VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY

CS112.P11.CTTN
PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

BTVN NHÓM 8

Sinh Viên :
Nguyễn Văn Minh
Đồng Quốc Thắng

Giảng viên :
Nguyễn Thanh Sơn

Ngày 9 tháng 11 năm 2024

Mục lục

Chương 1

Bài 1

1.1 Chia để trị

Chia để trị là 1 phương pháp áp dụng cho các bài toán có thể giải quyết bằng cách chia nhỏ ra thành các bài toán con từ việc giải quyết các bài toán con này. Sau đó lời giải của các bài toán nhỏ được tổng hợp lại thành lời giải cho bài toán ban đầu.
Phương thức triển khai bài toán

1. **Chia/Tách nhỏ:** Tại bước này thì bài toán ban đầu sẽ được chia thành các bài toán con cho đến khi không thể chia nhỏ được nữa. Các bài toán con kiểu sẽ trở thành 1 bước nhỏ trong việc giải quyết bài toán lớn.

2. **Trị/Giải quyết bài toán con:** Tại bước này ta sẽ phải tìm phương án để giải quyết cho bài toán con một cách cụ thể.

3. **Kết hợp lời giải lại để suy ra lời giải:** Khi đã giải quyết xong cái bài toán nhỏ, lặp lại các bước giải quyết đó và kết hợp lại những lời giải đó để suy ra kết quả cần tìm (có thể ở dạng đệ quy).

1.1.1 Ví dụ sử dụng

Tìm kiếm nhị phân

```
int binarySearch(int array[], int left, int right, int x)
{
    if (left > right) return -1;
    int mid = (left + right) / 2;
    if (x == array[mid])
        return mid;
    if (x < array[mid])
        return binarySearch(array, left , mid-1, x);
    if (x > array[mid])
        return binarySearch(array, mid+1 , right, x);
}
```

Quicksort

```
int partition(int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low-1);
    for (int j=low; j<high; j++)
    {
        if (arr[j] <= pivot)
        {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    int temp = arr[i+1];
    arr[i+1] = arr[high];
    arr[high] = temp;

    return i+1;
}

void sort(int arr[], int low, int high)
{
    if (low < high)
    {
        int pi = partition(arr, low, high);
        sort(arr, low, pi-1);
        sort(arr, pi+1, high);
    }
}
```

1.2 Giảm để trị

Kỹ thuật thiết kế giảm để trị lợi dụng mối liên hệ giữa lời giải cho một thể hiện của một bài toán và lời giải cho một thể hiện nhỏ hơn của cùng một bài toán.

1.2.1 Ví dụ sử dụng

Insertion sort

```
procedure insertion;
var i, j, v: integer;
begin
   for i := 2 to N do
   begin
       v := a[i];
```

```
        j := i;
        while a[j-1] > v do
        begin
            a[j] := a[j-1]; // pull down
            j := j - 1;
        end;
        a[j] := v;
    end;
end;
```

Tìm UCLN của 2 số

```
while n  0 do
    r := m mod n;
    m := n;
    n := r;
endwhile
return m;
```

1.3 Biến đổi để trị

Kỹ thuật này thường được triển khai theo 2 bước

1. Thể hiện của bài toán được biến đổi để chuyển sang một dạng để dẫn đến lời giải. Ở bước 1 này có thể có nhiều biến thể
   - Biến thể để đưa đến một thể hiện đơn giản hơn của bài toán.
   - Biến thể để đưa đến một biểu diễn khác của cùng bài toán
   - Biến thể để đưa đến một thể hiện của một bài toán khác mà đã có tồn tại giải thuật
2. Tìm ra lời giải cho bài toán.

1.3.1 Cây tìm kiếm nhị phân

```
Procedure TransformAndConquerSearch(arr, x)
    Input: Danh sach arr v phan tu can tim x
    Output: True neu x ton tai trong arr, False neu khong ton tai

    BuildBST(arr)
    Return SearchBST(root, x)
End Procedure
```

Chương 2

Bài 2

2.1 Đề bài:

Cho hai số nguyên x và n ($x \leq 10^{18}, n \leq 10^{18}$), tính tổng:

$$S = x^0 + x^1 + x^2 + \cdots + x^n$$

Ví dụ: Với $x = 5$ và $n = 5$, ta có $S = 3906$.
Yêu cầu:

- Suy nghĩ bài toán trên có mấy cách giải (Gợi ý có ít nhất 2 cách giải) và đối với mỗi cách hãy cho biết ứng dụng những kĩ thuật gì đã học.
- Viết mã giả cho các thuật toán các bạn đã nghĩ ra ở trên

2.2 Brute Force

Kỹ thuật: Phương pháp cơ bản, tính trực tiếp từng lũy thừa x^i với $i \in [0, n]$ và cộng chúng lại.
Mã giả:

```
function sum_of_powers(x, n):
    S = 0
    for i = 0 to n:
        S = S + x^i
    return S
```

2.3 Lũy thừa nhanh

Kỹ thuật: Chia để trị (Divide and Conquer).
Mã giả:

```
function fast_power(x, i):
    if i == 0:
        return 1
    y = fast_power(x, i // 2)
```

```
    if i is even:
        return y * y
    else:
        return x * y * y

function sum_of_powers_fast(x, n):
    S = 0
    for i = 0 to n:
        S = S + fast_power(x, i)
    return S
```

2.4 Công thức cấp số nhân

Kỹ thuật: Biến đổi và trị (Transform and Conquer).
Nếu $x \neq 1$, tổng $S = x^0 + x^1 + \cdots + x^n$ có thể tính bằng công thức cấp số nhân:

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

Mã giả:

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