Nghệ thuật giải bài

Nghệ thuật giải bài

Bài viết gốc: How to Find a Solution - đăng bởi Dumitru trên Topcoder

Giới thiệu

Với nhiều bài toán trên topcoder (TC), lời giải có thể được tìm ra ngay sau một nốt nhạc. Điều này có được là do những nét tương đồng giữa những bài toán có lời giải giống nhau. Những nét tương đồng này chính là những gợi ý tuyệt vời cho những coder kinh nghiệm để có thể nhận định được lời giải bài toán. Mục tiêu chính của bài viết này là hướng dẫn để các bạn cũng có thể nhận định được chúng.

Những bài toán không yêu cầu kĩ năng đặc biệt (mô phỏng, tìm kiếm, sắp xếp, ...)

Hầu hết các trường hợp, những bài toán này thường chỉ yêu cầu bạn thực hiện từng bước một với những công việc rất đơn giản. Giới hạn thì không quá lớn cũng không quá nhỏ. Và những bài này thường là bài đầu tiên (bài dễ nhất) trong TC SRM. Chúng thường để kiểm tra xem bạn code nhanh và chính xác như nào, và không yêu cầu kiến thức về thuật toán.

Hầu hết các bài toàn sẽ yêu cầu bạn mô phỏng tất cả các bước được nêu ra trong đề.

> **BusinessTasks - SRM 236 Div 1:**
> Có N nhiệm vụ được liệt kê dưới dạng 1 vòng tròn, nhiệm vụ đầu tiên kề với nhiệm vụ cuối cùng. Cho một số n. Bắt đầu từ nhiệm vụ đầu tiên, di chuyển theo chiều kim đồng hồ (từ thứ 1 đến thứ 2 rồi tiếp tục như vậy) và đồng thời đếm từ 1 đến n. Vừa di chuyển vừa đếm, khi đếm đến n, bỏ nhiệm vụ hiện tại ra khỏi danh sách và đếm từ nhiệm vụ tiếp theo. Lặp lại thủ tục này cho đến khi chỉ còn 1 nhiệm vụ. Tìm nhiệm vụ đó.

Với $N \leq 1000$ bài này chỉ là vấn đề của code, không có thuật toán gì đặc biệt - thực hiện từng bước một cho đến khi chỉ còn lại một nhiệm vụ. Những loại bài toán này thường có N nhỏ, vậy nên không cần phải quan tâm đến trường hợp N lớn. Cần nhớ rằng trong topcoder thì 100 triệu phép tính vẫn có thể chạy được.

Có một số bài yêu cầu bạn phải tìm kiếm

> **TallPeople - SRM 208 Div 1:**
> Có một nhóm người được xếp thành một ma trận $R * C$, R hàng, C cột. Nhiệm vụ của bạn là trả về 2 số - Số thứ nhất là chiều cao của người cao nhất trong những người thấp nhất ở mỗi hàng, số thứ hai là chiều cao của người thấp nhất trong những người cao nhất ở mỗi cột.

Như bạn có thể thấy, đây là một bài toán tìm kiếm rất đơn giản. Bạn chỉ cần theo các bước được mô tả trong đề và tìm ra 2 giá trị yêu cầu. Những bài TC khác có thể yêu cầu sắp xếp các bộ theo quy luật nào đó. Bạn có thể làm với thuật toán sort $O(N^2)$ hoặc sử dụng các thư viện có sẵn.

Ví dụ khác:

MedalTable - SRM 209 Div 1

Tìm kiếm theo chiều rộng (Breadth First Search - BFS)

Những bài sử dụng BFS thường yêu cầu tìm số bước ít nhất (hoặc đường đi ngắn nhất) từ điểm đầu đến điểm cuối. Bên cạnh đó, đường đi giữa 2 điểm bất kì thường có chung trọng số (và thường là 1). Phổ biến nhất là dạng bài cho bảng $N*M$, có những ô đi qua được và những ô không đi qua được. Bảng này có thể là mê cung, sơ đồ, các thành phố hoặc các thứ các thứ tương đương. Có thể nói đây là những bài toàn BFS kinh điển (classic). Bởi vì độ phức tạp của BFS là tuyến tính trong hầu hết các trường hợp (N^2 hoặc $NlogN$), giới hạn của N (hoặc M) có thể lớn, lên tới 1 triệu.

> **SmartWordToy - SRM 233 Div 1:**
> Cho một từ gồm 4 chữ cái Latin in thường. Với một lần click bạn có thể đổi bất kì chữ nào thành chữ cái trước hoặc sau nó trong bảng chữ cái (ví dụ 'c' có thể thành 'b' hoặc 'd'). Bảng chữ cái sẽ theo chu kì vòng lặp, tức là 'a' có thể thành 'z' và 'z' có thể thành 'a'.
>
> Cho một tập các rằng buộc, mỗi rằng buộc sẽ là tập các từ bị cấm. Một rằng buộc bao gồm 4 xâu kí tự. Một từ gọi là bị cấm nếu mỗi chữ cái của nó đều xuất hiện trong chính ràng buộc ở vị trí đó, i.e. chữ đầu tiên ở trong xâu đầu tiên, chữ thứ 2 trong xâu thứ 2, ... Ví dụ, có rằng buộc sau "lf a tc e" thì các từ sau bị cấm "late", "fate", "lace" và "face".
>
> Bạn cần phải tìm số lần bấm nút ít nhất để từ ban đầu biến thành từ đích mà không đi qua từ cấm, hoặc trả về -1 nếu không thể biến được.

Hints:

- Coi các từ là các trạng thái. Có nhiều nhất 26^4 từ khác nhau gồm 4 chữ cái.
- Có nhiều cách để biến một trạng thái về một trạng thái khác.
- Chi phí để biến đổi 1 trạng thái luôn là 1 (với 1 lần click)
- Bạn cần phải tìm số bước nhỏ nhất để đến được trạng thái đích từ trạng thái ban đầu.

Mọi thứ đều chỉ ra rằng bài này cần giải bằng BFS. Những điều tương tự như trên sẽ thường thấy trong các bài BFS. Bây giờ, chúng ta cùng xem một BFS bài khá thú vị.

> **CaptureThemAll - SRM 207 Div 2 (3rd problem):**
> Harry đang chơi cờ vua. Anh có một quân mã, còn đối thủ là Joe có 1 quân hậu và 1 quân xe. Hãy tìm số bước nhỏ nhất quân mã phải đi để ăn được cả hậu và xe.

Hints: Mới đầu, có vẻ như bài này là 1 bài quy hoạch động hoặc backtrack. Nhưng nếu để ý kĩ đề bài, ta sẽ có những nhận xét sau:

- Đề cho 1 bảng.
- Quân mã có thể đi từ 1 ô tới các ô mà nó có thể đi được.
- Chi phí ở mỗi bước đi là 1.
- Bạn cần tìm số bước đi nhỏ nhất.

Với những thông tin trên, ta có thể dễ dàng giải được bằng BFS. Đừng bối rối nếu những điểm liên thông là những ô không kề nhau. Hãy nghĩ mỗi ô là một điểm trên đồ thị hay một trạng thái hay bất cứ cái gì bạn muốn.

Chú ý rằng phần lớn những gợi ý về thuật giải BFS là chi phí bằng 1 cho mỗi bước.

Bạn có thể luyện tập thêm về BFS bằng những ví dụ sau:

Ví dụ khác:

RevolvingDoors - SRM 233 Div 1

WalkingHome - SRM 222 Div 1

TurntableService - SRM 219 Div 1

Loang (Flood Fill)

Thỉnh thoảng bạn sẽ gặp phải bài toán cần tới Loang, một kĩ thuật sử dụng BFS để tìm tất cả các điểm có thể đi tới. Điểm khác biệt giữa Loang so với những bài BFS ở trên là bạn không phải tìm chi phí nhỏ nhất.

Ví dụ, có một mê cung, ô 1 là không đi được và 0 là đi được. Ban cần phải tìm tất cả các ô mà có thể đi đến từ ô góc trái trên. Bài này chỉ cần lấy ra một đỉnh đã thăm, nhét tất cả các đỉnh chưa thăm mà kề với đỉnh hiện tại vào queue rồi tiếp tục làm như vậy cho đến khi queue rỗng. Lưu ý rằng nếu số đỉnh lớn, cài đặt bằng DFS (Depth First Search) sẽ có thể bị tràn stack do đệ quy quá sâu và compile code không tăng kích thước stack (xem thêm giải thích ở đây). Tốt hơn hết là nên dùng BFS. Đây là một bài ví dụ:

> **grafixMask** - SRM 211 Div 1:
> Cho một bitmap $400 * 600$. Có một tập hình chữ nhật bao phủ bitmap này (các góc của chúng có tọa độ nguyên). Bạn cần phải tìm ra tất cả các vùng liên tiếp không bị phủ và kích thước của chúng.

Hints:

- Có một map (bảng).
- Một số điểm không đi qua được.
- Cần tìm ra những vùng liên thông.

Những dấu hiệu trên cho thấy bài này cần phải sử dụng Loang

Duyệt trâu và quay lui (Brute Force and Backtracking)

Hai kĩ thuật này được gộp chung vào một loại vì chúng khá giống nhau. Quay lui là kĩ thuật nâng cao và tối ưu hơn so với duyệt trâu. Nó thường sử dụng đệ quy và áp dụng cho những bài có giới hạn nhỏ $(N \leq 20)$

Duyệt trâu (Brute Force)

Đối với những bài duyệt trâu thì giới hạn thường bé. Duyệt trâu bản chất đúng như cái tên của nó, là duyệt hết tất cả các trường hợp và chọn ra cái tốt nhất. Nó rất dễ xây dựng và cài đặt.

> **GeneralChess** - **SRM 197 Div 1:**
> Cho một số quân mã (nhiều nhất là 8) cùng với vị trí của chúng $(-10000 \leq x, y \leq 10000)$. Bạn cần phải tìm các vị trí để đặt thêm một quân mã mà nó có thể ăn được tất cả các quân đã cho.

Hints:

- Đề yêu cầu tìm các trường hợp thỏa mã 1 quy luật nhất định (ở đây là ăn tất cả các quân đã cho).
- Giới hạn bé - chỉ có 8 quân mã.

Giới hạn x, y trong bài trên không quan trọng, bạn chỉ cần thử các vị trí có thể ăn một quân mã và đối với từng vị trí thì xét xem nó có ăn được các quân còn lại không.

Một ví dụ khác:

> **LargestCircle** - **SRM 212 Div 2 (3rd problem):**
> Cho một lưới vuông với một số ô đã bị đánh dấu. Tìm đường tròn lớn nhất không đi qua các ô đã bị đánh dấu và có tâm nằm trên điểm giao của lưới ô vuông (bán kính là số nguyên). Trả lại bán kính của đường tròn.
>
> Kích thước lớn nhất của lưới là 50

Hints: Tại mỗi điểm giao bạn thử từng bán kính một sao cho thỏa mãn đề bài. Rồi chọn ra bán kính lớn nhất trong số chúng.
Phân tích độ phức tạp: Có nhiều nhất là $50 * 50$ ô, bán kính là một số nguyên với max là 25 và bạn cần kiểm tra các điểm trên thuộc đường tròn trong một thời gian tuyến tính. Tổng độ phức tạp sẽ rất bé và bạn có thể yên tâm duyệt trâu.

Ví dụ khác:

Cafeteria – SRM 229 Div 1

WordFind – SRM 232 Div 1

Quay lui (Backtracking)

Kĩ thuật này thường được sử dụng với những bài có giới hạn nhỏ. Các dạng quay lui mà bạn có thể gặp phải là tìm:

1. Tất cả các cấu hình hoặc một tập các cấu hình thỏa mãn một yêu cầu.
2. Cấu hình tốt nhất theo một một tiêu chí nào đó.

> **BridgeCrossing – SRM 146 Div 2 (3rd problem):**
> Có một nhóm người cần sang cầu. Tuy nhiên, do cầu quá cũ, nên một lần chỉ được tối đa 2 người qua cầu. Trời thì tối, chỉ có một cái đèn pin. Hai người cùng đi qua cầu thì thời gian của hai người sẽ là thời gian của người chậm hơn. Do không thể ném đèn pin từ bên này sang bên kia, nên một người sẽ phải quay lại rồi đi với một người khác. Hỏi thời gian ít nhất để tất cả cùng qua cầu.
>
> Có nhiều nhất là 6 người.

Hints:

- Giới hạn chỉ có nhiều nhất 6 người. Quá đủ để có thể sinh ra tất cả các hoán vị.
- Bạn cần tìm ra cách tốt nhất trong số các cách đưa mọi người sang cầu.

Ta sẽ bắt đầu bằng việc đưa 2 người bất kì qua cầu rồi tiếp tục đưa những người còn lại sang. Người quay lại sẽ là người nhanh nhất. So sánh và tìm ra phương án tốn ít thời gian nhất. Mặc dù còn có một thuật giải khác cho bài này nhưng với giới hạn bé như vậy thì chúng ta không cần thiết phải quan tâm.

> **MNS – SRM 148 Div 1:**
> Cho 9 số nguyên trong khoảng từ 0 đến 9. Một ô vuông thần kì là ô vuông mà các số được sắp xếp sao cho tổng mỗi hàng và mỗi cột đều bằng nhau. Tính số cách khác nhau để sắp xếp dãy số đã cho thành một ô vuông thần kì. Hai cách sắp xếp khác nhau nếu chúng khác nhau ở ít nhất 1 vị trí.

Hints: Do chỉ có 9 số nên hoàn toàn có thể sinh ra hết các cách sắp xếp của chúng. Liệt kê các hoán vị của 9 số, đồng thời lưu lại một danh sách các cách sắp xếp khác nhau đã có để kiểm tra.

Ví dụ khác:
WeirdRooks – SRM 234 Div 1

Quy hoạch động (Dynamic Programming)

Để giải quyết cũng như nhìn ra dạng bài này thì chủ yếu dựa vào kinh nghiệm. Thường thì giới hạn trong các bài QHĐ không lớn cũng không nhỏ, độ phức tạp thường là N^2 , N^3 , ... Nếu như giới hạn quá nhỏ (với TC thì thường $N \leq 30$) thì thường không phải là DP. Trong QHĐ thì các bài toán lớn sẽ được chia thành các bài toán nhỏ hơn và tính dựa vào chúng. Để hiểu hơn về QHĐ, bạn có thể tham khảo bài này .

> Thử phân tích một ví dụ đơn giản:
> Cho N đồng xu với giá trị của chúng(V_1, V_2, ..., V_N) và một số S. Tìm số đồng xu nhỏ nhất mà tổng giá trị của chúng bằng S (bạn có thể dùng một đồng nhiều lần) hoặc thông báo không có cách nào như vậy.
> $N \leq 1000$ và $S \leq 1000$

Hints:

- Giới hạn của N và S đều ở mức vừa phải.

- Một trạng thái định nghĩa là số đồng xu nhỏ nhất để tổng bằng 1 giá trị.
- Tổng i phụ thuộc vào các tổng bé hơn j (j < i).
- Bằng cách thêm một đồng xu, ta di chuyển từ trạng thái này sang trạng thái khác.

Tất cả những tính chất trên cho thấy đây là một bài QHĐ.

> **ZigZag – 2003 TCCC Semifinals 3:**
> Một dãy số được gọi là zig-zag nếu các hiệu giữa những liền nhau là một dãy đan dấu (luân phiên âm dương âm dương ...). Dãy có ít hơn 2 số là 1 dãy zig-zag. Cho một dãy số tìm dãy con zig-zag dài nhất. Một dãy con thu được bằng cách xóa bỏ một số số trong dãy ban đầu (hoặc không xóa). Giới hạn $N \leq 50$

Hints:

- Giới hạn của N không lớn không nhỏ.
- $f(i, d)$ là độ dài dãy con zig-zag dài nhất kết thúc tại i, còn d = 0 nếu số trước i bé hơn nó và d = 1 nếu ngược lại.
- $f(i, d)$ được tính bởi $f(j, e)$ $(j < i)$.
- Nếu thêm một số vào trong dãy hiện tại thì ta được một dãy dài hơn.

Như vậy, bài này cũng có những dâu hiệu giống với bài trên và những bài QHĐ khác. Phần khó nhất là xác định được bài đưa ra là QHĐ bằng những dấu hiệu trên. Sau đó bước tiếp theo chỉ là xây dựng thuật toán và cài đặt.

Ví dụ khác:

ChessMetric – 2003 TCCC Round 4

AvoidRoads – 2003 TCO Semifinals 4

FlowerGarden – 2004 TCCC Round 1

BadNeighbors – 2004 TCCC Round 4

Nâng cao (Hard Drills):

Luồng cực đại (Maximum Flow)

Cũng không dễ để có thể xác định được một bài toán sử dụng Luồng. Tuy nhiên, một số dấu hiệu sau có thể giúp bạn:

- Để ý vào giới hạn, những bài này thường có giới hạn phù hợp với $O(N^3)$ hoặc $O(N^4)$
- Đồ thị với các cạnh có trọng số (cho trực tiếp hoặc gián tiếp).
- Cần tìm giá trị lớn nhất của cái gì đó.

> Ví dụ
> Cho một danh sách các ống nước, mỗi ống chỉ cho một lượng nước nhất định được đi qua. Các ông được nối với nhau.

> Tìm lượng nước lớn nhất có thể chảy được từ điểm đầu đến điểm cuối trong một đơn vị thời gian.
>
> $N \leq 100$

Dễ dàng thấy được lượng nước ở mỗi ống là trọng số các cạnh, khớp nối các ống là các đỉnh trong đồ thị. Bạn phải tìm lượng nước lớn nhất chảy từ đỉnh bắt đầu đến đỉnh kết thúc.

Cặp ghép (Optimal Pair Matching)

Dạng bài này thường cho 2 tập và các quy tắc, bạn phải sử dụng các quy tắc này để ghép càng nhiều càng tốt các phần tử ở tập A với các phần tử ở tập B.

> **Parking – SRM 236 Div 1:**
> Cho N cái xe và M chỗ đỗ xe trên một cái bảng hình chữ nhật, trong đó có một số ô không đi qua được. Bạn cần tìm một cách để đưa tất cả các xe vào điểm đỗ xe, sao cho số lớn nhất trong các khoảng cách từ một xe đến chỗ đỗ của nó là nhỏ nhất có thể. Một điểm đỗ xe chỉ cho phép một xe đỗ.

Hints: Nhận xét bài này, ta thấy có 2 tập: xe và điểm đỗ xe, chúng ta cần thực hiện ghép mỗi xe với một điểm đỗ tương ứng. Tồn tại một cạnh giữa 1 xe và 1 điểm đỗ nếu có đường đi giữa chúng và trọng số của cạnh này là đường đi ngắn nhất giữa 2 điểm này. Bước tiếp theo là chia nhị phân khoảng cách dài nhất: ở mỗi bước, xóa đi các cạnh có độ dài lớn hơn C; nếu vẫn có thể ghép tất cả các xe vào điểm đỗ thì chọn một giá trị C nhỏ hơn; ngược lại thì chọn một giá trị C lớn hơn. Khi đã chia nhị phân xong, giá trị C nhỏ nhất sẽ là kết quả cần tìm.

Quy hoạch tuyến tính (Linear Programming) & Thuật toán Simplex

Dấu hiệu nhận biết:

- Cho một tập các vật với khối lượng hoặc số lượng cần đạt được của mỗi vật.
- Cho một danh sách các tập hợp. Mỗi tập bao gồm một số vật với một số lượng nhất định. Mỗi tập có một chi phí.
- Mục đích của bài toán là tìm một tập tối ưu từ các tập trên sao cho tổng số lượng của mỗi vật bằng số lượng cần đạt được.

Ví dụ:

> **Mixture – SRM 231 Div 1:**
> Bạn đang muốn làm ra một hỗn hợp từ các hợp chất khác nhau, mỗi hợp chất cần phải đúng liều lượng. Bởi vì các hợp chất đôi khi không được bán riêng, mà bạn phải mua các hỗn hợp có sẵn mà chứa chúng. Mỗi hỗn hợp lại có một lượng nhất định của mỗi hợp chất cấu thành. Bạn không cần phải mua toàn bộ hỗn hợp mà có thể mua những hợp chất bạn cần với liều lượng tùy chọn. Nhiệm vụ của bạn là tìm ra cách tiêu ít tiền nhất để có thể tạo ra được hỗn hợp mong muốn.

Hints:

- Một tập các vật (hợp chất).
- Một danh sách các tập (các hỗn hợp có sẵn).
- Tìm chi phí nhỏ nhất để tạo ra tập mong muốn từ các tập có sẵn.

Những tính chất trên giống hệt với dấu hiệu nhận biết của dạng Linear Programming.

Đây là 1 kĩ thuật nâng cao, bạn có thể tham khảo thêm ở:

- Wikipedia - Linear Programming
- Simplex algorithm

Kết luận

Để có thể tìm ra được lời giải cho một bài toán bạn cần phải làm nhiều bài tập để luyện khả năng nhận định bài toán cũng như tăng thêm kinh nghiệm. Ngoài ra, còn rất nhiều dạng bài khác mà trong phạm vi bài viết này không thể bao phủ hết được.