# Bài 1. Bảo vệ chất độc (7 điểm)

## Phân bổ điểm

- Có 30% số test ứng với 30% số điểm của bài thỏa mãn: $3 \le n \le 10^2; 1 \le m \le 15$;
- 30% số test khác ứng với 30% số điểm của bài thỏa mãn: tất cả các con rắn có độ dài bằng nhau (độ dài của con rắn thứ $i$ là $|hx_i - tx_i| + |hy_i - ty_i| + 1$);
- 40% số test còn lại ứng với 40% số điểm của bài không có ràng buộc gì thêm.

## Thuật toán

### Subtask 1 (30%): $3 \le n \le 10^2; 1 \le m \le 15$

Duyệt tất cả các tập con của $m$ con rắn. Mỗi tập con kiểm tra xem nó có là cấu hình "an toàn" hay không. Cập nhật cấu hình an toàn mà ít con rắn nhất.

Độ phức tạp $O(2^m.n)$.

### Subtask 2 (30%): Tất cả các con rắn có độ dài bằng nhau (độ dài của con rắn thứ $i$ là $|hx_i - tx_i| + |hy_i - ty_i| + 1$)

Đầu tiên chúng ta xem xét một số tối thiểu các con rắn bảo vệ cạnh ngang của lưới hình vuông $k \times k$ chứa chất độc. Chúng ta nói rằng một cấu hình các con rắn bảo vệ được cạnh ngang của lưới chứa chất độc khi và chỉ khi nếu có một mũi tên được bắn thì hoặc mũi tên này không chạm vào cạnh ngang hoặc các con rắn sẽ chặn nó trước khi đi vào cạnh ngang hoặc sau khi đi vào cạnh ngang này và thoát ra ở cạnh ngang kia thì bị chặn bởi một con rắn trong cấu hình.

Bây giờ chúng ta sẽ phát biểu mô hình toán học của bài toán như sau. Các con rắn $hx_i, hy_i, tx_i, ty_i$ được biểu thị các đoạn ngang $[a_i; b_i]$ với $a_i = min(hy_i, ty_i)$ và $b_i = max(hy_i, ty_i)$. Cạnh ngang của lưới hình vuông $k \times k$ được biểu thị đoạn ngang $[l; r]$ với $l = \frac{n-k}{2} + 1$ và $r = \frac{n-k}{2} + k$. Bài toán trở thành tìm một số tối thiểu các đoạn ngang $[a_i; b_i]$ sao cho nó bao phủ hoàn toàn đoạn $[l; r]$. Do các đoạn ngang có độ dài bằng nhau, nên việc lấy hợp các đoạn này khá dễ dàng và sau đó là kiểm tra hợp của chúng có chứa đoạn $[l; r]$.

Tiếp theo chúng ta sẽ xem xét một số tối thiểu các con rắn bảo vệ cạnh dọc của lưới hình vuông $k \times k$ chứa chất độc. Công việc này xử lý tương tự như cạnh ngang.

Cuối cùng chúng ta nhận thấy rằng không có con rắn nào có thể bảo vệ cả cạnh ngang và cạnh dọc của lưới $k \times k$ cùng một lúc. Vì vậy số con rắn tối thiểu để bảo vệ lưới chất độc bằng tổng số con rắn tối thiểu bảo vệ cạnh ngang và số con rắn tối thiểu bảo vệ cạnh dọc.

### Subtask 3 (40%): Không có thêm ràng buộc nào

Subtask này giải quyết cơ bản giống subatsk ở trên, chỉ khác việc xử lý tìm một số tối thiểu các đoạn ngang $[a_i; b_i]$ sao cho nó bao phủ hoàn toàn đoạn $[l; r]$ như sau.

Đây là một trong những bài toán cơ bản, có thể được giải quyết một cách tham lam. Ý tưởng là sắp xếp các đoạn $[a_i; b_i]$ theo đầu mút bên trái của chúng. Sau đó ta xử lý các đoạn này từ trái sang phải theo thứ tự đó và cố gắng phủ điểm ngoài cùng bên trái chưa được bao phủ $[l; r]$. Cụ thể hơn, giả sử rằng $cur$ là điểm bên trái chưa được bao phủ của $[l; r]$. Chúng ta kiểm tra tất cả các đoạn có điểm cuối bên trái không lớn hơn $cur$ và đưa ra đoạn có điểm cuối bên phải lớn nhất. Đặt điểm cuối lớn nhất này là $pos$, vì vậy chúng ta cập nhật $cur$ thành $pos + 1$ và tiếp tục quá trình cho đến khi tất cả các điểm của $[l, r]$ được bao phủ hoặc đã kiểm tra tất cả các đoạn mà không có cách nào bao phủ đoạn $[l; r]$. Trường hợp không có cách nào bao phủ đoạn $[l; r]$ thì câu trả lời của bài toán là "$-1$".

Độ phức tạp thuật toán là $O(m. log_2 m)$.

## Bài 2. Thương gia tham lam (7 điểm)

## Phân bổ điểm

- Có 25% số test ứng với 25% số điểm của bài thỏa mãn: $1 \leq n, m, k \leq 200$;
- 25% số test khác ứng với 25% số điểm của bài thỏa mãn: $1 \leq n, m, k \leq 2000$;
- 25% số test khác ứng với 25% số điểm của bài thỏa mãn: $m = n - 1$;
- 25% số test còn lại ứng với 25% số điểm của bài không có ràng buộc gì thêm.

## Thuật toán

**Subtask 1 (25%): $1 \leq n, m, k \leq 200$**

Bài toán yêu cầu trả lời các truy vấn, với mỗi truy vấn $(s_i, l_i)$ cần tính số cạnh sao cho khi loại bỏ nó thì không có đường đi giữa hai đỉnh $s_i$ và $l_i$.

Đối với mỗi cặp $(s_i, l_i)$, ta lần lượt thử bỏ từng cạnh đi một, sau đó xem có đường đi giữa $s_i$ và $l_i$ hay không.

Độ phức tạp $O(n.m.k)$.

**Subtask 2 (25%): $1 \leq n, m, k \leq 2000$**

Trước hết ta nhận thấy rằng các cạnh cần loại bỏ phải là cầu. Do vậy ta sẽ nén các thành phần liên thông 2-cạnh thành một đỉnh và thu được cây cầu của đồ thị.

Cuối cùng ta trả lời mỗi truy vấn $(s_i, l_i)$ như sau. Gọi $S, L$ là hai đỉnh trên cây cầu tương ứng với hai thành phần liên thông 2-cạnh chứa hai đỉnh $s_i, l_i$. Sau đó tính số cạnh trên đường đi ngắn nhất giữa hai đỉnh $S, L$ trên cây cầu là câu trả lời cho truy vấn đó.

Độ phức tạp $O\big((n+m).k\big)$.

**Subtask 3 (25%): $m = n - 1$**

Do đồ thị liên thông và $m = n - 1$ nên đồ thị là dạng cây. Vì vậy số đường đi quan trọng giữa $s_i$ và $l_i$ là số cạnh trên đường đi ngắn nhất giữa chúng.

Độ phức tạp thuật toán là $O(k.log_2 n)$.

**Subtask 4 (25%): Không có thêm ràng buộc nào**

Bài toán yêu cầu trả lời các truy vấn, với mỗi truy vấn $(s_i, l_i)$ cần tính số cạnh sao cho khi loại bỏ nó thì không có đường đi giữa hai đỉnh $s_i$ và $l_i$.

Trước hết ta nhận thấy rằng các cạnh cần loại bỏ phải là cầu. Do vậy ta sẽ nén các thành phần liên thông 2-cạnh thành một đỉnh và thu được cây cầu của đồ thị.

Tiếp theo ta sẽ chọn một đỉnh bất kỳ làm gốc của cây cầu và duyệt $dfs$ trên cây từ đỉnh gốc để tính $d[i]$ là số cạnh trên đường đi từ đỉnh gốc đến mỗi đỉnh $i$.

Cuối cùng ta trả lời mỗi truy vấn $(s_i, l_i)$ như sau. Gọi $S, L$ là hai đỉnh trên cây cầu tương ứng với hai thành phần liên thông 2-cạnh chứa hai đỉnh $s_i, l_i$. Kí hiệu $anc$ là tổ tiên chung gần nhất của hai đỉnh $S, L$ trên cây cầu. Khi đó câu trả lời của truy vấn sẽ là:

$$d[S] + d[L] - 2.d[anc]$$

Độ phức tạp thuật toán là $O(n + m + (n + k).log_2 n)$.

## Bài 3. Bữa tiệc (6 điểm)

## Phân bổ điểm

- Có 14% số test ứng với 14% số điểm của bài thỏa mãn: $a_i \geq 0$;
- 14% số test khác ứng với 14% số điểm của bài thỏa mãn: có đúng một số $a_i < 0$;
- 18% số test khác ứng với 18% số điểm của bài thỏa mãn: $k = 1$;
- 12% số test khác ứng với 12% số điểm của bài thỏa mãn: $1 \leq k \leq n \leq 80$;
- 14% số test khác ứng với 14% số điểm của bài thỏa mãn: $1 \leq k \leq n \leq 300$;
- 14% số test khác ứng với 14% số điểm của bài thỏa mãn: $1 \leq k \leq n \leq 2000$;
- 14% số test còn lại ứng với 14% số điểm của bài không có ràng buộc gì thêm.

## Thuật toán

**Subtask 1 (14%): $a_i \geq 0$**

Trong subtask này ta có $a_i \geq 0$, do đó cách tối ưu là phân bổ toàn bộ $n$ đĩa thức ăn này cho tất cả $k$ người. Câu trả lời đơn giản là $a_1 + a_2 + \cdots + a_n$.

Độ phức tạp thuật toán là $O(n)$.

**Subtask 2 (14%): Có đúng một số $a_i < 0$**

Trong subtask này ta có nhiều nhất một số $a_i < 0$. Gọi tổng tất cả các phần tử của mảng là $S$ và tổng tất cả các phần tử ở bên trái, bên phải phần tử $a_i$ tương ứng là $SL, SR$. Hay nói cách khác $S = a_1 + a_2 + \cdots + a_n, SL = a_1 + a_2 + \cdots + a_{i-1}$ và $SR = a_{i+1} + a_2 + \cdots + a_n$.

Khi đó câu trả lời sẽ là $max(S, SL, SR)$.

Độ phức tạp thuật toán là $O(n)$.

**Subtask 3 (18%): $k = 1$**

Trong subtask này ta có $k = 1$. Điều này làm bài toán trở thành tìm một đoạn gồm các phần tử liên tiếp có tổng các phần tử lớn nhất. Chú ý rằng đoạn này có thể rỗng tương ứng với tổng bằng 0. Đây là bài toán cơ bản được xử lý với tổng tiền tố.

Độ phức tạp thuật toán là $O(n)$.

**Subtask 4 (12%): $1 \leq k \leq n \leq 80$**

Gọi $dp[i][j]$ là tổng điểm hài lòng tối đa để chia $i$ đĩa thức ăn đầu tiên cho $j$ người bạn. Để tính $dp[i][j]$, chúng ta lấy giá trị lớn nhất của:

- $dp[i-1][j]$, nếu người thứ $j$ không phân bổ đĩa thức ăn nào;
- $dp[l][j-1] + a_{l+1} + a_{l+2} + \cdots + a_i$, nếu người thứ $j$ được phân bổ các đĩa thức ăn từ $l+1$ đến $i$.

Do đó ta có công thức truy hồi tính $dp[i][j]$ như sau:

- $dp[i][0] = 0$ với mọi $i = 0,1, \ldots, n$;
- $dp[0][j] = 0$ với mọi $j = 0,1, \ldots, k$;
- $dp[i][j] = \max\left(dp[i-1][j], \max\limits_{l=0,1,\ldots,i-1}(dp[l][j-1] + a_{l+1} + a_{l+2} + \cdots + a_i)\right)$ với $i = 1, 2, \ldots, n$ và $j = 1, 2, \ldots, k$.

Câu trả lời bài toán là $dp[n][k]$.

Độ phức tạp thuật toán là $O(n^3.k)$.

**Subtask 5 (14%): $1 \leq k \leq n \leq 300$**

Chúng ta tối ưu thuật toán trong subtask 4 bằng sử dụng tổng tiền tố. Ta tiền xử lý các tổng tiền tố $s_i = a_1 + a_2 + \cdots + a_i$ với mọi $i = 1, 2, \ldots, n$. Chú ý rằng ta thêm tổng tiền tố $s_0 = 0$.

Khi đó:

$$dp[i][j] = \max\left(dp[i-1][j], \max_{l=0,1,\dots,i-1}(dp[l][j-1] + a_{l+1} + a_{l+2} + \cdots + a_i)\right)$$
$$= \max\left(dp[i-1][j], \max_{l=0,1,\dots,i-1}(dp[l][j-1] + s_i - s_l)\right)$$

Độ phức tạp thuật toán là $O(n^2.k)$.

**Subtask 6 (14%): $1 \le k \le n \le 2000$**

Ta viết công thức tính $dp[i][j]$ thành:

$$dp[i][j] = \max\left(dp[i-1][j], \max_{l=0,1,\dots,i-1}(dp[l][j-1] - s_l) + s_i\right)$$

Ta sẽ tính các giá trị $dp[i][j]$ theo thứ tự cột $j$ từ 1 đến $k$. Hơn nữa trong quá trình tính, ta lưu lại giá trị lớn nhất $\max_{l=0,1,\dots,i-1}(dp[l][j-1] - s_l)$.

Độ phức tạp thuật toán là $O(n.k)$.

**Subtask 7 (14%): Không có thêm ràng buộc nào**

Nhận thấy rằng với $i$ cố định, $dp[i][j]$ sẽ tăng dần khi $j$ tăng và giảm dần khi $j$ giảm. Vì vậy ta có thể loại bỏ tham số $j$ và sử dụng tìm kiếm nhị phân để tính $dp[i]$ là tổng tối đa của các đoạn với $i$ phần tử đầu tiên, tức là tổng điểm hài lòng lớn nhất của $k$ người với $i$ đĩa thức ăn đầu tiên.

Ta sử dụng kỹ thuật "Alien trick", với mỗi đoạn sẽ trừ đi một chi phí $\lambda$. Chú ý rằng ở bài toán này, ta thực hiện trừ thay cho cộng $\lambda$ vào mỗi đoạn, vì cộng sẽ dẫn đến $dp[i]$ bị tràn số. Việc giảm/tăng $\lambda$ sẽ làm giảm/tăng hoặc giữ nguyên số người được phân bổ đoạn khác rỗng trong một giải pháp tối ưu. Điều đó cho thấy rằng có thể tìm kiếm nhị phân giá trị nhỏ nhất của $\lambda$ để tạo ra một giải pháp tối ưu có không nhiều hơn $k$ người. Gọi $dp_\lambda[i]$ là tổng tối đa của các đoạn với $i$ phần tử đầu tiên, trong đó mỗi đoạn sẽ được trừ đi một chi phí $\lambda$. Ta có:

$$dp_\lambda[i] = \max\left(dp_\lambda[i-1], \max_{l=0,1,\dots,i-1}(dp[l][j-1] - s_l) + s_i - \lambda\right)$$

Ngoài ra chúng ta sẽ lưu trữ một mảng bổ trợ khác $cnt_\lambda[i]$ là người tối thiểu mà $dp_\lambda[i]$ sử dụng.

Cuối cùng chú ý rằng tổng điểm hài lòng lớn nhất của $k$ người với $n$ đĩa thức ăn cần được điều chỉnh, bởi vì các mỗi đoạn được phân bổ cho một người ăn bị trừ đi một giá trị $\lambda$. Vì vậy câu trả lời của bài toán sẽ là:

$$dp_\lambda[n] + \lambda.cnt_\lambda[n]$$

Độ phức tạp thuật toán là $O(n.log_2 n)$.