Bài toán “**Bố trí phòng họp**”

1. **Đề bài** :Có N cuộc họp cùng đăng kí tại một phòng họp, cuộc họp thứ i bắt đầu vào thời điểm ai và kết thúc ở thời điểm bi. Mỗi thời điểm phòng họp chỉ phục vụ được một cuộc họp. Hãy bố trí để phòng họp phục vụ được nhiều cuộc họp nhất
2. Phân tích

-Dạng cơ bản:Dãy con đơn điệu tăng dài nhất

-Rõ ràng nếu i và j với i < j là hai cuộc họp được bố trí thì b[i] ≤a[j] ≤b[j]. Đâychính là quan hệ thứ tự được bài toán ẩn đi. Do đó trước hết ta sẽ sắp xếp lại các cuộc họp theo thứ tự tăng dần về thời điểm kết thúc.

Gọi L[i] là số cuộc họp nhiều nhất có thể tổ chức nếu chọn cuộc họp i là cuộc họp cuối cùng .Suy ra số cuộc họp nhiều nhất có thể tổ chức là giá trị lớn nhất của các L[i] khi đã hoàn thành bảng phương án

Nếu có thể tổ chức 1 số cuộc họp khác trước cuộc họp i thì số cuộc họp tối đa có thể tổ chức là L[i] ={L[1]…L[i-1]}+1.

“Max” thỏa mãn điều kiện cuộc họp cuối cùng của dãy cuộc họp đó phải kết thúc trước hoặc đúng lúc cuộc họp i bắt đầu. Hay nói rõ hơn L[j] thỏa mãn phải có b[j] <= a[i]

Nếu không thể tổ chức 1 cuộc họp nào trước cuộc họp i thì L[i] = 1

Tính dần dần các L[i] theo công thức: L[i] = max(i-1) + 1

Với hàm max(k) trả về số cuộc họp nhiều nhất có thể tổ chức trước cuộc họp i

1. **Cơ sở quy hoạch động**

L[1] = số cuộc họp có thể tổ chức nếu chọn cuộc họp số 1 là cuộc họp cuối cùng = 1

1. **Tìm kết quả tối ưu**

Số cuộc họp nhiều nhất có thể tổ chức là {L[i]}

Chú ý là sau khi tìm L[i] để biết số cuộc họp nhiều nhất có thể tổ chức, ta còn phải truy vết từ đó nên thay vì tìm chính xác L[i], ta đi tìm chỉ số i của nó để đỡ phải tìm lại.

1. **Truy vết**

Khác với bài toán Dãy con đơn điệu tăng dài nhất, việc sắp thứ tự trong bài này chỉ

dựa trên một yếu tố thời điểm kết thúc các cuộc họp do đó nếu ta chỉ truy vết như trong ví dụ trước thì có thể dẫn tới tình huống khi tìm cuộc họp bố trí trước cuộc họp i ta tìm được một cuộc họp j nào đó đứng trước cuộc họp i trong dãy được sắp nhưng lại không thể bố trí được

Bắt đầu từ L[i] là giá trị lớn nhất của mảng L

Kết thúc khi tới giá trị L[j] = 1

Với mỗi L[i] đang xét, tìm ra L[j] đứng trước L[i] đầu tiên thỏa mãn

L[j] = L[i] – 1

b[j] <= a[i]

In cuộc họp j và chuyển sang xét L[j]

Thêm nữa, khi bắt đầu tìm L[j] đứng trước L[i] thì ta nên bắt đầu từ i-1 về 1 thay vì bắt đầu từ 1 và duyệt tới i-1 vì đi theo chiều ngược lại thường nhanh gặp giá trị cần tìm hơn.

1. Thiết kế

/\*Hàm max\*/

max(i){

max = 0

for j = 1…i

if(L[j] > max && b[j] <= a[i])

max = L[j]

return max

}

/\*Tạo cơ sở quy hoạch động\*/

L[1] = 1

/\*Xây dựng bảng phương án\*/

for i = 2…n

L[i] = max(i-1) + 1

/\*Số cuộc họp nhiều nhất có thể tổ chức\*/

maxIndex = 1

for i = 2…n

if L[i] > L[maxIndex]

maxIndex = i

print L[maxIndex]

/\*Lưu các cuộc họp được chọn vào ngăn xếp\*/

i = maxIndex

top = 0

while(L[i] != 1)

stack[top] = i

top = top + 1

for j = i-1…1

if(L[j] = L[i] – 1 && b[j] <= a[i])

i = j

break

stack[top] = i

/\*In các cuộc họp theo đúng thứ tự\*/

while(top >= 0)

i = stack[top]

print i, a[i], b[i]

top = top – 1

1. Vi dụ

Input Output

|  |  |  |
| --- | --- | --- |
| 10 | | |
| 1 | 3 | 8 |
| 2 | 4 | 6 |
| 3 | 8 | 9 |
| 4 | 7 | 8 |
| 5 | 3 | 4 |
| 6 | 1 | 4 |
| 7 | 1 | 8 |
| 8 | 4 | 8 |
| 9 | 2 | 4 |
| 10 | 2 | 8 |

|  |  |  |
| --- | --- | --- |
| 4 | | |
| 1 | 3 | 4 |
| 2 | 4 | 6 |
| 3 | 7 | 8 |
| 4 | 8 | 9 |

1. Đánh giá độ phức tạp của thuật toán:

- Thuật toán sắp xếp có độ phức tạp về mặt thời gian thực hiện là O(Nlog N).

Việc sắp xếp lại các cuộc họp cũng có độ phức tạp O(Nlog N).

- Thủ tục bố trí phòng họp có độ phức tạp cho việc lập bảng phương án là O (N2).

- Thủ tục truy vết có số lần so sánh lớn nhất là 3N, do đó chi phí tối đa là O (N) .

Tóm lại chi phí thời gian cho bài toán là cỡ O (N2).