SIROCCO CUP 2026 / KC2

Không chuyên - Đề 2

Thứ Tư, 04/11/2026 · Online

180 phút · 4 bài · 400 điểm

Đoạn liên tiếp, DP và truy vấn số học

Bài	Tên bài	Tag
A	Dãy cây	Hai con trỏ, Đoạn liên tiếp
B	Tăng chữ số	Quy hoạch động, Tiền xử lý
C	Giải phóng bộ nhớ	Tham lam, Sắp xếp, Tổng tiền tố
D	Cắt dãy	Số học, Tham lam, Nhảy nhị phân

Problemset trực tuyến
https://codeforces.com/gym/729105/problems

KC2-A / KHÁM PHÁ LỜI GIẢI

A. Dãy cây

Hai con trỏ | Đoạn liên tiếp

ĐIỂM MẤU CHỐT

Một cặp chiều cao không hợp lệ chia dãy thành hai phần không thể nằm chung trong đáp án.

Từ quan sát đến thuật toán

Đoạn hợp lệ không được đi qua vị trí h[i] không chia hết cho h[i+1]. Vì vậy, khi xét đầu phải r và gặp một vị trí như thế, bỏ toàn bộ cửa sổ cũ, đặt l=r và tổng bằng 0 trước khi thêm a[r].

Thêm a[r]. Khi tổng vượt k, bỏ dần a[l] và tăng l. Do mọi a[i] dương, bỏ bên trái chỉ làm tổng giảm; mở rộng bên phải chỉ làm tổng tăng. Cập nhật đáp án bằng r-l+1, kể cả khi cửa sổ rỗng.

Kiểm tra bằng ví dụ

Test đầu chọn [1,3]: tổng bằng 9 và 4 chia hết cho 4, 4 chia hết cho 2. Test sau, phần tử duy nhất vượt k.

KC2-A / CHỨNG MINH VÀ CÀI ĐẶT

Dãy cây

Vì sao thuật toán đúng?

Sau khi xử lý r, cửa sổ không đi qua cặp chiều cao hỏng và có tổng ≤ k. Những vị trí trái bị loại vì cặp chiều cao hỏng không thể dùng lại ở đầu phải r. Những vị trí bị loại vì tổng quá lớn cũng không thể tạo đoạn hợp lệ kết thúc tại r. Do đó l là đầu trái nhỏ nhất còn có thể dùng, và cửa sổ là đoạn dài nhất kết thúc tại r. Lấy lớn nhất qua mọi r cho đáp án toàn cục.

Mã giả

```
l=1; total=0; ans=0
for r=1..n:
    if r>1 and h[r-1] % h[r] != 0:
        l=r; total=0
    total += a[r]
    while total > k:
        total -= a[l]; l += 1
    ans=max(ans,r-l+1)
print ans
```

Độ phức tạp

O(n) thời gian mỗi test; O(1) bộ nhớ phụ ngoài hai dãy đầu vào.

Các lỗi dễ gặp

- Không đảo điều kiện chia hết thành h[i+1] % h[i].
- Khi cắt tại r phải xóa tổng cũ trước khi thêm a[r].
- Nếu a[r] > k, vòng while có thể đưa l đến r+1; độ dài lúc đó là 0.

KC2-A / LỜI GIẢI TỪNG SUBTASK

Dãy cây

Subtask	Điểm	Giới hạn bổ sung
1	30	Tổng n ≤ 500.
2	20	Mọi h[i] = 1.
3	50	Không có giới hạn bổ sung.

Subtask 1 - 30 điểm

Duyệt mọi đầu trái, mở rộng đầu phải và cộng tổng. Dừng khi gặp cặp chiều cao sai hoặc tổng vượt k. Thử hết các đoạn còn lại.

Độ phức tạp: $O(\Sigma n^2)$ thời gian, O(1) bộ nhớ phụ.

Subtask 2 - 20 điểm

Mọi cặp chiều cao đều hợp lệ. Chỉ cần cửa sổ tổng ≤ k với hai con trỏ.

Độ phức tạp: $O(\Sigma n)$ thời gian, O(1) bộ nhớ phụ.

Subtask 3 - 50 điểm

Thực hiện hai con trỏ kết hợp cắt cửa sổ tại cặp chiều cao không hợp lệ như mã giả.

Độ phức tạp: $O(\Sigma n)$ thời gian, O(1) bộ nhớ phụ.

KC2-B / KHÁM PHÁ LỜI GIẢI

B. Tăng chữ số

Quy hoạch động | Tiền xử lý

ĐIỂM MẤU CHỐT

Các chữ số ban đầu phát triển độc lập; chỉ cần biết một chữ số sinh ra bao nhiêu chữ số.

Từ quan sát đến thuật toán

Đặt f[s][d] là số chữ số sinh ra từ chữ số d sau s bước. Ban đầu f[0][d]=1. Với d<9, bước đầu chỉ đổi d thành d+1, nên f[s][d]=f[s-1][d+1].

Với d=9, bước đầu sinh hai chữ số 1 và 0, nên f[s][9]=f[s-1][1]+f[s-1][0]. Tiền xử lý đến m lớn nhất trong toàn bộ input. Đáp án của n là tổng f[m][d] trên từng chữ số của n, kể cả chữ số 0 nằm bên trong n.

Kiểm tra bằng ví dụ

19 → 210 có 3 chữ số; 9 → 10 → 21 có 2 chữ số; 88 → 99 → 1010 có 4 chữ số.

KC2-B / CHỨNG MINH VÀ CÀI ĐẶT

Tăng chữ số

Vì sao thuật toán đúng?

Một phép biến đổi áp dụng riêng rẽ cho từng chữ số, không có phép nhớ sang chữ số bên cạnh. Vì thế, số chữ số của chuỗi ghép bằng tổng số chữ số của các nhánh. Hai công thức chuyển xét hết d<9 và d=9, đúng ở s=0 và bảo toàn tính đúng bằng quy nạp theo s. Cộng các nhánh ban đầu cho đúng độ dài cần tìm.

Mã giả

```
MOD=1000000007
for d=0..9: f[0][d]=1
for s=1..M:
    for d=0..8: f[s][d]=f[s-1][d+1]
    f[s][9]=(f[s-1][1]+f[s-1][0]) % MOD
for each (n,m):
    ans=0
    for digit d in decimal(n):
        ans=(ans+f[m][d]) % MOD
    print ans
```

Độ phức tạp

O(10M + tổng số chữ số đầu vào) thời gian; O(10M) bộ nhớ, M là m lớn nhất.

Các lỗi dễ gặp

- Thao tác là thay từng chữ số, không phải cộng 1 cho toàn bộ số.
- Các chữ số mới sinh ra chỉ được biến đổi ở bước tiếp theo.
- Không dựng chuỗi ở full test: độ dài có thể tăng rất lớn.

KC2-B / LỜI GIẢI TỪNG SUBTASK

Tăng chữ số

Subtask	Điểm	Giới hạn bổ sung
1	20	t ≤ 100 và m ≤ 10.
2	30	t ≤ 10 và m ≤ 2000.
3	50	Không có giới hạn bổ sung.

Subtask 1 - 20 điểm

Giữ chuỗi chữ số. Mỗi bước dựng chuỗi mới bằng cách nối biểu diễn d+1 của từng chữ số trong chuỗi cũ.

Độ phức tạp: O(t×D×2^m) là cận trên, D ≤ 10; bộ nhớ O(D×2^m).

Subtask 2 - 30 điểm

Với từng test, duy trì cnt[0..9]. Mỗi bước chuyển cnt[d] sang d+1 nếu d<9; cnt[9] sang cả 0 và 1. Dùng mảng mới, lấy modulo.

Độ phức tạp: O(10Σm) thời gian, O(10) bộ nhớ.

Subtask 3 - 50 điểm

Tiền xử lý bảng f một lần cho tất cả test, sau đó cộng kết quả theo các chữ số ban đầu.

Độ phức tạp: O(10M + ΣD) thời gian, O(10M) bộ nhớ.

KC2-C / KHÁM PHÁ LỜI GIẢI

C. Giải phóng bộ nhớ

Tham lam | Sắp xếp | Tổng tiền tố

ĐIỂM MẤU CHỐT

Chi phí chỉ có hai loại: cố định số món mỗi loại thì luôn chọn các món có a lớn nhất.

Từ quan sát đến thuật toán

Chia ứng dụng thành nhóm chi phí 1 và nhóm chi phí 2, sắp a giảm dần trong từng nhóm. Lập các tổng tiền tố P và Q, gồm cả P[0]=Q[0]=0.

Duyệt số x ứng dụng chọn từ nhóm 1. Cần Q[y] ≥ m-P[x]. Vì Q tăng nghiêm ngặt, tìm y nhỏ nhất bằng lower_bound, rồi cập nhật x+2y. Nếu phần còn thiếu không dương, dùng y=0. Nếu không có cặp nào đủ bộ nhớ, in -1.

Kiểm tra bằng ví dụ

Test đầu xóa ứng dụng 2 và 5 được 7 đơn vị, mất 2 điểm. Test sau chỉ có 2 đơn vị để giải phóng.

KC2-C / CHỨNG MINH VÀ CÀI ĐẶT

Giải phóng bộ nhớ

Vì sao thuật toán đúng?

Với số lượng x,y cố định, thay một ứng dụng đã chọn bằng ứng dụng chưa chọn có cùng b và a lớn hơn không tăng chi phí và không giảm bộ nhớ giải phóng. Vì vậy tồn tại nghiệm tối ưu chỉ chọn tiền tố của mỗi nhóm đã sắp xếp. Thuật toán duyệt mọi x và tìm y nhỏ nhất đủ nhu cầu, nên xét được một nghiệm không đắt hơn nghiệm tối ưu; mọi phương án được xét đều hợp lệ.

Mã giả

```
U=sorted(a[i] with b[i]=1, descending)
V=sorted(a[i] with b[i]=2, descending)
P=prefix_sums(U); Q=prefix_sums(V)
ans=INF
for x=0..len(U):
    need=max(0,m-P[x])
    y=lower_bound(Q,need)
    if y<=len(V): ans=min(ans,x+2*y)
print (-1 if ans=INF else ans)
```

Độ phức tạp

O(n log n) thời gian và O(n) bộ nhớ mỗi test; tổng tiền tố dùng số nguyên 64 bit.

Các lỗi dễ gặp

- Không tham lam theo a/b: tỷ lệ tốt nhất chưa chắc cho chi phí nguyên nhỏ nhất.
- Phải duyệt cả x=0 và cho phép y=0.
- Tổng a có thể tới 2×10^14, vượt số nguyên 32 bit.

KC2-C / LỜI GIẢI TỪNG SUBTASK

Giải phóng bộ nhớ

Subtask	Điểm	Giới hạn bổ sung
1	20	t ≤ 5 và n ≤ 18.
2	30	Tổng n ≤ 2000.
3	50	Không có giới hạn bổ sung.

Subtask 1 - 20 điểm

Duyệt mọi tập con; tính tổng a và tổng b. Trong các tập có tổng a ≥ m, lấy tổng b nhỏ nhất.

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

Subtask 2 - 30 điểm

DP theo chi phí từ 0 đến 2n: dp[c] là bộ nhớ tối đa giải phóng với đúng c điểm. Khởi tạo dp[0]=0, còn lại âm vô cùng; duyệt c giảm khi thêm từng ứng dụng.

Độ phức tạp: O(Σn²) thời gian, O(n) bộ nhớ.

Subtask 3 - 50 điểm

Chia theo b, chọn các tiền tố tốt nhất và tìm nhị phân như lời giải đầy đủ.

Độ phức tạp: O(Σn log n) thời gian, O(n) bộ nhớ.

KC2-D / KHÁM PHÁ LỜI GIẢI

D. Cắt dãy

Số học | Tham lam | Nhảy nhị phân

ĐIỂM MẤU CHỐT

Tích bằng BCNN khi và chỉ khi không có số nguyên tố xuất hiện trong hai phần tử khác nhau.

Từ quan sát đến thuật toán

Số mũ của một nguyên tố trong tích là tổng số mũ, còn trong BCNN là số mũ lớn nhất. Hai giá trị chỉ bằng nhau nếu nhiều nhất một phần tử chứa nguyên tố đó; một phần tử như 4 vẫn tự nó hợp lệ.

Gọi nxt[i] là vị trí đầu tiên phải cắt khi bắt đầu tại i: [i,nxt[i]-1] dài nhất hợp lệ. Quét i từ phải sang trái, lưu last[p] là vị trí gần nhất bên phải chứa p. Khi đó nxt[i]=min(nxt[i+1], mọi last[p] với p chia a[i]); sau đó gán last[p]=i. Chỉ lấy các thừa số nguyên tố phân biệt.

Tham lam lấy đoạn dài nhất rồi bắt đầu đoạn tiếp theo tại nxt[i]. Lập up[j][i] bằng 2^j lần áp dụng nxt. Trả lời [l,r] bằng cách nhảy nhiều nhất mà vị trí mới vẫn ≤ r; số lần nhảy cộng 1 là đáp án. Đặt nxt[n+1]=n+1.

Kiểm tra bằng ví dụ

Truy vấn đầu có thể chia [2,3] | [10,7] | [5,14]. Mỗi đoạn không có hai phần tử cùng chứa một thừa số nguyên tố.

KC2-D / CHỨNG MINH VÀ CÀI ĐẶT

Cắt dãy

Vì sao thuật toán đúng?

Đẳng thức tích và BCNN tương đương điều kiện nguyên tố đã nêu. Một xung đột đầu tiên trong [i,..] hoặc đã nằm trong hậu tố bắt đầu i+1, hoặc do một nguyên tố của a[i] lặp ở last[p]; công thức nxt xét hết hai khả năng. Mọi đoạn đầu của một cách chia hợp lệ kết thúc không muộn hơn nxt[i]-1. Kéo dài đoạn đầu đến đó chỉ bỏ đi một tiền tố của các đoạn phía sau, không tăng số đoạn còn cần. Do đó tham lam là tối ưu; bảng nhảy chỉ tăng tốc cùng chuỗi quyết định.

Mã giả

```
nxt[n+1]=n+1; last[p]=n+1 for every prime p
for i=n..1:
    nxt[i]=nxt[i+1]
    primes=distinct_prime_factors(a[i])
    for p in primes: nxt[i]=min(nxt[i],last[p])
    for p in primes: last[p]=i
up[0]=nxt
for j=1..LOG:
    for i=1..n+1: up[j][i]=up[j-1][up[j-1][i]]
for (l,r):
    pos=l; ans=1
    for j=LOG..0:
        if up[j][pos]<=r:
            pos=up[j][pos]; ans+=2^j
    print ans
```

Độ phức tạp

Với sàng SPF: O(V log log V + n log V + (n+q) log n) thời gian, O(V+n log n) bộ nhớ; V=max a[i].

Các lỗi dễ gặp

- Không yêu cầu mỗi a[i] là số không chứa bình phương: [4,3] vẫn hợp lệ.
- Chỉ cập nhật last sau khi đã xét hết thừa số của a[i].
- Không nhân trực tiếp cả đoạn; tích có thể vượt mọi kiểu số thông thường.

KC2-D / LỜI GIẢI TỪNG SUBTASK

Cắt dãy

Subtask	Điểm	Giới hạn bổ sung
1	20	n,q ≤ 150.
2	30	n,q ≤ 2000.
3	50	Không có giới hạn bổ sung.

Subtask 1 - 20 điểm

Duyệt các điểm kết thúc đoạn đầu. Một đoạn hợp lệ khi mọi cặp phần tử nguyên tố cùng nhau. Tiền xử lý hợp lệ cho mọi đoạn rồi DP số đoạn cho từng truy vấn.

Độ phức tạp: $O(n^3 \log V + qn^2)$ thời gian; $O(n^2)$ bộ nhớ.

Subtask 2 - 30 điểm

Với từng truy vấn, quét từ l đến r, giữ tập nguyên tố đã dùng trong đoạn hiện tại. Khi a[i] có nguyên tố trùng, tăng số đoạn và bắt đầu lại từ i. Có thể dùng timestamp để xóa tập nhanh.

Độ phức tạp: $O(V \log \log V + qn \log V)$ thời gian, $O(V+n)$ bộ nhớ.

Subtask 3 - 50 điểm

Tính nxt bằng lần xuất hiện gần nhất của nguyên tố và dùng bảng nhảy nhị phân trả lời mọi truy vấn.

Độ phức tạp: $O(V \log \log V + n \log V + (n+q) \log n)$ thời gian; $O(V+n \log n)$ bộ nhớ.