử n khỏi hệ thức:

$$S_n = 3S_{n-1}$$

Tiếp tục sử dụng phương pháp cây đệ qui,

$$S_n = O(3^n)$$

Như vậy, độ phức tạp của thuật toán là $O(3^n)$.