TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

———-oOo———-

BÀI TẬP SỐ 04.c
THIẾT KẾ GIẢI THUẬT
PHƯƠNG PHÁP CHIA ĐỂ TRỊ

Giảng viên hướng dẫn: ThS. Huỳnh Thị Thanh Thương

Nhóm sinh viên:

1. Phạm Bá Đạt 17520337
2. Phan Thanh Hải 18520705

TP. HỒ CHÍ MINH, 16/12/2019

Mục lục

Bài tập 1

BÀI TOÁN TÌM MAX MIN

Tìm giá trị max, min trong đoạn $[l, r]$ của mảng A có n phần tử.

a. Phát biểu bài toán

Cho một mảng A gồm n các số nguyên ($n > 0$). Cần tìm giá trị lớn nhất và giá trị nhỏ nhất của mảng A trên một đoạn $[l, r]$, với điều kiện là $l \leq r$.

INPUT

Một mảng A gồm n các số nguyên ($n > 0$).

Hai số nguyên không âm l và r thể hiện vị trí bắt đầu và vị trí kết thúc duyệt mảng.

OUTPUT

Giá trị lớn nhất và giá trị nhỏ nhất của mảng A trên một đoạn $[l, r]$.

b. Ý tưởng giải

Divide: Chia đôi mảng ban đầu thành 2 đoạn con nhỏ hơn (kích thước bằng nhau). Quá trình chia này cứ tiếp tục cho đến khi đoạn con chỉ còn có 1 phần tử. Khi đó giá trị lớn nhất và giá trị nhỏ nhất cần tìm là chính phần tử đó.

Conquer: Gọi đệ quy tìm giá trị nhỏ nhất và giá trị lớn nhất của mảng con.

Combine: Cập nhật giá trị *max* và *min* của từ 2 mảng con.

c. Thuật giải

Hàm sẽ không trả về trực tiếp giá trị của *min* và *max*, mà thay vào đó sẽ thay đổi trực tiếp lên biến *min* và *max*. Do đó tham số đầu vào của hàm **MinMax** có thêm biến *min* và *max*, được truyền vào theo kiểu tham biến (pass by reference).

```
MinMax(array[], left, right, &min, &max)
if left = right
    min = max = arr[left];
mid = (left + right) / 2;
MinMax(array, left, mid, min, max);
MinMax(array, mid + 1, right, min1, max1);
if min1 < min
    min = min1;
if max < max1
    max = max1;
```

d. VD minh họa

INPUT	OUTPUT
1 10 0 0	10 10
6 1 2 3 4 5 6 0 4	1 5
6 12 4 5 10 9 7 0 5	4 12

e. Độ phức tạp của thuật toán

$T(n) = c_1, \text{if } n = 1$

$T(n) = 2T\left(\frac{n}{2}\right) + c_2, \text{if } n \leq 2$

Áp dụng định lý Master case 1, ta có: $T(n) = \Theta(n^{log_2 2}) = \Theta(n)$

Bài tập 2

MERGESORT

a. Phát biểu bài toán

Cho một mảng A gồm n các số nguyên ($n > 0$). Sắp xếp mảng A theo thứ tự cho trước (cho cụ thể thì ta cần sắp xếp thứ tự tăng dần theo giá trị phần tử trong mảng).

INPUT

Một mảng A gồm n các số nguyên ($n > 0$).

OUTPUT

Mảng A gồm n các số nguyên đã được sắp xếp tăng dần theo giá trị phần tử trong mảng.

b. Ý tưởng giải

Divide: Chia đôi đoạn ban đầu thành 2 đoạn con nhỏ hơn (số lượng phần tử nhỏ hơn). Quá trình chia này cứ tiếp tục cho đến khi mảng con chỉ còn có 1 phần tử. Khi đó thì mảng đã được sắp xếp (theo thứ tự).

Conquer: Gọi đệ quy sắp xếp mảng con.

Combine: Sau khi sắp xếp 2 mảng con, ta cần trộn 2 mảng con thành 1 mảng mà vẫn đảm bảo thứ tự sắp xếp.

Ý tưởng của việc trộn 2 mảng con (hàm **Merge**) đã sắp xếp là lần lượt duyệt từng phần tử trong 2 mảng con, nếu phần tử nào trong mảng con A có giá trị nhỏ hơn trong phần tử trong mảng con B thì ta đưa vào mảng chính và xét phần tử tiếp theo của mảng con A (trường hợp tương tự nếu mảng con B có giá trị phần tử đang xét nhỏ hơn giá trị đang xét trong mảng con A).

Quá trình này tiếp tục cho đến khi xét hết mọi phần tử trong mảng con A và mảng con B.

c. Thuật giải

```
MergeSort(A[], left, right)
 1. if left < right
 2.     min = (left + right) / 2;
 3.     MergeSort(A, left, mid);
 4.     MergeSort(A, mid + 1, right);
 5.     Merge(A, left, mid, right);
```

d. VD minh họa

INPUT	OUTPUT
1 10	10
6 1 2 3 4 5 6	1 2 3 4 5 6
6 3 10 9 2 1 5	1 2 3 5 9 10

e. Độ phức tạp của thuật toán

Quá trình trộn 2 mảng con thực chất là quá trình duyệt 2 mảng con nên thời gian thực hiện là $O(n)$.

$T(n) = c_1, \text{if } n = 1$

$T(n) = 2T\left(\frac{n}{2}\right) + c_2 n, \text{if } n \leq 2$

Áp dụng định lý Master case 2, ta có: $T(n) = \Theta(n \log n)$

Bài tập 3

QUICKSORT

a. Phát biểu bài toán

Cho một mảng A gồm n các số nguyên ($n > 0$). Sắp xếp mảng A theo thứ tự cho trước (cho cụ thể thì ta cần sắp xếp thứ tự tăng dần theo giá trị phần tử trong mảng).

INPUT

Một mảng A gồm n các số nguyên ($n > 0$).

OUTPUT

Mảng A gồm n các số nguyên đã được sắp xếp tăng dần theo giá trị phần tử trong mảng.

b. Ý tưởng giải

Divide: Chọn một phần tử trung vị $A[pivot]$ bất kỳ trong mảng. Cần phân hoạch các phần tử còn lại trong mảng sao cho những phần tử bên trái phần tử trung vị ($A[0]..A[pivot-1]$) có giá trị nhỏ hơn hoặc bằng $A[pivot]$, và những phần tử bên phải phần tử trung vị ($A[pivot+1]..A[n-1]$) có giá trị lớn hơn $A[pivot]$.

Conquer: Gọi đệ quy sắp xếp mảng con.

Combine: Không cần tổng hợp kết quả vì quá trình sắp xếp đã diễn ra tường minh trong khâu phân hoạch.

c. Thuật giải

```
QuickSort(A[], left, right)
if left < right
    pivot ← Partition(A, left, right);
    QuickSort(A, left, mid - 1);
    QuickSort(A, mid + 1, right);
```

```
Partition(A[], left, right)
x = A[right];
i = left - 1;
for j = left to right - 1
    if A[j] ≤ x;
        i = i + 1;
        Hoán vị phần tử A[i] với A[j];
Hoán vị phần tử A[i + 1] với A[r];
return i + 1;
```

d. VD minh họa

INPUT	OUTPUT
1 10	10
6 1 2 3 4 5 6	1 2 3 4 5 6
6 3 10 9 2 1 5	1 2 3 5 9 10

Bài tập 4

BÀI TOÁN SẮP HẠNG TRONG KHÔNG GIAN 2D

a. Phát biểu bài toán

INPUT

Một mảng S gồm n các điểm ($n > 0$). Mỗi điểm được biểu diễn bởi các thông tin hoành độ, tung độ trong hệ trục tọa độ 2D và hạng của nó (ban đầu bằng 0).

OUTPUT

Hạng của các điểm lần lượt có trong mảng S.

b. Ý tưởng giải

Bước 1: Chia không gian cần xếp hạng ra làm 2 với kích thước giảm một nửa

a. Nếu tập S chỉ có 1 một phần tử thì rank của điểm đó là 0.

b. Ngược lại, ta chia không gian ra làm 2 không gian con (chia theo trục hoành).

Bước 2: Gọi đệ quy xếp hạng các điểm trong 2 không gian con.

Bước 3: Kết quả xếp hạng các điểm của không gian A không cần tổng hợp bởi vì kết quả đã quá tường minh. Kết quả xếp hạng các điểm của không gian B = kết quả xếp hạng của điểm đó trong không gian B + số điểm trong không gian A mà giá trị tung độ nhỏ giá trị tung độ của điểm đang xét trong không gian B.

c. Thuật giải

```
SapXepTheoHoanhDoTangDan(S, n);
XepHang(S, n, left, right)
if left = right
    rank[left] = 0;
mid = (left + right) / 2;
XepHang(S, n/2, left, mid);
XepHang(S, n/2, mid + 1, right);
for i = mid + 1 to r
    for j = left to mid
        if S[i].tung > S[j].tung
            rank[i]++;
```

d. VD minh họa

INPUT	OUTPUT
5 961 404 640 145 983 888 539 71 437 532	2 1 4 0 0
5 1 4 6 2 8 0 2 7 7 5	0 0 0 1 2
7 1 2 -3 - 4 5 6 -7 -8 3 7 -5 -9 0 0	4 2 5 0 5 0 3

e. Độ phức tạp của thuật toán

Quá trình sắp xếp các điểm, nếu ta sử dụng thuật toán hiệu quả như MergeSort thì thời gian thực hiện là $O(n \log n)$. Quá trình duyệt các điểm trong không gian A và B để xếp hạng các điểm trong không gian B có thời gian thực hiện là $O(n)$.

$T(n) = c_1, \text{if } n = 1$

$T(n) = 2T\left(\frac{n}{2}\right) + c_2 n \log n + c_3 n, \text{if } n \leq 2$

Áp dụng định lý Master, ta có: $T(n) = \Theta(n \log^2 n)$

Bài tập 5

BÀI TOÁN VẠCH THƯỚC (DRAWING A RULER)

Cho một cây thước có độ dài l và một chiều cao h nguyên cho trước.
Tại vị trí chính giữa của cây thước, vạch một vạch có chiều cao h.
Tại vị trí 1/4 và 3/4 của cây thước, vạch một vạch có chiều cao $h - 1$.
Tại vị trí 1/8, 3/8, 5/8, và 7/8 của cây thước, vạch một vạch có chiều cao $h - 2$.
...
Cho đến khi không thể vạch được nữa (chiều của vạch bằng 0).

a. Phát biểu bài toán

INPUT

Một số nguyên dương l là độ dài của cây thước. Một số nguyên dương $h (h \leq l)$ là độ cao của vạch ban đầu.

OUTPUT

Xuất ra tọa độ tương ứng của vạch và độ cao của vạch đó

b. Ý tưởng giải

Divide: Chia cây thước ra làm 2 phần có độ dài bằng nhau, là cây thước con bên trái và phải.

Conquer: Gọi đệ quy kẻ vạch cho cây thước con bên trái và phải.

Combine: Sau khi đã kẻ vạch xong cho cây thước con bên trái và phải, ta tiến hành kẻ vạch vào chính giữa cây thước.

c. Thuật giải

```
DrawingARuler(left, right, length, height)
 1. if height = 0
 2.     return;
 3. mid = (left + right) / 2;
 4. DrawingARuler(left, mid, length / 2, height - 1);
 5. DrawingALine(mid, height);
 6. DrawingARuler(mid, right, length / 2, height - 1);
```

d. VD minh họa

INPUT	OUTPUT
1 1	0.5 1
4 3	0.5 1 1 2 1.5 1 2 3 2.5 1 3 2 3.5 1

e. Độ phức tạp của thuật toán

$T(n) = c_1, \text{if } n = 1$

$T(n) = 2T\left(\frac{n}{2}\right) + c_2, \text{if } n \leq 2$

Áp dụng định lý Master case 2, ta có: $T(n) = \Theta(n \log n)$

Bài tập 6

BÀI TOÁN NHÂN 2 SỐ NGUYÊN LỚN

a. Phát biểu bài toán

INPUT

Hai số nguyên lớn X và Y có cùng n chữ số.

OUTPUT

Tích hai số nguyên lớn X và Y.

b. Thuật giải

```
NhanSoNguyen(A, B, n)
 1. if n = 1
 2.     return A * B;
 3. A1 ← TachNuaDau(A, n/2);
 4. A0 ← TachNuaSau(A, n/2);
 5. B1 ← TachNuaDau(B, n/2);
 6. B0 ← TachNuaSau(B, n/2);
 7. C3 ← NhanMaTran(A1, B1, n/2);
 8. C2 ← NhanSoNguyen(A1, B0, n/2);
 9. C1 ← NhanSoNguyen(A0, B1, n/2);
10. C0 ← NhanSoNguyen(A0, B0, n/2);
11. return C3 * 10^n + (C2 + C1) * 10^(n/2) + C0;
```

Từ dòng 3 đến dòng 10, thời gian thực hiện phép toán là tuyến tính. Từ dòng 11 đến 18, ta gọi 8 lần đệ quy hàm **NhanMaTran** với kích thước giảm đi một nửa. Chi phí tổng hợp kết quả từ dòng 19 đến dòng 22 là $\Theta(n^2)$. Vậy phương trình thời gian của thuật toán trên có dạng:

$$T(n) = 4T\left(\frac{n}{2}\right) + \Theta(n)$$

Áp dụng định lý Master case 1, ta có: $T(n) = \Theta(n^{log_2 4}) = \Theta(n^2)$

Cải tiến:

```
NhanSoNguyen(A, B, n)
 1. if n = 1
 2.     return A * B;
 3. A1 ← TachNuaDau(A, n/2);
 4. A0 ← TachNuaSau(A, n/2);
 5. B1 ← TachNuaDau(B, n/2);
 6. B0 ← TachNuaSau(B, n/2);
 7. C2 ← NhanSoNguyen(A1, B1, n/2);
```

```
 8. C0 ← NhanSoNguyen(A0, B0, n/2);
 9. C1 ← NhanSoNguyen(A1 + A0, B1 + B0, n/2) - (C2 + C0);
10. return C2 * 10^n + C1 * 10^(n/2) + C0;
```

Chứng minh:

$$\begin{aligned} c_1 &= (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0) \\ &= (a_1 + a_0) * (b_1 + b_0) - (a_1 * b_1 + a_0 * b_0) \\ &= a_1 * b_1 + a_1 * b_0 + a_0 * b_1 + a_0 * b_0 - a_1 * b_1 - a_0 * b_0 \\ &= a_1 * b_0 + a_0 * b_1 \end{aligned}$$

$$T(n) = 3T\left(\frac{n}{2}\right) + c_2 n$$

Áp dụng định lý Master case 2, ta có: $T(n) = \Theta(n^{log_2 3}) = \Theta(n^{2.81})$

Bài tập 7

SỐ CHÍNH PHƯƠNG

Bài toán tìm tất cả các số chính phương trong một danh sách các số nguyên cho trước.

a. Viết hàm kiểm tra số chính phương.

b. Áp dụng chia để trị để viết thuật toán tìm tất cả các số chính phương.

c. Phân tích độ phức tạp của thuật toán tìm tất cả các số chính phương.

```
TimChinhPhuong(A, n, left, right)
if left = right
    if KiemTraChinhPhuong(left) = true
    Xuat(A[left]);
mid = (left + right) / 2;
TimChinhPhuong(A, n/2, left, mid);
TimChinhPhuong(A, n/2, mid + 1, right);
```

```
KiemTraChinhPhuong(n)
for i = 2 to n/2
    if i*i = n;
        return true;
return false;
```

Bài tập 8

ĐẾM SỐ NGHỊCH THẾ (COUNTING INVERSIONS)

Cho một mảng A gồm n các số nguyên ($n > 0$). Hai số $A[i]$ và $A[j]$ được gọi là nghịch thế nếu $A[i] > A[j]$ với $i < j$. Đếm số nghịch thế có trong mảng A.