SIROCCO CUP 2026 / KC3

Không chuyên - Đề 3

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

180 phút · 4 bài · 400 điểm

Quan sát, duyệt và đếm

Bài	Tên bài	Tag
A	GDSC và BKAC	Xâu, Mảng đánh dấu
B	Bánh quy	Toán học, Quy hoạch động, Bất biến tổng
C	Tối ưu triển khai	Quay lui, Duyệt vét cạn, Bài toán chọn nhóm
D	Đồ thị tối giản	Đồ thị, Đếm tổ hợp, Phân tầng BFS

Problemset trực tuyến
https://codeforces.com/gym/728948/problems

KC3-A / KHÁM PHÁ LỜI GIẢI

A. GDSC và BKAC

Xâu | Mảng đánh dấu

ĐIỂM MẤU CHỐT

Thời điểm hoàn tất là vị trí xuất hiện muộn nhất trong các lần xuất hiện đầu tiên của bốn chữ cần có.

Từ quan sát đến thuật toán

Mỗi câu lạc bộ cần một tập chữ, không phải một xâu con hay một dãy con theo thứ tự. Vì vậy, với mỗi chữ in hoa, chỉ cần biết vị trí xuất hiện đầu tiên của nó.

Đặt first[c] là vị trí đầu tiên của chữ c, hoặc vô cùng nếu không xuất hiện. Thời điểm GDSC hoàn tất là max(first[G],first[D],first[S],first[C]); tương tự cho BKAC. So sánh hai thời điểm này.

Cũng có thể quét từ trái sang phải, đánh dấu chữ và ghi nhận lúc một bên lần đầu có đủ bốn chữ. Cách dùng vị trí đầu tiên giúp nhìn rõ vì sao các chữ lặp lại không ảnh hưởng.

Ví dụ để tự kiểm tra

Với GDSBKAC, G,D,S nằm ở vị trí 1,2,3 và C ở vị trí 7, nên GDSC hoàn tất ở 7. B,K,A ở vị trí 4,5,6 nhưng BKAC cũng chờ C tại vị trí 7: kết quả DRAW. Không được coi GDSC là một mẫu cần xuất hiện liên tiếp.

KC3-A / CHỨNG MINH VÀ CÀI ĐẶT

GDSC và BKAC

Vì sao thuật toán đúng?

Một tiền tố kết thúc ở i có đủ một tập chữ khi và chỉ khi lần xuất hiện đầu tiên của mọi chữ trong tập đều không quá i. Vị trí nhỏ nhất thỏa điều này là giá trị lớn nhất của các vị trí đầu tiên. Hai giá trị mà thuật toán so sánh chính là hai thời điểm hoàn tất, nên kết luận thắng hoặc hòa là đúng.

Mã giả

```
first[A..Z] = vô_cùng
cho i = 1..n:
    nếu first[s[i]] = vô_cùng: first[s[i]] = i
g = max(first[G],first[D],first[S],first[C])
b = max(first[B],first[K],first[A],first[C])
nếu g < b: in GDSC
nếu b < g: in BKAC
nếu g = b: in DRAW
```

Độ phức tạp

O(n) thời gian và O(26)=O(1) bộ nhớ cho một test.

Các lỗi dễ gặp

- Đặt vô cùng lớn hơn n; câu lạc bộ không thể hoàn tất phải luôn thua bên hoàn tất được.
- Chữ C được dùng cho cả hai bên. Không xóa hay đánh dấu chữ là đã bị bên kia lấy mất.
- Không kiểm tra thứ tự G,D,S,C. Đề chỉ yêu cầu có đủ chữ.

Lời giải subtask duy nhất - 100 điểm

Bài chỉ có một nhóm điểm. Quét xâu để tìm vị trí xuất hiện đầu tiên của từng chữ. Mỗi câu lạc bộ hoàn tất tại vị trí lớn nhất trong bốn chữ cần thu thập. So sánh hai vị trí, dùng vô cùng cho chữ không xuất hiện; không yêu cầu các chữ theo thứ tự.

O(n) thời gian và O(1) bộ nhớ cho mỗi test.

KC3-B / KHÁM PHÁ LỜI GIẢI

B. Bánh quy

Toán học | Quy hoạch động | Bất biến tổng

ĐIỂM MẤU CHỐT

Không cần biết từng giá trị có thể đạt: tổng của hai nhánh min và max luôn bằng x+a.

Từ quan sát đến thuật toán

Sau khi xử lý một số lò, gọi ways là số cấu hình đã chọn và sum là tổng giá trị hiện tại của bánh qua tất cả các cấu hình ấy. Ban đầu ways=1 và sum=a_1.

Khi gặp tham số a tiếp theo, một cấu hình có giá trị x sinh ra hai cấu hình con: min(x,a) và max(x,a). Hai giá trị này cộng lại luôn bằng x+a. Cộng trên toàn bộ ways cấu hình cho new_sum=sum+ways×a. Số cấu hình mới là new_ways=2×ways.

Cập nhật hai đại lượng theo modulo 10^9+7. Kể cả khi min và max bằng nhau, hai chế độ vẫn là hai cấu hình khác nhau: luôn nhân ways với 2.

Ví dụ để tự kiểm tra

Với a=(2,1,3): ban đầu (sum,ways)=(2,1). Qua số 1: sum=2+1×1=3, ways=2; hai giá trị là 1 và 2. Qua số 3: sum=3+2×3=9, ways=4. Bốn giá trị cuối là 1,3,2,3, tổng 9.

KC3-B / CHỨNG MINH VÀ CÀI ĐẶT

Bánh quy

Vì sao thuật toán đúng?

Bất biến sum là tổng giá trị và ways là số cấu hình đúng với tiền tố đã xử lý. Với mỗi cấu hình x, tổng giá trị hai cấu hình con là x+a, nên tổng mới là sum+ways×a và số cấu hình tăng gấp đôi. Khởi tạo đúng cho tiền tố một phần tử; quy nạp chứng minh giá trị sum cuối là đáp án.

Mã giả

```
MOD = 1000000007
sum = a[1] mod MOD
ways = 1
cho i = 2..n:
    sum = (sum + ways*a[i]) mod MOD
    ways = (2*ways) mod MOD
in sum
```

Độ phức tạp

O(n) thời gian và O(1) bộ nhớ phụ; có thể xử lý khi đọc dãy.

Các lỗi dễ gặp

- Cập nhật sum bằng ways cũ rồi mới nhân đôi ways.
- Không gộp hai cấu hình chỉ vì cho cùng kết quả cuối.
- Dùng long long cho tích ways×a[i] trước khi lấy modulo.

KC3-B / LỜI GIẢI TỪNG SUBTASK

Bánh quy

Subtask	Điểm	Giới hạn bổ sung
1	30	$n \le 20$
2	30	Mọi $a_i \le 20$
3	40	Không có giới hạn bổ sung

Subtask 1 - 30 điểm

Duyệt hai lựa chọn min/max cho từng lò. DFS giữ giá trị hiện tại x; tới hết dãy thì cộng x vào đáp án. Hai nhánh vẫn phải duyệt riêng khi cùng cho một giá trị, vì chúng đại diện hai cấu hình chế độ khác nhau.

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

```
DFS(i,x): nếu i>n thì cộng x vào ans
    ngược lại gọi DFS(i+1,min(x,a[i]))
                và DFS(i+1,max(x,a[i]))
```

Subtask 2 - 30 điểm

Mọi kết quả min/max vẫn nằm trong [1,20]. Lưu f[v] là số cấu hình đang có giá trị v. Khi gặp a, từ mỗi v chuyển f[v] cách tới min(v,a) và max(v,a); dùng một mảng mới cho lớp kế tiếp. Cuối cùng lấy tổng v×f[v].

Độ phức tạp: O(20n) thời gian, O(20) bộ nhớ; tính số cách theo modulo.

```
g[1..20]=0
cho v=1..20: g[min(v,a)]+=f[v]; g[max(v,a)]+=f[v]
f=g; cuối cùng ans=Σ v*f[v] (mod MOD)
```

Subtask 3 - 40 điểm

Không giữ phân bố theo giá trị nữa. Dùng min(x,a)+max(x,a)=x+a để chỉ giữ sum và ways. Mỗi bước cập nhật sum←sum+ways×a, rồi ways←2×ways; tất cả lấy modulo 10^9+7.

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

KC3-C / KHÁM PHÁ LỜI GIẢI

C. Tối ưu triển khai

Quay lui | Duyệt vét cạn | Bài toán chọn nhóm

ĐIỂM MẤU CHỐT

Ngân sách lớn không cản vét cạn: tổng số phương án chỉ có 20, còn mỗi dự án được chọn tối đa một.

Từ quan sát đến thuật toán

Một bước DFS xử lý một dự án. Từ trạng thái (i,chi_phí,doanh_thu), có k_i+1 nhánh: bỏ dự án hoặc chọn một trong k_i phương án. Nếu chi phí mới vượt ngân sách thì bỏ nhánh đó.

Tổng số lựa chọn cuối là tích (k_i+1). Với $k_i \geq 1$, ta có $k_i+1 \leq 2^{k_i}$; nhân lại được tích $(k_i+1) \leq 2^K$, $K=\Sigma k_i \leq 20$. Chính giới hạn tổng phương án khiến phép duyệt này đủ nhỏ.

Chi phí và doanh thu đều không âm, nên cắt nhánh khi chi phí vượt ngân sách là an toàn. Tuy nhiên doanh thu không được cộng lại vào ngân sách: đây là ràng buộc tổng chi phí so với ngân sách ban đầu.

Ví dụ để tự kiểm tra

Ngân sách 5. Dự án 1 có (doanh thu,chi phí)=(10,3) hoặc (14,5); dự án 2 có (9,2). Chọn (10,3) và (9,2) cho doanh thu 19, tốt hơn phương án doanh thu 14. Không được chọn cả hai phương án của dự án 1.

KC3-C / CHỨNG MINH VÀ CÀI ĐẶT

Tối ưu triển khai

Vì sao thuật toán đúng?

Mỗi nghiệm hợp lệ quyết định duy nhất một lựa chọn cho từng dự án: bỏ hoặc một phương án. DFS sinh đúng các quyết định đó và không bao giờ chọn hai phương án của cùng dự án. Chi phí không âm nên nhánh đã vượt ngân sách không thể trở lại hợp lệ. Tất cả nghiệm hợp lệ đều được xét; lấy doanh thu lớn nhất trả đúng tối ưu.

Mã giả

```
best = 0
DFS(i,cost,revenue):
    nếu cost > budget: trả về
    nếu i = n+1:
        best = max(best,revenue)
        trả về
    DFS(i+1,cost,revenue)  // bỏ dự án i
    với mỗi phương án (r,e) của dự án i:
        DFS(i+1,cost+e,revenue+r)

DFS(1,0,0)
in best
```

Độ phức tạp

O(2^K) nút duyệt theo cận trên hình học khi mọi $k_i \geq 1$; O(n+K) bộ nhớ gồm dữ liệu và ngăn xếp đệ quy. $K \leq 20$.

Các lỗi dễ gặp

- Không dùng mảng DP kích thước budget khi budget có thể tới 10^9.
- Dùng long long cho tổng doanh thu và tổng chi phí; mỗi tổng có thể vượt 32 bit.
- Không dùng doanh thu để tài trợ các lựa chọn sau.

KC3-C / LỜI GIẢI TỪNG SUBTASK

Tối ưu triển khai

Subtask	Điểm	Giới hạn bổ sung
1	30	$\Sigma k_i \le 12$
2	30	$b \le 2000$
3	40	Không có giới hạn bổ sung

Subtask 1 - 30 điểm

DFS theo dự án; tại mỗi dự án chọn bỏ qua hoặc đúng một phương án. Giữ tổng chi phí và doanh thu, cắt nhánh khi chi phí vượt ngân sách. Có tối đa tích $(k_i+1) \le 2^{12}=4096$ tổ hợp nên duyệt hết được.

Độ phức tạp: O(2^K) thời gian, O(n+K) bộ nhớ; $K=\Sigma k_i \le 12$.

```
DFS(i,cost,revenue):
    thử bỏ dự án i
    thử từng phương án (r,e) của dự án i
    chỉ đi tiếp nếu cost+e<=budget
```

Subtask 2 - 30 điểm

DP theo ngân sách và theo lớp dự án. dp[c] là doanh thu lớn nhất khi dùng đúng chi phí c sau các dự án đã xét. Khởi tạo dp[0]=0, còn lại không thể đạt. Với mỗi dự án, sao chép dp sang next để cho phép bỏ qua; mọi phương án chỉ chuyển từ dp cũ sang next.

Độ phức tạp: O(bK) thời gian, O(b+K) bộ nhớ; dùng 64 bit cho doanh thu.

```
next=dp
cho mỗi phương án (r,e) của dự án:
    với c>=e và dp[c-e] đạt được:
        next[c]=max(next[c],dp[c-e]+r)
dp=next; đáp án=max(dp[0..b])
```

Subtask 3 - 40 điểm

Ngân sách lớn nhưng K vẫn không quá 20. Quay lại DFS theo dự án, vì $k_i+1 \le 2^{k_i}$ cho tích số lựa chọn không quá 2^{20}. Không chọn hai phương án cùng dự án, và không dùng doanh thu thu được để tăng ngân sách.

Độ phức tạp: O(2^K) thời gian, O(n+K) bộ nhớ; $K \le 20$.

KC3-D / KHÁM PHÁ LỜI GIẢI

D. Đồ thị tối giản

Đồ thị | Đếm tổ hợp | Phân tầng BFS

ĐIỂM MẤU CHỐT

Ít cạnh nhất nghĩa là một cây; mỗi đỉnh ở tầng d chỉ cần chọn một cha ở tầng d-1.

Từ quan sát đến thuật toán

Gọi cnt[d] là số đỉnh có khoảng cách d đến đỉnh 1, với cnt[0]=1. Nếu có đỉnh ở tầng d nhưng tầng d-1 rỗng thì không tồn tại đồ thị thỏa mãn.

Đồ thị liên thông luôn cần ít nhất n-1 cạnh. Nếu mọi tầng trước đều có đỉnh, ta dựng được một cây bằng cách cho mỗi đỉnh tầng d chọn một cha ở tầng d-1. Cây ấy có đúng khoảng cách yêu cầu và n-1 cạnh, nên n-1 là số cạnh ít nhất.

Trong một cây hợp lệ, đường duy nhất từ mỗi đỉnh về gốc phải đi qua một cha ở tầng trước. Mỗi đỉnh tầng d có cnt[d-1] cách chọn cha, độc lập với các đỉnh khác. Nhân tất cả lựa chọn: đáp án = tích cnt[d-1]^cnt[d], modulo 998244353.

Ví dụ để tự kiểm tra

Khoảng cách (1,1,2) tạo các tầng cnt=(1,2,1). Hai đỉnh tầng 1 bắt buộc nối gốc; đỉnh tầng 2 được nối một trong hai đỉnh tầng 1. Có $1^2 \times 2^1 = 2$ cây. Khoảng cách (2,2) thiếu tầng 1 nên đáp án 0.

KC3-D / CHỨNG MINH VÀ CÀI ĐẶT

Đồ thị tối giản

Vì sao thuật toán đúng?

Cách dựng nối về tầng trước giảm khoảng cách tầng đúng 1 ở mỗi bước, nên không có chu trình và mọi đỉnh về được gốc; độ dài đường về gốc đúng bằng tầng đã gán. Nó cho một đồ thị tối thiểu n-1 cạnh. Ngược lại, mọi đồ thị tối thiểu là cây và mỗi đỉnh có đúng một cha thuộc tầng trước. Hai chiều này tạo song ánh giữa các cây hợp lệ và các lựa chọn cha được công thức đếm.

Mã giả

```
cnt[0] = 1; các cnt khác = 0
đếm các khoảng cách d[2..n] vào cnt
ans = 1
cho d = 1..max_distance:
    nếu cnt[d] > 0 và cnt[d-1] = 0:
        ans = 0; dừng
    ans = ans * POWMOD(cnt[d-1],cnt[d]) mod MOD
in ans
```

Độ phức tạp

O(n+Σ log(cnt[d]+1)) thời gian với lũy thừa nhị phân; tổng này là O(n). Bộ nhớ O(n).

Các lỗi dễ gặp

- Đỉnh 1 không nằm trong $d_2,\ldots,d_n$; phải tự đặt cnt[0]=1.
- Không thêm cạnh trong cùng tầng: chúng làm tăng số cạnh và không thuộc đồ thị tối giản.
- Lấy modulo sau mỗi phép nhân; giữ tích trung gian trong 64 bit.

KC3-D / LỜI GIẢI TỪNG SUBTASK

Đồ thị tối giản

Subtask	Điểm	Giới hạn bổ sung
1	30	$n \le 6, t \le 20$
2	30	Mọi $d_i \le 2$
3	40	Không có giới hạn bổ sung

Subtask 1 - 30 điểm

Có nhiều nhất 15 cạnh tiềm năng. Duyệt các tập cạnh nhưng chỉ xét tập có đúng n-1 cạnh, rồi BFS từ đỉnh 1. Giữ tập nếu mọi đỉnh đến được và khoảng cách khớp dữ liệu. Các đồ thị tối giản hợp lệ đều là cây nên việc chỉ xét n-1 cạnh không bỏ sót đáp án.

Độ phức tạp: O(2^E + C(E,n-1)×n²) thời gian, O(n²) bộ nhớ; E=n(n-1)/2≤15.

```
duyệt mask cạnh có đúng n-1 bit bật
    dựng đồ thị, BFS từ đỉnh 1
    nếu khoảng cách mọi đỉnh khớp: ans++
```

Subtask 2 - 30 điểm

Chỉ có gốc, tầng 1 và tầng 2. Mọi đỉnh tầng 1 phải nối gốc; mỗi đỉnh tầng 2 chọn một cha bất kỳ trong tầng 1. Nếu tầng 1 rỗng thì không thể liên thông. Ngược lại đáp án là cnt[1]^cnt[2], tính theo modulo.

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

```
nếu cnt[1]=0: in 0
ngược lại: in POWMOD(cnt[1],cnt[2],MOD)
```

Subtask 3 - 40 điểm

Đếm số đỉnh ở mọi tầng, kiểm tra không bị đứt tầng, rồi nhân cnt[d-1]^cnt[d]. Mỗi đỉnh chọn đúng một cha ở tầng trước; phép nối này tạo một cây có khoảng cách đúng yêu cầu. Lũy thừa modulo dùng bình phương nhanh.

Độ phức tạp: O(n) thời gian theo tổng số đỉnh, O(n) bộ nhớ.