trong đó $u_0 = f(1)$. Khi đó

$$f(n) = f(b^k) = u_k = u_{\log_b n}.$$

Một số thuật toán chia để trị điển hình như tìm số lớn nhất của dãy, tìm kiếm nhị phân, và sắp xếp trộn.

Ví dụ 9.30. Trình bày một thuật toán chia để trị để tìm số lớn nhất M của dãy $a_1, a_2, \dots, a_n$, và đánh giá độ phức tạp của thuật toán.

Giải. Xét thuật toán chia để trị sau:

1) Nếu dãy chỉ có một phần tử, thì $M = a_1$. Nếu không thì xuống thực hiện bước (2).

2) Chia đôi dãy được hai dãy con $a_1, a_2, \dots, a_k$ và $a_{k+1}, a_{k+2}, \dots, a_n$. Mỗi dãy có $\frac{n}{2}$ số. Gọi M_1, M_2 là số lớn nhất của hai dãy con này.

3) $M = \max\{M_1, M_2\}$.

Chia đôi dãy $a_1, a_2, \dots, a_n$ thành hai dãy con.

```
1  def my_max(a):
2    n = len(a)            # 1 phép tính, 1 phép gán
3    if n == 1:            # 1 phép so sánh
4      return a[0]
5    k = n // 2            # 1 phép toán
6    M1 = my_max(a[:k])    # bài toán với cỡ đầu vào n/2, và 1 phép gán
7    M2 = my_max(a[k:])    # như dòng 6
8    if M1 > M2:           # 1 phép so sánh
9      return M1
10   else:
11     return M2

12 my_max([2, 8, 6, 1, 0, 5, 4, 1, 7])  # → 8
```

Gọi $f(n)$ là số phép so sánh để tìm số lớn nhất của dãy cỡ n, được dùng để đánh giá độ phức tạp của thuật toán. Ta có

$$f(n) = \underbrace{1}_{\text{bước 1}} + \underbrace{2 \cdot f\left(\frac{n}{2}\right)}_{\text{bước 2}} + \underbrace{1}_{\text{bước 3}} = 2f\left(\frac{n}{2}\right) + 2.$$

Xét $n = 2^k$, $a_k = f(n) = f(2^k)$ thì

$$a_k = 2f(2^{k-1}) + 2 = 2a_{k-1} + 2$$

trong đó $a_0 = f(1) = 0$.

Giải hệ thức đệ quy, ta được $a_k = 3 \cdot 2^k - 2$, hay $f(n) = 3n - 2 \in O(n)$. Thuật toán có độ phức tạp tuyến tính.

Thay vì chỉ đếm số phép so sánh, nếu đặt $f(n)$ là bao gồm các phép toán, phép so sánh, và phép gán của thuật toán trên, thì $f(n) = 2f\left(\frac{n}{2}\right) + 7$, tương ứng $a_k = 2a_{k-1} + 7$, với $a_0 = f(1) = 3$. Khi đó $a_k = 5 \cdot 2^k - 2$, tức là $f(n) = 5n - 2$. □

Ví dụ 9.31 (Tìm kiếm nhị phân)**.** Tìm vị trí của x trong dãy đơn điệu $a_1 < a_2 < ... < a_n$: nêu một thuật toán chia để trị và đánh giá độ phức tạp của thuật toán.

Giải. Xét thuật toán chia để trị tìm vị trí của x trong dãy $a_i, a_{i+1}, ..., a_j$ với $i \leq j$. Ban đầu $i = 1, j = n$.

1) Nếu dãy chỉ có một phần tử, tức là $i = j$, thì chỉ cần so sánh x với a_i [Nếu $x = a_i$ thì i là vị trí của x trong dãy, ngược lại không tìm được x trong dãy]. Ngược lại thì xuống thực hiện bước (2).

2) Chia đôi dãy được hai dãy con $a_i, a_{i+1}, ..., a_k$ và $a_{k+1}, a_{k+2}, ..., a_j$. Mỗi dãy có $\frac{n}{2}$ số. Do tính đơn điệu của dãy, nên nếu $x \leq a_k$ thì chỉ cần tìm x trong dãy thứ nhất, và nếu ngược lại, chỉ cần tìm x trong dãy thứ hai. Tức là, chỉ thực hiện một bài bài toán với cỡ đầu vào $\frac{n}{2}$.

```
1  def binary_search(x, a, i, j):
2    if i == j:              # 1 phép so sánh
3      if x == a[i]:         # 1 phép so sánh
4        return i
5      else:
6        return 'Không thấy'
7    k = (i + j) // 2        # 2 phép toán, 1 phép gán
8    if x <= a[k]:           # 1 phép so sánh
9      return binary_search(x, a, i, k)       # bài toán với cỡ
       đầu vào n/2
10   else:
11     return binary_search(x, a, k + 1, j)   # như dòng 9

12 a = [0, 2, 3, 6, 7, 10, 11, 15]
13 binary_search(10, a, 0, len(a) - 1)        # → 5
```

Đặt $f(n)$ là số phép so sánh cho bài toán với đầu vào cỡ n. Khi đó

$$f(n) = \underbrace{2}_{\text{dòng 2, 8}} + \underbrace{f\left(\frac{n}{2}\right)}_{\text{dòng 9 hoặc 11}}$$

Xét $n = 2^k \Leftrightarrow k = \log_2 n$, và $a_k = f(n) = f(2^k)$, thì

$$a_k = 2 + f(2^{k-1}) = 2 + a_{k-1}$$

trong đó $a_0 = f(1) = 2$ (dòng 2, 3). Suy ra $a_k = 2 + 2k$, tức là $f(n) = 2 + 2\log_2 n \in O(\log_2 n)$. Thuật toán có độ phức tạp loga. □

Ví dụ 9.32 (Sắp xếp trộn)**.** Sắp xếp dãy $a_1, a_2, \ldots, a_n$ thành dãy tăng dần: trình bày thuật toán chia để trị và đánh giá độ phức tạp của thuật toán.

Giải. Các bước của thuật toán:

1) Nếu dãy chỉ có 1 số, ta không cần làm gì cả. Ngược lại, chuyển xuống thực hiện bước (2).

2) Chia đôi dãy được hai dãy con $a_1, a_2, \ldots, a_k$ và $a_{k+1}, a_{k+2}, \ldots, a_n$ cỡ $\frac{n}{2}$.
   Sắp xếp hai dãy này, được hai dãy tăng.

3) Trộn hai dãy tăng này cho thành một dãy tăng.

Để thực hiện bước (3), ta dùng thuật toán dưới đây. Trong thuật toán này, để trộn hai dãy tăng cỡ m và n, số chu trình tối giản là $m + n$.

```
1  def merge(a, b):
2    n, m = len(a), len(b)
3    c = [0] * (n + m)
4    i, j, k = 0, 0, 0
5    while i < n and j < m:
6      if a[i] < b[j]:
7        c[k] = a[i]
8        i += 1
9      else:
10       c[k] = b[j]
11       j += 1
12     k += 1

13   while i < n:
14     c[k] = a[i]
15     i += 1
16     k += 1
17   while j < m:
18     c[k] = b[j]
19     j += 1
```

```
    k += 1
  return c


merge([2, 3], [1, 4, 5])   # → 1,2,3,4,5
```

Chương trình chính như sau:

```
def merge_sort(a):
  n = len(a)
  if n == 1:
    return a
  k = n // 2
  L = merge_sort(a[:k])        # sắp xếp dãy cỡ n/2
  R = merge_sort(a[k:])        # như dòng 6
  return merge(L, R)           # n/2 + n/2 = n chu trình tối giản

merge_sort([1, 5, 3, 4, 2])   # → 1,2,3,4,5
```

Gọi $f(n)$ là số chu trình tối giản của thuật toán với dãy cỡ n. Ta có

$$f(n) = 2f\left(\frac{n}{2}\right) + n$$

Xét $n = 2^k \Leftrightarrow k = \log_2 n$, và $a_k = f(n) = f(2^k)$, thì

$$a_k = 2f(2^{k-1}) + 2^k = 2a_{k-1} + 2^k,$$

trong đó $a_0 = f(1) = 0$.

Giải hệ thức đệ quy này, được $a_k = 2^k k$, tức là $f(n) = n \log_2 n$. Thuật toán có độ phức tạp $n \log_2 n$.

□

Dưới đây là một số kết luận, đánh giá về hệ thức chia để trị.

Định lý 9.1. *Cho $a, b, c \in \mathbb{Z}^+$, $b \geq 2$, và $f : \mathbb{Z}^+ \to \mathbb{R}$. Nếu*

$$f(1) = c, \quad \text{và}$$
$$f(n) = af\left(\frac{n}{b}\right) + c, \quad \text{với } n = b^k, k \geq 1,$$

thì với mọi $n = 1, b, b^2, \ldots$

$$f(n) = \begin{cases} c(\log_b n + 1) & \text{nếu } a = 1 \\ \dfrac{c(an^{\log_b a} - 1)}{a - 1} & \text{nếu } a \geq 2. \end{cases}$$