CS112.P11.KHTN - Bài tập nhóm 8

Nhóm 3
Nguyễn Hữu Đặng Nguyên - 23521045
Trần Vạn Tấn - 23521407

Ngày 11 tháng 11 năm 2024

Mục lục

1 Bài 1

1.1 Divide and Conquer

Ví dụ: Giải thuật Quicksort
Mô tả bài toán: Quicksort là một thuật toán sắp xếp rất hiệu quả và phổ biến. Nó hoạt động bằng cách chọn một phần tử trong mảng (gọi là "pivot"), sau đó phân chia mảng thành hai phần: các phần tử nhỏ hơn pivot và các phần tử lớn hơn pivot. Sau đó, thuật toán sẽ được gọi đệ quy để sắp xếp các phần con.
Cách áp dụng:

- Chọn một phần tử làm pivot.

- Phân chia mảng xung quanh pivot sao cho các phần tử bên trái nhỏ hơn pivot và bên phải lớn hơn pivot.
- Áp dụng quicksort đệ quy trên các phần con.

Ứng dụng thực tế: Quicksort được sử dụng trong các hệ thống cần sắp xếp nhanh với không gian bộ nhớ hạn chế. Chẳng hạn, nó thường được dùng trong các hệ thống cơ sở dữ liệu để sắp xếp dữ liệu nhanh chóng.

1.2 Decrease and Conquer

Ví dụ: Thuật toán Sắp xếp chèn (Insertion Sort)
Mô tả bài toán: Trong thuật toán sắp xếp chèn, chúng ta có một danh sách các phần tử cần sắp xếp. Mỗi lần lặp của thuật toán, chúng ta sẽ đưa một phần tử về đúng vị trí của nó trong dãy đã được sắp xếp trước đó, nhờ đó mảng từng bước trở nên có thứ tự.
Cách áp dụng Giảm để trị:

- Xem dãy cần sắp xếp như một mảng có kích thước giảm dần.
- Bắt đầu từ phần tử thứ hai (giả định rằng phần tử đầu tiên đã được sắp xếp), so sánh phần tử đó với các phần tử trong dãy đã sắp xếp trước đó và chèn nó vào đúng vị trí.
- Lặp lại quá trình này cho các phần tử còn lại trong danh sách, mỗi lần mở rộng dãy đã sắp xếp thêm một phần tử.

Ứng dụng thực tế:

- **Sắp xếp số lượng nhỏ**: Insertion Sort hoạt động tốt khi kích thước của mảng nhỏ hoặc khi mảng đã gần như có thứ tự.
- **Hệ thống nhúng và bộ nhớ hạn chế**: Sử dụng trong các hệ thống nhúng hoặc các môi trường bộ nhớ hạn chế.

1.3 Transform and Conquer

Ví dụ: Thuật toán Heapsort.
Mô tả bài toán: Heapsort là một thuật toán sắp xếp sử dụng cấu trúc dữ liệu Heap (cây nhị phân). Thuật toán này chuyển đổi mảng thành một max-heap (cấu trúc mà phần tử lớn nhất nằm ở gốc) rồi lần lượt đưa phần tử lớn nhất về cuối mảng đã sắp xếp.
Cách áp dụng:

- Chuyển mảng thành một max-heap.
- Đưa phần tử lớn nhất (ở gốc heap) về cuối mảng.
- Giảm kích thước của heap và duy trì tính chất heap cho các phần tử còn lại, tiếp tục đưa phần tử lớn nhất về cuối.
- Lặp lại cho đến khi tất cả các phần tử đã được sắp xếp.

Ứng dụng thực tế: Heapsort được sử dụng trong các hệ thống yêu cầu sắp xếp mà không muốn tiêu tốn quá nhiều bộ nhớ bổ sung. Nó thường được dùng trong các hệ thống nhúng hoặc các môi trường bộ nhớ hạn chế. Ngoài ra, heap còn được sử dụng trong hàng đợi ưu tiên (priority queue) và các bài toán liên quan.

2 Bài 2: Bài toán tổng luỹ thừa

2.1 Cách 1: Dùng công thức toán - Transform and Conquer

Giải thích: Dãy S là một cấp số nhân có công bội x. Tổng của cấp số nhân này có thể được tính bằng công thức:

$$S = \frac{x^{n+1} - 1}{x - 1}$$

với điều kiện $x \neq 1$. Nếu $x = 1$, tổng sẽ là $S = n + 1$ vì tất cả các số hạng đều bằng 1.
Mã giả

```
function calculateSum(x, n):
    if x == 1:
        return n + 1
    else:
        return (x^(n + 1) - 1) / (x - 1)
```

2.2 Cách 2: Chia để trị - Divide and conquer

Giải thích: Một cách khác để tính tổng $S = x^0 + x^1 + x^2 + \cdots + x^n$ là sử dụng kỹ thuật chia để trị. Ta có thể chia dãy thành các phần nhỏ hơn, sau đó tính tổng cho từng phần và ghép lại.

- Nếu $n = 0$, tổng $S = 1$.

- Nếu n là số chẵn, ta có thể chia dãy thành hai phần:

$$S = \left(x^0 + x^1 + \cdots + x^{\frac{n}{2}-1}\right) + x^{\frac{n}{2}} \times \left(x^0 + x^1 + \cdots + x^{\frac{n}{2}-1}\right)$$

- Nếu n là số lẻ, ta tách riêng x^n và tính tổng cho các số hạng còn lại:

$$S = x^n + \left(x^0 + x^1 + \cdots + x^{n-1}\right)$$

Bằng cách chia nhỏ dãy và tính tổng các phần con, chúng ta có thể giảm đáng kể số phép tính.

Mã giả

```
function calculateSum(x, n):
    if n == 0:
        return 1
    elif n % 2 == 0:
        half_sum = calculateSum(x, n // 2 - 1)
        return half_sum + x^(n // 2) * half_sum
    else:
        return x^n + calculateSum(x, n - 1)
```