Hướng Dẫn Giải
Đề Thi Thử VOI2023 Ngày 1

TỔNG QUAN ĐỀ THI

Tên bài	**Tên chương trình**	**Dữ liệu đầu vào**	**Kết quả đầu ra**	**Số điểm**
Bài 1. Thuỷ cung	AQUARIUM.*	AQUARIUM.INP	AQUARIUM.OUT	7 điểm
Bài 2. Xử lý xâu	KSTRING.*	KSTRING.INP	KSTRING.OUT	7 điểm
Bài 3. Đếm hình chữ nhật 0	RECTCNT.*	RECTCNT.INP	RECTCNT.OUT	6 điểm

Dấu * được thay thế bởi `pas` hoặc `cpp` của ngôn ngữ lập trình tương ứng là Pascal và C++.

Mục lục

Thủy cung — AQUARIUM 2
Xử lý xâu — KSTRING 3
Đếm hình chữ nhật 0 — RECTCNT 4

Bài 1. Thủy cung — AQUARIUM

Hướng dẫn giải

Subtask 1: 20% số test ứng với $n \leq 8$.

Dùng phương pháp đệ quy quay lui (backtracking). Với phần tử i, giả sử hiện tại đã có g nhóm, ta sẽ quyết định xem nên đặt phần tử i vào một trong g nhóm có sẵn, hay tạo nhóm mới (tăng số nhóm lên $g + 1$).

Subtask 2: 20% số test ứng với $n \leq 15$.

Lời giải 1: Cải tiến phương pháp đệ quy quay lui bằng kĩ thuật nhánh cận: Khi đặt một phần tử mới vào nhóm, ta sẽ cập nhật lại tổng độ bất ổn hiện tại. Nếu tổng độ bất ổn vượt quá đáp án tốt nhất hiện tại thì ta ngay lập tức thoát khỏi nhánh hiện tại.

Lời giải 2: Dùng phương pháp quy hoạch động (QHĐ) mặt nạ bit:

Gọi $dp[i][mask]$ là tổng độ bất ổn nhỏ nhất khi chia các phần tử trong dãy con của a được biểu diễn bởi $mask$ thành i nhóm (dãy con sẽ chứa phần tử a_i khi và chỉ khi bit thứ i của $mask$ là bit 1).

Gọi $cost(s)$ là tổng độ bất ổn của các phần tử trong tập con được biểu diễn bởi mặt nạ bit s. Ta có công thức QHĐ như sau:

$$dp[i][mask] = min(dp[i-1][mask \oplus s] + cost(s)) \text{ với mọi } s \text{ là tập con của } mask$$

Ta dùng kĩ thuật duyệt qua từng mặt nạ bit con của từng $mask$. Khi đó, tổng số mặt nạ bit được duyệt qua là 3^n, và độ phức tạp là $O(3^n m)$. Với ngôn ngữ C++, việc duyệt qua các mặt nạ bit con của $mask$ có thể được thực hiện như sau:

```
for(int mask = 0; mask < (1<<n); ++mask) {
  for(int s = mask; s > 0; s = (s - 1)&mask) {
    // s là các mặt nạ bit con của mask
  }
}
```

Subtask 3: 20% số test ứng với $m = 2$.

Nhận xét rằng, tổng độ bất ổn của mỗi phần tử có thể được tính theo công thức: $S = \sum_{i=1}^{n} a_i * c_i$ với c_i là số lượng phần tử nằm cùng nhóm với phần tử a_i.

Khi đó, với $m = 2$, ta sẽ có một nhóm gồm x phần tử và nhóm còn lại gồm $n - x$ phần tử (không mất tính tổng quát, giả sử $x \leq n - x$ hay $x \leq \lfloor \frac{n}{2} \rfloor$). Để tổng độ bất ổn nhỏ nhất, ta cần đặt các phần tử nhỏ hơn vào nhóm có ít phần tử hơn.

Do đó, ta sẽ duyệt x từ 1 đến $\lfloor \frac{n}{2} \rfloor$. Với mỗi x, ta sẽ được x phần tử có giá trị nhỏ nhất vào nhóm 1, $n - x$ phần tử có giá trị lớn nhất vào nhóm 2 rồi cập nhật lại đáp án với độ bất ổn của cách chia nhóm này.

Subtask 4: 30% số test khác ứng với $n \leq 100$.

Ta mở rộng nhận xét từ lời giải subtask 3 như sau:

Với một cách chia nhóm bất kì, giả sử ta sắp xếp lại dãy a sao cho các phần tử cùng nhóm sẽ nằm liên tiếp nhau, đồng thời các nhóm sẽ có số lượng phần tử giảm dần (nói cách khác, $c_1 \geq c_2 \geq \ldots \geq c_n$). Khi đó, để tổng $S = \sum_{i=1}^{n} a_i * c_i$ đạt giá trị nhỏ nhất, ta nên sắp xếp lại các phần tử a_i theo giá trị tăng dần.

Nói cách khác, giả sử dãy a được sắp xếp tăng dần. Khi đó, cách chia nhóm tối ưu sẽ là chia thành các nhóm liên tiếp nhau, đồng thời số lượng phần tử của các nhóm giảm dần.

Với nhận xét trên, ta có lời giải QHĐ như sau: ban đầu, ta sắp xếp dãy a theo thứ tự tăng dần. Gọi $dp[i][j]$ là tổng độ bất ổn nhỏ nhất khi chia i phần tử đầu tiên vào j nhóm. Ta có công thức QHĐ sau:

$$dp[i][j] = min(dp[i-1][j-len] + len \times (a_{j-len+1} + \ldots + a_i)) \text{ với mọi } 1 \leq 1en \leq i$$

Tổng $a_{j-len+1} + \ldots + a_i$ có thể được tính nhanh bằng kĩ thuật tổng tiền tố.

Độ phức tạp: $O(n^2 m)$

Subtask 5: 10% số test còn lại ứng với $n \leq 2\,000$.

Ta nhận xét rằng, do các nhóm có kích thước giảm dần, nên kích thước của nhóm thứ j sẽ không thể vượt quá $\frac{i}{j}$ (vì nếu không, tổng số lượng phần tử của các nhóm sẽ lớn hơn i). Do đó, ta cải tiến công thức QHĐ như sau:

$$dp[i][j] = min(dp[i-1][j-len] + len \times (a_{j-len+1} + \ldots + a_i)) \text{ với mọi } 1 \leq 1en \leq \left\lfloor \frac{i}{j} \right\rfloor$$

Nhận xét rằng $\frac{i}{1} + \frac{i}{2} + \ldots + \frac{i}{i} = i \times \left(\frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{i}\right) \approx i \ln i$ (tính chất của dãy điều hòa). Do đó, độ phức tạp của thuật toán là $O(nm \ln n)$.

Bài 2. Xử lý xâu — KSTRING

Hướng dẫn giải

Ý tưởng chủ đạo trong bài là sử dụng thuật toán hash.

Các Subtask 1,2, 4 mô tả cách làm với $d = 1$, với $d = 0$ đơn giản hơn, có thể tham khảo cách làm Subtask 3.

Subtask 1: 25% test ứng với $d \leq 1, N \leq 1000$.

Thuật toán trực tiếp duyệt toàn bộ để so sánh từng cặp xâu một.

Subtask 2: 25% test ứng với $d \leq 1, N \leq 3000$.

- Gọi $DPL[i][j]$ là độ dài lớn nhất mà xâu bắt đầu từ vị trí i và xâu bắt đầu từ vị trí j từ trái qua phải là bằng nhau.
- Gọi $DPR[i][j]$ là độ dài lớn nhất mà xâu bắt đầu từ vị trí i và xâu bắt đầu từ vị trí j từ phải qua trái là bằng nhau.

Hai mảng trên chuẩn bị bằng hàm quy hoạch động đơn giản.

Với 2 xâu bất kì độ dài k, giả xử 2 xâu là $S[x, x+k-1]$ và $S[y, y+k-1]$, kiểm tra nếu $DPL[x][y] + DPR[x+k-1][y+k-1] \geq k-1$, khi đó khoảng cách Hamming của 2 xâu là ≤ 1. Từ đây duyệt trực tiếp và kiểm tra từng cập xâu một.

Subtask 3: 25% test ứng với $d = 0, N \leq 50000$.

Với mỗi k, tính hash của các xâu và đếm các cặp xâu giống nhau.

Subtask 4: 25% test ứng với $d \leq 1, N \leq 50000$.

Ta có thể giải bài toán cho 1 giá trị k với độ phức tạp $O(n)$ như sau.

- Với mỗi xâu độ dài k được cắt, ta sẽ tính hash của xâu đó và k xâu nữa, với kí tự thứ i của xâu đó biến thành kí tự ?. Bằng cách duyệt qua từng xâu một và duy trì số lần xuất hiện của mỗi hash, ta có thể tính được kết quả. Duy trì số lần xuất hiện của mỗi hash ta có thể sử dụng hash table để đạt được độ phức tạp $O(n)$.
- Với mỗi k ta cũng có thể giải được trong $O((\frac{n}{k})^2)$ như sau. Với mỗi cặp xâu, lấy hiệu của hai giá trị hash. Khi đó ta có thể kiểm tra được hiệu của hai hash này khác nhau không quá một kí tự bằng cách lưu lại tất cả các giá trị hash của các xâu độ dài k mà tất cả các kí tự có giá trị 0 còn 1 kí tự có giá trị là từ -25 đến 25.
- Lưu tất cả các giá trị này trong một hashset, khi đó ta có thể kiểm tra được hiệu 2 hash có nằm trong hashset hay không.
- Với $k \leq \sqrt{n}$, ta sẽ chạy thuật toán $O(n)$, với $k \geq \sqrt{n}$ ta sẽ chạy thuật toán $O((\frac{n}{k})^2)$.
- Như vậy độ phức tạp tổng là $O(n\sqrt{n})$, tuy nhiên hằng số khá lớn do thuật toán hash và phải sử dụng hashtable và hashset

Bài 3. Đếm hình chữ nhật 0 — RECTCNT

Hướng dẫn giải

Subtask 1: 20% số test ứng với $n, q \leq 50$.

Duyệt tất cả các hình chữ nhật và dùng mảng tiền tố (prefix sum) để kiểm tra có chứa toàn 0 hay không.

ĐPT: $O(q * n^4)$.

Subtask 2: 20% số test ứng với $n, q \leq 150$.

Chặn hai hàng trên dưới của hình chữ nhật là i_1 và i_2, bây giờ ta cần đếm số số cặp j_1 j_2 là chặn cột của hình chữ nhật thỏa mãn, dùng kỹ thuật hai con trỏ (two pointers) để tìm các vùng $[\ell, r]$ và từ cột ℓ đến cột r xét từ hàng i_1 đến hàng i_2 chỉ toàn chứa số 0, như vậy số cặp (j_1, j_2) là tổng $\frac{(r-\ell)(r-\ell+1)}{2}$.

ĐPT: $O(q * n^3)$.

Subtask 3: 20% số test ứng với $n, q \leq 400$.

Tương tự như subtask 2, ta chặn hàng i_2 nhưng sẽ duyệt i_1 giảm dần, giả sử với i_1 ta có tập các đoạn có thể chọn (j_1, j_2) là $S = \{[\ell_1, r_1], [\ell_2, r_2], ...\}$, ta xem xét sự thay đổi của S khi thay đổi $i_1 = i_1 - 1$, sẽ có các đoạn $[\ell, r] \in S$ bị cắt ra thành nhiều đoạn nhỏ hơn, bằng cách lưu trữ các ô 1 tại hàng i và dùng set để lưu các đoạn $[\ell, r]$ ta sẽ cập nhật được tập S và đếm số cặp (j_1, j_2).

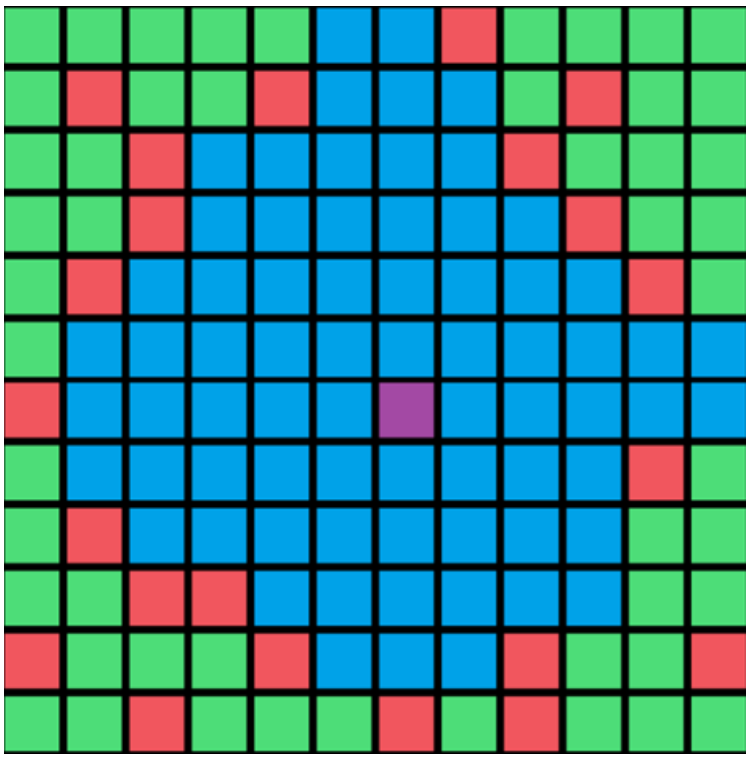
ĐPT: $O(q * n^2 \log n)$.

Subtask 4: 20% số test ứng với $n, q \leq 1000$.

Làm giống Subtask 3 với việc đếm số lượng hình chữ nhật thỏa mãn của bảng đầu tiên, với mỗi truy vấn ta chỉ cần đếm xem có bao nhiêu hình chữ nhất thỏa mãn mà chứa ô (r, c), ta cần tìm các ô bao quanh ô (r, c) như hình dưới đây.

Mỗi cột ta lưu danh sách các ô 1 và có thể dùng hàm `lower_bound` để tìm các ô bao quanh, sau đó đếm như Subtask 3 nhưng vì chỉ cần quan tâm đến các ô bao quanh nên số lượng cần xét chỉ là $2n$. Do đó với mỗi truy vấn ta chỉ cần ĐPT $O(n * \log n)$.

Đpt của thuật toán là $O((n+q) * n * \log n)$.



Subtask 5: 20% số test ứng với $n, q \leq 5000$.

Ý tưởng giống Subtask 4 nhưng dùng mảng `next` và `prev` để tìm nhanh các đoạn và các ô bao quanh.

ĐPT: $O((n+q)n)$.