Thuật toán bài TWOVALS

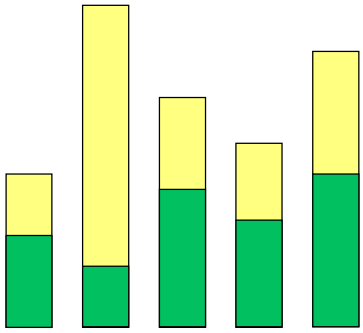
Thuật toán bài này không khó, nhưng khâu coding cần rất khéo léo mới có thể đạt được 2 yêu cầu sau:

- Độ phức tạp $O(n)$ về thời gian
- Độ phức tạp $O(1)$ về bộ nhớ. Cái này tôi không ép chặt nhưng để code không cần mảng nào hết cần khá "nghệ thuật"

Thuật toán bài SCHEDULE

Trường hợp vô nghiệm khá dễ phát hiện: Khi $d < \sum_{i=1}^{n} a_i$ hoặc $d > \sum_{i=1}^{n} c_i$, có thể xử lý riêng hoặc lồng ghép vào trong lõi thuật toán bên dưới.

Ta coi như cần đổ d lít nước và n bình sao cho bình thứ i có tối thiểu a_i lít và có tối đa c_i lít. Trước hết ta đổ vào mỗi bình i đúng a_i lít. Trong biểu đồ cột dưới đây, chiều cao cột là dung tích tối đa c_i, phần màu xanh là dung tích tối thiểu a_i ta đã đổ đầy:



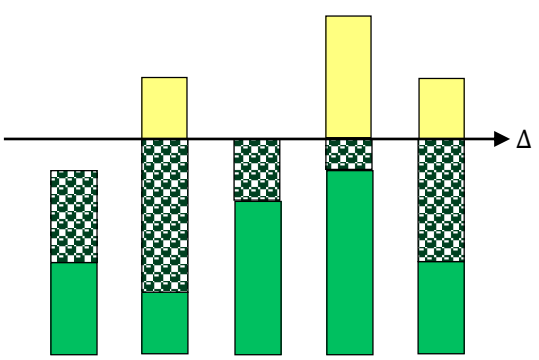
Với một ngưỡng Δ, viết hàm $Align(\Delta)$, hàm này thực hiện thuật toán "thử rót nước" như sau: Với mỗi bình i, ta cố gắng đổ thêm một lượng nước tối đa sao cho lượng nước trong bình không vượt quá Δ nhưng vẫn nằm trong phạm vi $[a_i; c_i]$, cụ thể là:

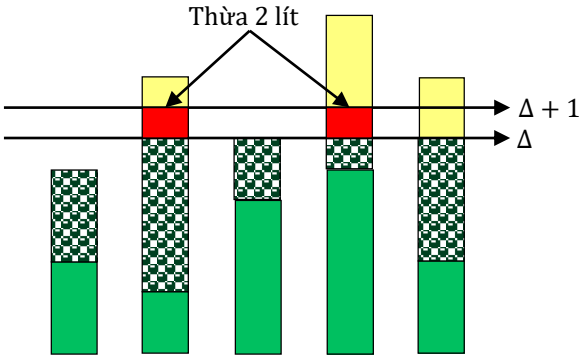
- Nếu $\Delta \leq a_i$: Bỏ qua
- Nếu $a_i < \Delta \leq c_i$: Rót thêm vào bình i để lượng nước trong bình đạt đúng Δ lít
- Nếu $c_i < \Delta$: Rót thêm vào bình i để lượng nước trong bình đạt đúng c_i lít

Hàm trả về tổng lượng nước rót trong các bình.

```
function Align(Δ): Lượng nước có thể rót thêm vào các bình;
var
  i: Integer;
begin
  Result := 0;
  for i := 1 to n do //thêm vào bình i sao cho lượng nước trong bình không vượt quá Δ và không vượt quá c[i]
    Result + Max(a[i], Min(c[i], Δ));
end;
```

Nhận xét: Nếu $q = Align(\Delta)$ thì trong mọi cách đổ thêm q lít vào các bình, cách đổ trong hàm Align cho lượng nước trong bình ít nước nhất là cực đại (xem hình dưới bên trái).





Hàm $Align(\Delta)$ có độ phức tạp $O(n)$. Dùng thuật toán tìm kiếm nhị phân, xác định giá trị Δ lớn nhất sao cho $Align(\Delta) \leq d$. Tức là:

$$Align(\Delta) \leq d < Align(\Delta + 1)$$

Dùng kỹ thuật trong hàm Align đổ nước thêm nước theo ngưỡng Δ. Nếu còn thừa nước, xét các bình đang có lượng nước bằng Δ và có thể đổ thêm ($c_i > \Delta$), đổ thêm vào mỗi bình này đúng 1 lít nước cho tới khi hết nước. (Xem hình trên bên phải)

Giải thích: Giá trị Δ tìm được chính là lượng nước nhỏ nhất trong các bình theo phương án tối ưu. Nếu còn thừa nước, giá trị lượng nước lớn nhất trong các bình chắc chắn $\geq \Delta + 1$ nên việc đổ thêm chỉ 1 lít nước vào những bình đang có Δ lít không làm ảnh hưởng tới mục tiêu cần đạt.

Độ phức tạp $O(n \log maxC)$. Có thể cải tiến xuống còn $O(n \log n)$ nếu tiền xử lý hai mảng A, C bằng một thuật toán sắp xếp (tuy nhiên điều này không thực sự cần thiết)

Thuật toán bài MINER

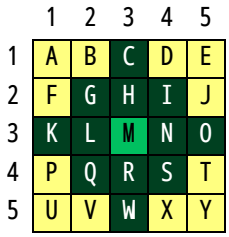
Nhận xét trước hết là nếu không bị giới hạn bởi cạnh bảng, vùng các ô đến được của một robot có dạng một hình vuông nghiêng 45 độ so với cạnh bảng. Nếu Robot xuất phát từ ô (i, j) thì 4 đỉnh của hình vuông này là các ô $(i + k, j), (i, j + k), (i - k, j), (i, j - k)$

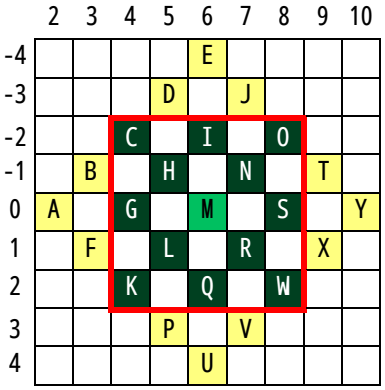
Ví dụ với $k = 2$ và bảng kích thước 5×5 như hình dưới đây

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I	J
3	K	L	M	N	O
4	P	Q	R	S	T
5	U	V	W	X	Y

Hình vuông dạng này khá khó để xử lý trong những thao tác truy vấn tổng, max, min. Kỹ thuật để đơn giản hóa khá phổ biến là ta "xoay" sao cho cạnh hình vuông song song với cạnh bảng (hình vuông trực giao) nhằm dễ dàng tận dụng các cấu trúc dữ liệu truy vấn phạm vi.

Khởi tạo bảng A' ban đầu các phần tử bằng 0. Với mỗi phần tử $a[i, j]$, ta đặt $a'[i - j, i + j] := a[i, j]$. Cách ánh xạ tọa độ này dựa trên quan sát: Trên bảng A, những ô nằm trên một đường chéo song song với đường chéo chính có tính chất hàng $-$ cột $=$ hằng số, còn những ô nằm trên một đường chéo song song với đường chéo phụ có tính chất hàng $+$ cột $=$ hằng số. Dưới đây là ví dụ về bảng A và bảng A' tương ứng:



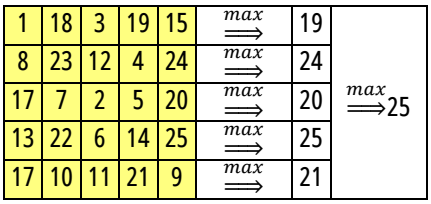


Bởi những ô trong bảng A' không có ô tương ứng trong bảng A được gán bằng 0. Việc tìm giá trị lớn nhất trong một phạm vi hình vuông nghiêng 45° của bảng A được quy về tìm giá trị lớn nhất trong một phạm vi hình vuông trực giao của bảng A'. Cạnh hình vuông truy vấn trên bảng A' có độ dài $2k + 1$ (Suy ra từ phép ánh xạ tọa độ 4 ô đỉnh).

Việc tìm max trong tất cả các hình vuông kích thước $(2k + 1) \times (2k + 1)$ thực hiện như sau:

- Bước 1: Với mỗi ô (x, y), xác định $c[x, y]$ là số lớn nhất ghi trong $2k + 1$ ô tính từ ô (x, y) về bên phải của bảng A'
- Bước 2: Với mỗi ô (x, y), xác định $d[x, y]$ là số lớn nhất ghi trong $2k + 1$ ô tính từ ô (x, y) về phía dưới của bảng C

Khi đó, $\forall x, y$; $d[x, y]$ là giá trị lớn nhất trong hình vuông kích thước $(2k + 1) \times (2k + 1)$ của bảng A' với góc trái trên là ô (x, y):



Bởi việc tính các giá trị lớn nhất trong tất cả các phạm vi $2k + 1$ phần tử liên tiếp trên một mảng độ dài n có thể thực hiện trong thời gian $O(n)$ (sử dụng hàng đợi hai đầu – DEQUEUE). Suy ra độ phức tạp tính toán của thuật toán là $O(mn)$.