SIROCCO CUP 2026 / CT3

Chuyên - Đề 3

Thứ Bảy, 07/11/2026 · Offline tại phòng sinh hoạt câu lạc bộ

180 phút · 4 bài · 400 điểm

Cửa sổ trượt, tham lam và xâu động

Bài	Tên bài	Tag
A	Panda 2	Tham lam, Đếm, Mô phỏng
B	Đếm số đoạn con	Hai con trỏ, Cửa sổ trượt, Đếm tần suất
C	Chia kho báu	Tham lam, Sắp xếp, Tổng tiền tố, Tổng điều hòa
D	Đếm cặp hậu tố	Trie, Xâu đảo, Đếm cặp động

Problemset trực tuyến
https://codeforces.com/gym/728950/problems

CT3-A / KHÁM PHÁ LỜI GIẢI

A. Panda 2

Tham lam | Đếm | Mô phỏng

ĐIỂM MẤU CHỐT

Phản hồi là tùy chọn: chỉ dùng cho bài đang trượt nhưng sẽ đỗ sau một lần thay đổi.

Từ quan sát đến thuật toán

Với từng bài báo, tính tổng ban đầu old. Tổng sau phản hồi new được tính trực tiếp: mỗi điểm dương giảm 1, mỗi điểm không dương tăng 1. Việc thay đổi của một bài không ảnh hưởng bài khác.

Đếm accepted là số bài đã có old≥k. Không cần phản hồi những bài này vì chúng đã đóng góp một bài đỗ; phản hồi còn có thể làm chúng trượt. Đếm improve là số bài có old<k nhưng new≥k.

Mỗi bài thuộc improve cần đúng một lượt phản hồi và cho thêm đúng một bài đỗ. Có tối đa b lượt nên đáp án = accepted+min(b,improve). Không cần knapsack: mọi lựa chọn có cùng chi phí và cùng lợi ích.

Ví dụ để tự kiểm tra

Với m=2,k=1,b=1: (1,1) đã đỗ, không chọn; (0,0) trượt nhưng thành (1,1), nên chọn để đỗ; (-1,0) thành (0,1) cũng có thể đỗ nhưng hết lượt; (-3,-3) vẫn trượt. Một bài đỗ sẵn cộng một bài cải thiện cho đáp án 2.

CT3-A / CHỨNG MINH VÀ CÀI ĐẶT

Panda 2

Vì sao thuật toán đúng?

Một bài đã đỗ đóng góp nhiều nhất 1, nên phản hồi nó không thể tăng đáp án. Một bài trượt không thể đỗ sau phản hồi cũng không tạo lợi ích. Chỉ các bài thuộc improve có thể tăng đáp án; mỗi bài cho thêm 1 và tốn 1 lượt. Bất kỳ phương án nào tăng nhiều nhất min(b,improve), và chọn đúng số bài như vậy đạt cận trên. Công thức là tối ưu.

Mã giả

```
accepted = 0; improve = 0
cho mỗi bài báo:
    old = 0; new = 0
    cho mỗi điểm x:
        old += x
        nếu x > 0: new += x-1
        ngược lại: new += x+1
    nếu old >= k: accepted += 1
    ngược lại nếu new >= k: improve += 1
in accepted + min(b,improve)
```

Độ phức tạp

O(nm) thời gian và O(1) bộ nhớ phụ.

Các lỗi dễ gặp

- Điểm 0 thuộc nhánh tăng 1, không thuộc nhánh giảm.
- Được chọn tối đa b bài, không bắt buộc chọn đúng b.
- Không chỉ nhìn tổng điểm để suy ra new: hai bài cùng tổng có thể có số điểm dương khác nhau.

Lời giải subtask duy nhất - 100 điểm

Bài chỉ có một nhóm điểm. Tính old và new cho từng bài báo. Đếm accepted là số đã đạt, improve là số đang trượt nhưng đạt sau phản hồi. Mỗi bài cải thiện tốn một lượt và thêm một bài đạt, nên đáp án accepted+min(b,improve).

O(nm) thời gian, O(1) bộ nhớ phụ.

CT3-B / KHÁM PHÁ LỜI GIẢI

B. Đếm số đoạn con

Hai con trỏ | Cửa sổ trượt | Đếm tần suất

ĐIỂM MẤU CHỐT

"Đúng k" khó giữ trực tiếp; lấy "không quá k" trừ "không quá k-1".

Từ quan sát đến thuật toán

Gọi F(K) là số đoạn con có không quá K giá trị phân biệt. Mỗi đoạn có đúng k được tính trong F(k) nhưng không trong F(k-1); các đoạn có ít hơn k bị trừ hết. Do đó đáp án = F(k)-F(k-1).

Để tính F(K), quét đầu phải r từ trái sang phải. Tăng tần suất a[r]. Nếu số loại vượt K, tăng đầu trái l và giảm tần suất đến khi cửa sổ [l,r] hợp lệ.

Lúc này, các đoạn kết thúc tại r và có đầu trái từ l đến r đều hợp lệ: có r-l+1 đoạn. Các đầu trái nhỏ hơn l đều không hợp lệ. Cộng số này cho từng r. Nếu K<0 trả 0; trong bài, K=0 tự cho tổng 0.

Ví dụ để tự kiểm tra

Với (1,2,1,2,3), F(2)=12 và F(1)=5, nên có 7 đoạn đúng hai loại. Khi r=4, cửa sổ dài 4 vẫn chỉ chứa hai loại; khi thêm 3 ở r=5, l phải dịch tới 4, còn cửa sổ (2,3).

CT3-B / CHỨNG MINH VÀ CÀI ĐẶT

Đếm số đoạn con

Vì sao thuật toán đúng?

Sau vòng thu hẹp, [l,r] có không quá K loại. Bỏ bớt phần đầu không làm tăng số loại nên mọi đầu trái lớn hơn l đều hợp lệ. Các vị trí đã bị l đi qua tại bước r hoặc trước đó từng cho quá K loại; khi r chỉ tăng, chúng không thể trở lại hợp lệ. Do đó r-l+1 đếm đúng mọi đoạn kết thúc ở r. Cộng theo r không trùng lặp, rồi hiệu hai hàm lọc đúng k loại.

Mã giả

```
AT_MOST(K):
    nếu K < 0: trả 0
    l = 1; distinct = 0; total = 0
    freq = bảng tần suất rỗng
    cho r = 1..n:
        tăng freq[a[r]]; nếu từ 0 lên 1: distinct++
        trong khi distinct > K:
            giảm freq[a[l]]
            nếu từ 1 xuống 0: distinct--
            l++
        total += r-l+1
    trả total

in AT_MOST(k)-AT_MOST(k-1)
```

Độ phức tạp

Dùng hash: O(n) kỳ vọng, O(n) bộ nhớ. Nén giá trị bằng sắp xếp: O(n log n) tổng thời gian, rồi hai lượt cửa sổ O(n), bộ nhớ O(n).

Các lỗi dễ gặp

- Đáp án có thể là n(n+1)/2, phải dùng long long.
- Reset freq, l và distinct khi gọi AT_MOST lần thứ hai.
- Chỉ giảm distinct khi tần suất về 0. Không dùng mảng kích thước 10^9 theo giá trị đầu vào.

CT3-B / LỜI GIẢI TỪNG SUBTASK

Đếm số đoạn con

Subtask	Điểm	Giới hạn bổ sung
1	30	n ≤ 500; tổng n trong file ≤ 2000
2	30	k = 1
3	40	Không có giới hạn bổ sung

Subtask 1 - 30 điểm

Cố định đầu trái l rồi tăng đầu phải r. Duy trì bảng tần suất và số giá trị phân biệt trong đoạn; mỗi khi số loại bằng k thì tăng đáp án. Nếu số loại đã vượt k thì dừng vòng r, vì mở rộng không thể làm giảm số loại.

Độ phức tạp: O(n²) thời gian, O(n) bộ nhớ sau nén giá trị.

```
cho l=1..n: xóa bảng tần suất; distinct=0
    cho r=l..n: thêm a[r], cập nhật distinct
        distinct=k: ans++; distinct>k: dừng
```

Subtask 2 - 30 điểm

Một đoạn có đúng một giá trị phân biệt phải nằm hoàn toàn trong một khối các số bằng nhau liên tiếp. Khối dài z có z(z+1)/2 đoạn con. Quét dãy để lấy độ dài từng khối rồi cộng công thức này.

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

```
chia dãy thành các khối bằng nhau liên tiếp
với độ dài z của mỗi khối: ans+=z*(z+1)/2
```

Subtask 3 - 40 điểm

Tính F(k)-F(k-1), trong đó F(K) đếm đoạn có không quá K loại bằng cửa sổ trượt. Sau khi thu hẹp đầu trái tới cửa sổ hợp lệ, mỗi đầu phải r đóng góp r-l+1 đoạn. Nén giá trị để dùng mảng tần suất ổn định.

Độ phức tạp: O(n log n) nếu nén bằng sắp xếp; hai lượt cửa sổ O(n). O(n) bộ nhớ.

CT3-C / KHÁM PHÁ LỜI GIẢI

C. Chia kho báu

Tham lam | Sắp xếp | Tổng tiền tố | Tổng điều hòa

ĐIỂM MẤU CHỐT

Mỗi hệ số 1,2,3,... có k chỗ: ghép số lớn nhất với các hệ số nhỏ nhất.

Từ quan sát đến thuật toán

Trong k dãy, có k vị trí hệ số 1, k vị trí hệ số 2, v.v. Vì mọi giá trị đều dương, nếu còn một vị trí hệ số nhỏ bỏ trống trong khi dùng một hệ số lớn hơn, chuyển một phần tử sang vị trí nhỏ sẽ giảm tổng. Vì thế tối ưu dùng n hệ số nhỏ nhất của các chỗ này.

Nếu x≥y nhưng x đang nhân với hệ số u>v của y, đổi chỗ cho chênh lệch mới-cũ = (xv+yu)-(xu+yv) = -(x-y)(u-v)≤0. Do đó sắp giá trị giảm dần và hệ số tăng dần là tối ưu.

Sau khi sắp a giảm, đáp án cho k là Σ a[i]×ceil(i/k). Các phần tử từ (j-1)k+1 đến min(jk,n) cùng hệ số j. Dùng tổng tiền tố P để cộng cả khối trong O(1). Với một k cần ceil(n/k) khối; cộng cho mọi k là O(n log n).

Ví dụ để tự kiểm tra

S=(3,1,2), sắp thành (3,2,1). k=1: hệ số (1,2,3), điểm 3+4+3=10. k=2: hệ số (1,1,2), điểm 3+2+2=7; có thể chia (3,1) và (2). k=3: mỗi số đứng đầu một dãy, điểm 6.

CT3-C / CHỨNG MINH VÀ CÀI ĐẶT

Chia kho báu

Vì sao thuật toán đúng?

Đổi chỗ loại được mọi cặp nghịch thế giữa giá trị và hệ số, nên cách ghép giảm-tăng đạt tổng nhỏ nhất trên n hệ số được chọn. Tính dương cho phép thay mọi chỗ hệ số lớn không cần thiết bằng chỗ nhỏ còn trống, nên bộ n hệ số nhỏ nhất là tối ưu. Các hệ số này thực hiện được bằng cách phân phối các số theo từng tầng vị trí của k dãy; không xuất hiện lỗ trước một vị trí được dùng. Công thức khối tính chính xác tổng ghép ấy.

Mã giả

```
sắp a[1..n] giảm dần
P[0] = 0
cho i = 1..n: P[i] = P[i-1]+a[i]
cho k = 1..n:
    ans = 0; coefficient = 1
    cho l = 1,1+k,1+2*k,... không vượt n:
        r = min(n,l+k-1)
        ans += coefficient*(P[r]-P[l-1])
        coefficient++
    in ans
```

Độ phức tạp

O(n log n) thời gian: sắp xếp và tổng số khối Σ ceil(n/k). O(n) bộ nhớ cho dãy và tổng tiền tố. Đáp án tối đa khoảng 2×10^{16}, cần 64 bit.

Các lỗi dễ gặp

- Không sắp tăng dần: số lớn phải nhận hệ số nhỏ.
- Các dãy được rỗng; không tự thêm điều kiện mỗi dãy đều phải có phần tử.
- Tổng tiền tố dùng 64 bit, và phép nhân hệ số×tổng khối cũng dùng 64 bit.

CT3-C / LỜI GIẢI TỪNG SUBTASK

Chia kho báu

Subtask	Điểm	Giới hạn bổ sung
1	30	n ≤ 8, t ≤ 20
2	30	Các phần tử trong từng test bằng nhau
3	40	Không có giới hạn bổ sung

Subtask 1 - 30 điểm

DP tập con. Tính g[mask] bằng cách xếp các phần tử của mask giảm dần vào một dãy. Đặt F[j][mask] là điểm nhỏ nhất chia mask thành j dãy không rỗng; thử mọi sub không rỗng làm dãy cuối. Vì k≤n và các giá trị dương, nếu còn dãy rỗng thì có thể tách phần tử cuối từ dãy dài ít nhất 2 sang đó để giảm điểm. Do đó xét đúng k dãy không rỗng là đủ.

Độ phức tạp: O(n×3^n + n log n×2^n) thời gian, O(n×2^n) bộ nhớ.

```
F[0][0]=0; các trạng thái khác=INF
F[j][mask]=min(g[sub]+F[j-1][mask bỏ sub])
    với mọi sub không rỗng của mask
ans[k]=F[k][toàn_bộ]
```

Subtask 2 - 30 điểm

Gọi giá trị chung là v, a=floor(n/k), r=n mod k. Các hệ số nhỏ nhất gồm k bản của 1,...,a và r bản của a+1. Vì các số bằng nhau nên không cần sắp xếp: đáp án là v×(k×a(a+1)/2+r×(a+1)). Tính độc lập cho mọi k.

Độ phức tạp: O(n) thời gian cho toàn bộ n đáp án, O(1) bộ nhớ phụ.

```
cho k=1..n: a=n/k; r=n%k
    ans[k]=v*(k*a*(a+1)/2+r*(a+1))
```

Subtask 3 - 40 điểm

Sắp các giá trị giảm dần, ghép với hệ số ceil(i/k). Dùng tổng tiền tố để cộng một khối k phần tử trong O(1). Với một k có ceil(n/k) khối; tổng số khối qua mọi k là O(n log n).

Độ phức tạp: O(n log n) thời gian, O(n) bộ nhớ; mọi tích và tổng dùng 64 bit.

CT3-D / KHÁM PHÁ LỜI GIẢI

D. Đếm cặp hậu tố

Trie | Xâu đảo | Đếm cặp động

ĐIỂM MẤU CHỐT

Đảo chuỗi biến quan hệ hậu tố thành tiền tố; một thay đổi chỉ tạo hoặc mất các cặp chứa chính từ đó.

Từ quan sát đến thuật toán

A là hậu tố của B khi và chỉ khi reverse(A) là tiền tố của reverse(B). Dùng trie chứa các từ đảo. Mỗi nút lưu terminal: có từ đang hoạt động kết thúc tại nút không; sub: số từ đang hoạt động kết thúc trong cả cây con của nút, kể cả chính nút.

Với từ đảo r kết thúc ở nút v, các từ ngắn hơn là tiền tố thực sự của r nằm ở tổ tiên thực sự của v. Cộng terminal trên đường nhưng bỏ v. Các từ dài hơn có r làm tiền tố nằm trong cây con v nhưng bỏ chính từ r.

Khi thêm một từ chưa tồn tại: trước cập nhật, số cặp mới = tổng terminal của tổ tiên thực sự + sub[v]. Khi xóa một từ đang tồn tại: trước cập nhật, số cặp mất = tổng terminal của tổ tiên thực sự + sub[v]-1. Sau đó mới tăng/giảm sub trên cả đường và bật/tắt terminal[v]. Các nút có thể giữ lại sau khi từ bị xóa.

Ví dụ để tự kiểm tra

Thêm a: 0 cặp. Thêm ba: cặp (a,ba), được 1. Thêm aba: thêm (a,aba) và (ba,aba), tổng 3. Xóa a: mất (a,ba) và (a,aba), còn (ba,aba), tổng 1. Sau khi đảo, a,ab,aba cùng nằm trên một nhánh trie.

CT3-D / CHỨNG MINH VÀ CÀI ĐẶT

Đếm cặp hậu tố

Vì sao thuật toán đúng?

Các cặp không chứa s không đổi khi thêm hoặc xóa s. Cặp chứa s chỉ có hai dạng: (t,s), t là hậu tố thực sự của s; hoặc (s,t), s là hậu tố thực sự của t. Đảo chuỗi cho hai tập lần lượt là các điểm kết thúc ở tổ tiên và ở hậu duệ. Hai tập rời nhau; terminal và sub đếm chính xác chúng. Công thức thay đổi ans vì vậy thêm/bớt đúng và đủ các cặp liên quan, còn cập nhật đường giữ đúng số từ hoạt động.

Mã giả

```
ans = 0
VỚI MỖI THAO TÁC (op,s):
    r = đảo(s)
    lấy/tạo đường nút v1,...,vm theo r
    v = vm
    shorter = tổng terminal[v1..v(m-1)]
    nếu op = thêm:
        ans += shorter + sub[v]  // từ s chưa có
        tăng sub của mỗi nút trên đường
        terminal[v] = 1
    nếu op = xóa:
        ans -= shorter + sub[v]-1
        giảm sub của mỗi nút trên đường
        terminal[v] = 0
    in ans
```

Độ phức tạp

O(L) thời gian với bảng chữ cái cố định và O(L) nút, L là tổng độ dài qua mọi thao tác. Trie mảng 26 con dùng khoảng 112 MB cho 10^6 nút; cần xét giới hạn bộ nhớ hoặc dùng biểu diễn cạnh thưa.

Các lỗi dễ gặp

- Không tính (s,s). Khi xóa, trừ 1 trong sub[v] vì từ s vẫn đang hoạt động trước cập nhật.
- Tính số cặp thay đổi trước khi tăng/giảm bộ đếm; thứ tự sai dễ lệch một cặp.
- ans dùng 64 bit. Không giảm sub của các nhánh ngoài đường của s, và không xóa nút chỉ vì terminal về 0.

CT3-D / LỜI GIẢI TỪNG SUBTASK

Đếm cặp hậu tố

Subtask	Điểm	Giới hạn bổ sung
1	30	q ≤ 500; tổng độ dài ≤ 5000
2	30	Chỉ có thao tác thêm
3	40	Không có giới hạn bổ sung

Subtask 1 - 30 điểm

Giữ các từ hiện có trong một tập. Khi thêm hoặc xóa s, chỉ các cặp chứa s thay đổi. So sánh s với từng từ t khác đang hoạt động theo cả hai chiều hậu tố để đếm số cặp cần cộng hoặc trừ. Không đếm cặp (s,s); tính thay đổi trước khi sửa tập.

Độ phức tạp: O(qL) thời gian theo cận tổng, O(L) bộ nhớ; L là tổng độ dài thao tác.

```
delta=0
cho mọi t đang có và t!=s:
    delta += [s là hậu tố t]+[t là hậu tố s]
thêm: ans+=delta; xóa: ans-=delta; rồi sửa tập
```

Subtask 2 - 30 điểm

Dùng trie của chuỗi đảo. Trước khi thêm s, cộng số từ kết thúc ở các tổ tiên thực sự của nút cuối và sub tại nút cuối. Hai số này đếm từ ngắn hơn và dài hơn có quan hệ hậu tố với s. Sau đó tăng các bộ đếm trên đường.

Độ phức tạp: O(L) thời gian với bảng chữ cái cố định, O(L) nút.

```
ans += số terminal ở tổ tiên thực sự + sub[nút cuối]
tăng sub trên đường, bật terminal[nút cuối]
```

Subtask 3 - 40 điểm

Giữ trie và bộ đếm từ đang hoạt động. Khi xóa s, trước cập nhật phải trừ shorter+sub[v]-1; số 1 loại chính s khỏi cây con. Khi thêm dùng shorter+sub[v]. Nút trie có thể giữ lại sau khi xóa, nhưng terminal và sub phải phản ánh đúng tập hiện tại.

Độ phức tạp: O(L) thời gian, O(L) nút; ans dùng 64 bit.