SIROCCO CUP 2026 / CT1

Chuyên - Đề 1

Thứ Hai, 02/11/2026 · Online

180 phút · 4 bài · 400 điểm

Quy hoạch động, lưới và cây

Bài	Tên bài	Tag
A	Chữ số 0 tận cùng	Số học, Phân tích thừa số
B	Buổi trà thư giãn	Quy hoạch động đoạn, Tối ưu bộ nhớ
C	Vượt lũ	Đường đi minimax, Quy hoạch động trên lưới, Tìm kiếm đáp án
D	Lá	Cây, DFS có hoàn tác, Duy trì top-k

Problemset trực tuyến
https://codeforces.com/gym/728949/problems

CT1-A / KHÁM PHÁ LỜI GIẢI

A. Chữ số 0 tận cùng

Số học | Phân tích thừa số

ĐIỂM MẤU CHỐT

Mỗi chữ số 0 tận cùng cần một cặp thừa số 2 và 5; không được nhân cả dãy.

Từ quan sát đến thuật toán

Một số có ít nhất z chữ số 0 tận cùng khi và chỉ khi chia hết cho 10^z = 2^z×5^z. Vì thế số chữ số 0 tận cùng của P là $\min(v_2(P), v_5(P))$.

Số mũ trong một tích bằng tổng số mũ trong các thừa số. Đọc từng a_i, chia liên tiếp cho 2 để đếm số lần, rồi chia cho 5 để đếm số lần. Cộng vào hai biến tổng. Không cần phân tích các thừa số nguyên tố khác.

P có thể dài đến hàng triệu chữ số, nhưng hai tổng số mũ vẫn nhỏ và dễ lưu. Tính chất chia hết cho phép tránh hoàn toàn việc dựng tích.

Ví dụ để tự kiểm tra

Với (10,25,4): 10 góp (1 thừa số 2,1 thừa số 5); 25 góp (0,2); 4 góp (2,0). Hai tổng đều bằng 3, nên có 3 chữ số 0. Thật vậy 10×25×4=1000.

CT1-A / CHỨNG MINH VÀ CÀI ĐẶT

Chữ số 0 tận cùng

Vì sao thuật toán đúng?

Gọi c_2 và c_5 là số mũ 2 và 5 trong tích. Ghép được đúng $\min(c_2,c_5)$ cặp (2,5), tức có thể tách đúng số đó thừa số 10. Sau khi tách, ít nhất một loại thừa số đã hết nên không thể tách thêm 10. Các vòng lặp cộng chính xác c_2,c_5, do đó kết quả đúng.

Mã giả

```
count2 = 0; count5 = 0
cho mỗi x trong dãy:
    trong khi x mod 2 = 0:
        count2 += 1
        x /= 2
    trong khi x mod 5 = 0:
        count5 += 1
        x /= 5
in min(count2,count5)
```

Độ phức tạp

$O(\Sigma \log a_i)$ thời gian theo cận trên và O(1) bộ nhớ phụ.

Các lỗi dễ gặp

- Đầu vào chỉ có số dương. Nếu cho x=0, vòng chia sẽ không kết thúc; không tự mở rộng đề sang trường hợp đó.
- Không đếm số chữ số 0 của từng a_i rồi cộng: 2×5 tạo thêm một số 0 dù cả hai thừa số đều chưa có số 0.
- Không dùng logarit để đếm chữ số 0; vấn đề là chia hết, không phải số chữ số của tích.

Lời giải subtask duy nhất - 100 điểm

Bài chỉ có một nhóm điểm. Với từng số, đếm số lần chia hết cho 2 và 5, cộng vào hai biến tổng. Đáp án là tổng nhỏ hơn, vì mỗi thừa số 10 cần một cặp 2 và 5. Không tính tích cả dãy.

$O(\Sigma \log a_i)$ thời gian, O(1) bộ nhớ phụ.

CT1-B / KHÁM PHÁ LỜI GIẢI

B. Buổi trà thư giãn

Quy hoạch động đoạn | Tối ưu bộ nhớ

ĐIỂM MẤU CHỐT

Sau một số lượt, phần chưa uống luôn là một đoạn liên tiếp; chỉ hai đầu của đoạn có thể được chọn.

Từ quan sát đến thuật toán

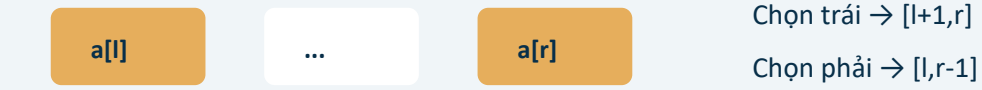
Gọi F(l,r) là tổng điểm lớn nhất còn có thể nhận khi đoạn chưa uống là [l,r]. Độ dài còn lại là r-l+1, nên lượt hiện tại được suy ra: day = n-(r-l+1)+1. Không cần thêm chiều day.

Nếu chọn trái, nhận day×a[l] rồi còn [l+1,r]. Nếu chọn phải, nhận day×a[r] rồi còn [l,r-1]. Vì vậy F(l,r)=max(day×a[l]+F(l+1,r), day×a[r]+F(l,r-1)). Cơ sở F(i,i)=n×a[i].

Tính theo độ dài đoạn tăng dần. Với n=5000, bảng n^2 số 64 bit cần khoảng 200 MB. Chỉ giữ một hàng dp[l], dùng dp[l] và dp[l+1] của độ dài trước; cập nhật l tăng dần để không phá dp[l+1].

Ví dụ để tự kiểm tra

Với (2,1,3), ở trạng thái [1,3], day=1. Nếu uống trái trước rồi uống 1,3, tổng là 1×2+2×1+3×3=13. Một mẹo tham lam “luôn lấy đầu nhỏ hơn” không đúng tổng quát: với (2,3,5,1,4), mẹo ấy cho 49 nhưng tối ưu là 50.



CT1-B / CHỨNG MINH VÀ CÀI ĐẶT

Buổi trà thư giãn

Vì sao thuật toán đúng?

Mọi cách uống hợp lệ bắt đầu bằng đúng một trong hai đầu. Sau lựa chọn đó, bài toán còn lại vẫn có cùng cấu trúc trên đoạn ngắn hơn. Giả sử các đoạn ngắn hơn đã được tính tối ưu; hai biểu thức là điểm tốt nhất khi chọn từng đầu. Lấy max xét đủ mọi khả năng và đạt được giá trị lớn nhất. Quy nạp theo độ dài chứng minh công thức và thứ tự tính.

Mã giả

```
cho i = 1..n: dp[i] = n*a[i]
cho len = 2..n:
    day = n-len+1
    cho l = 1..n-len+1 theo thứ tự tăng:
        r = l+len-1
        left = day*a[l] + dp[l+1]
        right = day*a[r] + dp[l]
        dp[l] = max(left,right)
in dp[1]
```

Độ phức tạp

$O(n^2)$ thời gian; $O(n)$ bộ nhớ với mảng cuốn. Dùng 64 bit: đáp án có thể vượt $2^{31}-1$.

Các lỗi dễ gặp

- Không nhân a[i] với số phần tử còn lại; hệ số là số thứ tự lượt uống.
- Trong mảng cuốn phải chạy l tăng, để dp[l+1] còn là trạng thái cũ.
- Đừng chọn đầu lớn hoặc nhỏ theo tham lam khi chưa chứng minh; lựa chọn hiện tại làm thay đổi các đầu về sau.

CT1-B / LỜI GIẢI TỪNG SUBTASK

Buổi trà thư giãn

Subtask	Điểm	Giới hạn bổ sung
1	20	n ≤ 20
2	30	n ≤ 500
3	50	Không có giới hạn bổ sung

Subtask 1 - 20 điểm

DFS trên đoạn chưa uống [l,r]. Mỗi bước chọn trái hoặc phải và cộng day×giá trị; tới đoạn rỗng thì cập nhật tổng lớn nhất. Duyệt đủ hai lựa chọn ở mỗi bước vì tham lam theo giá trị hai đầu không luôn đúng.

Độ phức tạp: O(2^n) thời gian, O(n) ngăn xếp.

```
DFS(l,r,day,total): nếu l>r thì cập nhật đáp án
    thử DFS(l+1,r,day+1,total+day*a[l])
    thử DFS(l,r-1,day+1,total+day*a[r])
```

Subtask 2 - 30 điểm

DP đoạn hai chiều. F(l,r) là điểm tốt nhất còn nhận với đoạn [l,r]; lượt hiện tại day=n-(r-l+1)+1. Chọn trái hoặc phải theo công thức lời giải đầy đủ, tính độ dài tăng dần. Bảng 500² số 64 bit chỉ khoảng 2 MB.

Độ phức tạp: $O(n^2)$ thời gian và $O(n^2)$ bộ nhớ.

```
F(i,i)=n*a[i]
F(l,r)=max(day*a[l]+F(l+1,r),
           day*a[r]+F(l,r-1))
```

Subtask 3 - 50 điểm

Giữ cùng chuyển trạng thái nhưng nén bộ nhớ xuống một mảng. Với mỗi độ dài, duyệt l tăng để dp[l+1] vẫn thuộc độ dài trước. Tránh bảng n^2 khi n=5000; tổng điểm phải dùng 64 bit.

Độ phức tạp: $O(n^2)$ thời gian, O(n) bộ nhớ.

CT1-C / KHÁM PHÁ LỜI GIẢI

C. Vượt lũ

Đường đi minimax | Quy hoạch động trên lưới | Tìm kiếm đáp án

ĐIỂM MẤU CHỐT

Đường đi tối ưu hóa ô cao nhất, không tối ưu tổng; hướng đi lên bị cấm.

Từ quan sát đến thuật toán

Cách dễ kiểm chứng: đoán một mức H và chỉ cho đi qua ô có độ ngập ≤H. BFS với đúng ba hướng trái, phải, xuống trả lời có đến đích hay không. Nếu đi được ở H thì mọi mức lớn hơn cũng đi được: có thể nhị phân trên các giá trị ô đã sắp xếp.

Có lời giải tuyến tính theo từng hàng. Gọi prev[j] là mức ngập nhỏ nhất để đến cột j của hàng trước. Ở hàng hiện tại, đường đi vào tại một cột rồi di chuyển ngang. Một đoạn ngang có thể bỏ các lần quay đầu; chỉ còn đi thẳng sang trái hoặc sang phải.

Quét trái sang phải: L[j]=max(a[i][j], min(prev[j],L[j-1])). Hai cách đến từ trên hoặc từ trái. Quét phải sang trái tương tự: R[j]=max(a[i][j], min(prev[j],R[j+1])). Đặt cur[j]=min(L[j],R[j]). Khởi tạo một hàng ảo có prev[1]=0 và các cột khác vô cùng, để chỉ được xuất phát tại (1,1).

Ví dụ để tự kiểm tra

Lưới hai hàng: [0,5,2] và [4,1,0]. Đường đi xuống ngay ở cột 1 rồi sang phải có mức lớn nhất 4; đi qua ô 5 cho mức ít nhất 5. Đáp án 4. “Đi xuống rồi sang phải” đúng ở ví dụ này nhưng không phải quy tắc cho mọi lưới.

ĐƯỜNG CÓ MỨC NGẬP 4

0	5	2
4	1	0

Chỉ đi trái, phải, xuống.

Giá trị đường đi = max của các ô đã qua.

CT1-C / CHỨNG MINH VÀ CÀI ĐẶT

Vượt lũ

Vì sao thuật toán đúng?

Do không được đi lên, sau khi vào một hàng thì mọi ô đi qua trong hàng đó nằm trên một đoạn ngang, cho đến lúc đi xuống. Bỏ quay đầu không làm tăng mức ngập. Với đích ở cột j, điểm vào hàng nằm bên trái hoặc bên phải j. Phép quét L xét tối ưu mọi điểm vào ở bên trái, R xét mọi điểm vào ở bên phải. Lấy min bao quát mọi đường đi; lấy max khi nối một ô là đúng phép tính mức ngập lớn nhất. Quy nạp qua các hàng hoàn tất chứng minh.

Mã giả

```
prev[1] = 0; prev[2..m] = INF
cho mỗi hàng i từ 1 đến n:
    L[0] = INF
    cho j = 1..m:
        L[j] = max(a[i][j], min(prev[j],L[j-1]))
    R[m+1] = INF
    cho j = m..1:
        R[j] = max(a[i][j], min(prev[j],R[j+1]))
    cho j = 1..m: cur[j] = min(L[j],R[j])
    prev = cur
in prev[m]
```

Độ phức tạp

Quét hai chiều: O(nm) thời gian, O(m) bộ nhớ nếu đọc từng hàng. Nhị phân+BFS: O(nm log(nm)) thời gian và O(nm) bộ nhớ.

Các lỗi dễ gặp

- Cập nhật bằng max, không phải cộng: mục tiêu là ô lớn nhất trên đường.
- BFS/Dijkstra không được thêm cạnh đi lên. Chỉ đi phải và xuống cũng bỏ sót đường hợp lệ.
- Quét R phải dùng prev của hàng cũ, không dùng cur vừa cập nhật; đặt INF lớn hơn 10^9.

CT1-C / LỜI GIẢI TỪNG SUBTASK

Vượt lũ

Subtask	Điểm	Giới hạn bổ sung
1	30	n,m ≤ 5
2	30	Mọi d_{ij} bằng 0 hoặc 1
3	40	Không có giới hạn bổ sung

Subtask 1 - 30 điểm

Không cần duyệt các đường có vòng lặp: bỏ vòng không làm tăng mức ngập. Trong mỗi hàng, chọn cột sẽ đi xuống hàng kế rồi đi ngang thẳng tới cột đó. Thử mọi cột đi xuống của n-1 hàng; hàng cuối đi ngang tới cột m. Lấy mức ngập nhỏ nhất trên các cách.

Độ phức tạp: O(nm^n) theo cách quét lại từng đường, O(nm) bộ nhớ; n,m≤5.

```
DFS(hàng,cột_vào,mức):
    thử cột_ra=1..m, quét đoạn ngang rồi đi xuống
    ở hàng cuối, quét tới m và cập nhật đáp án
```

Subtask 2 - 30 điểm

Đáp án chỉ có thể là 0 hoặc 1. BFS từ (1,1) qua các ô bằng 0, dùng đúng ba hướng trái/phải/xuống. Nếu tới đích thì trả 0; nếu không thì trả 1, vì khi cho phép ô 1 luôn có ít nhất đường sang phải rồi đi xuống tới đích.

Độ phức tạp: O(nm) thời gian, O(nm) bộ nhớ.

```
BFS chỉ trên các ô có giá trị 0
nếu tới (n,m): in 0; ngược lại: in 1
```

Subtask 3 - 40 điểm

Có thể nhị phân ngưỡng rồi BFS vì tính khả thi đơn điệu. Để đạt tuyến tính, dùng hai lượt quét mỗi hàng như lời giải đầy đủ: L xét mọi điểm vào ở bên trái, R xét mọi điểm vào ở bên phải; lấy min hai giá trị. Cả hai lượt dùng prev của hàng cũ.

Độ phức tạp: Quét hàng: O(nm) thời gian, O(m) bộ nhớ phụ. Nhị phân+BFS: O(nm log(nm)).

CT1-D / KHÁM PHÁ LỜI GIẢI

D. Lá

Cây | DFS có hoàn tác | Duy trì top-k

ĐIỂM MẤU CHỐT

Chỉ duy trì k giá trị lớn nhất trên đường hiện tại; xóa đúng giá trị khi DFS quay lui.

Từ quan sát đến thuật toán

Duyệt cây từ gốc xuống. Tại một đỉnh, tập giá trị đang giữ phải đúng bằng đường từ gốc tới đỉnh đó. Khi vào đỉnh, thêm q[u]; khi rời đỉnh, xóa q[u]. Chỉ tại lá mới dùng tổng để cập nhật đáp án.

Dùng hai tập có thứ tự: hi chứa min(k,số đỉnh trên đường) giá trị lớn nhất; lo chứa phần còn lại. Giữ sumHi là tổng hi. Bất biến: mọi phần tử hi lớn hơn mọi phần tử lo. Có thể dùng cặp (q[u],u) làm khóa để cài đặt không phụ thuộc tính phân biệt.

Thêm x vào hi; nếu hi có quá k phần tử thì chuyển phần tử nhỏ nhất của hi xuống lo. Khi xóa, nếu x thuộc hi thì xóa khỏi hi và bù bằng phần tử lớn nhất của lo nếu có; nếu x thuộc lo thì chỉ xóa khỏi lo. Mỗi bước chỉ cần số hằng thao tác O(log n).

Ví dụ để tự kiểm tra

Trong cây ví dụ, đường 1→3 có giá trị (5,9); top 2 cho tổng 14. Đường 1→2→4 có (5,2,8), top 2 cho tổng 13. Đường 1→2→5 cho tổng 7. Vì vậy chọn lá 3 và trả 14, dù đường này ngắn hơn đường tới lá 4.

CT1-D / CHỨNG MINH VÀ CÀI ĐẶT

Lá

Vì sao thuật toán đúng?

Các thao tác thêm/xóa giữ hi có đúng số phần tử cần thiết và mọi phần tử hi đều lớn hơn lo, nên sumHi chính là tổng top-k của đường hiện tại. DFS thêm khi vào và xóa khi ra giữ tập đúng bằng đường gốc-đỉnh. Mỗi lá được ghé đúng một lần, vì vậy max của các sumHi tại lá chính là đáp án.

Mã giả

```
hi,lo = hai tập rỗng; sumHi = 0; ans = 0
stack = [(gốc, VÀO)]
trong khi stack không rỗng:
    lấy (u,event) khỏi stack
    nếu event = RA:
        xóa q[u], cân bằng hi và lo
        tiếp tục
    thêm q[u], cân bằng hi và lo
    nếu u là lá: ans = max(ans,sumHi)
    đẩy (u,RA)
    đẩy các (con của u,VÀO)
in ans
```

Độ phức tạp

O(n log n) thời gian và O(n) bộ nhớ, gồm cây, hai tập và ngăn xếp duyệt.

Các lỗi dễ gặp

- Tìm gốc bằng p[u]=0; không mặc định gốc là 1.
- n tới 300000: dùng DFS lặp hoặc bảo đảm ngăn xếp đủ; đệ quy sâu có thể tràn stack.
- Hoàn tác sau toàn bộ con của một đỉnh, không xóa trước khi đi xuống. Dùng long long cho tổng, tối đa 3×10^{14}.

CT1-D / LỜI GIẢI TỪNG SUBTASK

Lá

Subtask	Điểm	Giới hạn bổ sung
1	30	n ≤ 2000
2	20	k = 1
3	50	Không có giới hạn bổ sung

Subtask 1 - 30 điểm

Với từng lá, lần lượt đi theo cha về gốc để thu tất cả giá trị trên đường. Sắp chúng giảm dần rồi cộng min(k,độ dài đường) phần tử đầu. Thử mọi lá và lấy tổng lớn nhất. Phải tìm gốc bằng p[u]=0, không mặc định đỉnh 1.

Độ phức tạp: O(n² log n) thời gian theo cận trên, O(n) bộ nhớ phụ.

```
với từng lá u: lấy dãy giá trị khi lần cha về gốc
    sắp giảm, cộng tối đa k phần tử đầu
đáp án là tổng lớn nhất
```

Subtask 2 - 20 điểm

Chỉ cần giá trị lớn nhất trên mỗi đường. DFS truyền bestPath[v]=max(bestPath[u],q[v]) từ cha u xuống con v. Tại lá, dùng bestPath để cập nhật đáp án. Dùng ngăn xếp lặp nếu cây sâu để tránh tràn stack.

Độ phức tạp: O(n) thời gian, O(n) bộ nhớ gồm cây và ngăn xếp.

```
bestPath[gốc]=q[gốc]
cho mỗi cạnh cha u → con v:
    bestPath[v]=max(bestPath[u],q[v])
lấy max của bestPath tại các lá
```

Subtask 3 - 50 điểm

DFS có hoàn tác, duy trì hi chứa top-k và lo chứa phần còn lại của đường hiện tại. Khi thêm hoặc xóa một giá trị, cân bằng hai tập và cập nhật sumHi. Chỉ đọc sumHi tại lá; sau khi xử lý xong các con mới xóa giá trị của đỉnh.

Độ phức tạp: O(n log n) thời gian, O(n) bộ nhớ. Tổng dùng 64 bit.