Educational Round 2 Tutorial

Trương Trần Nhật Huy

17 - 3 - 2021

Lời mở đầu

Sau một khoảng thời gian khá dài nghiên cứu, học hỏi trong bộ môn Tin học, mình đã tìm tòi, quan sát được khá nhiều những hướng đi trong cách giải quyết một bài toán. Để thành thục được kĩ năng suy nghĩ bài, mình đã phải chú tâm quan sát quá trình suy nghĩ của bản thân, đồng thời nhìn nhận kĩ hơn mô hình bài toán để nhận ra những yếu tố quan trọng ảnh hướng đến cách giải bài.

Những kinh nghiệm mà mình trân quý ấy, đã được cố gắng thể hiện trong cuộc thi vừa rồi. Mặc dù 6 bài toán không đủ để thể hiện hết góc nhìn của bản thân, đây sẽ là bước đầu trong quá trình chia sẻ trải nghiệm của mình đến với các bạn.

Xét về tổng quan, đề thi gồm 6 bài, phân bố chủ yếu ở các chủ đề quy hoạch động và đồ thị. Mình tin rằng, mỗi bài đều chứa đựng những kiến thức quan trọng, những góc nhìn có thể được áp dụng cho nhiều bài toán khác.

Điểm khác biệt lớn nhất trong bài viết này là việc mình sẽ cố gắng chú tâm đến quá trình suy nghĩ để dẫn đến lời giải, đưa ra những lí do tại sao lại nên đi theo một hướng nào đó chứ không chỉ đưa ra lời giải thuần như những bài viết khác.

Mình hy vọng rằng các bạn sẽ mở lòng để sẵn sàng đón nhận một góc nhìn thú vị mới và lấy bài viết làm động lực để thúc đẩy bản thân khám phá tri thức.

Dù đã rất cố gắng, bài viết không thể tránh được sai sót. Nếu có mắc lỗi, hy vọng bạn đọc sẽ bỏ qua và báo giúp mình :p

Mục lục

1 Xâu rối

Bài toán này thực chất chỉ yêu cầu gì thì làm đó. Việc đầu tiên cần làm là phải khôi phục lại vị trí ban đầu của các xâu và ghép chúng lại thành một xâu duy nhất, sau đó chỉ cần duyệt qua các truy vấn và tráo các kí tự.

Để khôi phục lại vị trí ban đầu, ta cần phải thực hiện phép toán bổ sung một xâu vào xâu hiện tại. Để thực hiện điều này có hai cách điển hình là S = S + T; hoặc S += T. Tuy nhiên nếu ai dùng cách đầu tiên sẽ bị TLE! Đó là bởi vì phép + đối với xâu có độ phức tạp là độ lớn của cả xâu mới (chứ không phải chỉ là độ dài của xâu cộng thêm vào), giả sử chỉ thêm vào 1 kí tự vẫn có độ phức tạp $\mathcal{O}(n)$. Trên lí thuyết, cách làm thứ hai vẫn có độ phức tạp như vậy, nhưng do được compiler tối ưu nên trên thực tế chạy nhanh hơn rất nhiều, thông thường sẽ có cỡ độ phức tạp tuyến tính theo độ dài xâu thêm vào. Lí do tại sao xem thêm tại đây

2 ABSum

Nếu như với mỗi i ta trừ A_i cho i, thì bài toán ban đầu sẽ không khác gì bài toán sau đây : Chọn số x sao cho biểu thức sau đây là nhỏ nhất :

$$\sum_{i=1}^{n} |A_i - x| = 1$$

Để bài toán đơn giản, ta sắp xếp lại mảng A. Với mỗi cách chọn x ta có thể viết lại chí là : numberSmaller[x] * x - sumSmaller[x] + sumGreater[x] - numberGreater[x] * x. Nếu ta chọn x thuộc khoảng (A_i, A_{i+1}) bất kì thì kết quả sẽ không tốt hơn so với việc ta chọn $x = A_i$ hoặc $x = A_{i+1}$

Nhận thấy điều đó, các bạn có thể thử x là giá trị của các A_i và so sánh với kết quả hiện tại. Điều này có thể làm được hiệu quả nếu như bạn sắp xếp mảng A theo thứ tự tăng dần (hoặc giảm dần). Độ phức tạp : $\mathcal{O}(n \log n)$

Tuy nhiên có một cách còn ngắn gọn hơn thế nữa. Ta có thể chứng minh được rằng, phần tử trung vị của mảng luôn luôn là một lựa chọn tối ưu cho x. Vậy nên ta chỉ cần tìm xem trung vị của mảng A là phần tử nào và tính chi phí khi x bằng giá trị với phần tử đó. Khuyến khích các bạn tự mình chứng minh sự thật trên! Độ phức tạp của cách làm này chỉ là $\mathcal{O}(n)$ bởi vì ta có thể tìm được phần tử trung vị trong $\mathcal{O}(n)$

3 Xâu mất

3.1 Tóm tắt đề

Cho một xâu kí tự từ 'a' đến 'z'. Bạn cần phải thực hiện nhiều thao tác, mỗi thao tác chọn một xâu con gồm những chứ cái giống nhau và xóa nó ra khỏi xâu. Hỏi cần phải thực hiện ít nhất bao nhiêu thao tác để làm xâu biến mất hoàn toàn.

3.2 Lời giải ngắn gọn

Gọi $f(i,j)$ là số thao tác ít nhất để xóa hoàn toàn xâu con $s[i..j]$. Ta có công thức truy hồi sau đây

$$f(i,j) = min \begin{cases} f(i+1,j)+1 \\ f(i+1,k-1)+f(k,j)\ \forall k \in [i+1,j] \end{cases}$$

Dựa vào công thức truy hồi trên, ta có thể dùng quy hoạch động để tìm ra được đáp án bài toán. Đáp án bài toán là $f(1,n)$. Độ phức tạp $\mathcal{O}(n^3)$

3.3 Phân tích bài toán

Chúng ta hãy bắt đầu bài toán với việc quan sát thật kĩ cấu trúc của bài, chứ không nên máy móc áp đặt bất kì giải thuật gì vào. Ta nhận thấy rằng ở bài toán này, ta gặp phải những khó khăn sau đây :

- Trạng thái của bài toán (xâu kí tự trở thành như thế nào sau một số lần thực hiện thao tác) rất cần thiết để quyết định thao tác tiếp theo nên thực hiện như thế nào. Tuy nhiên trạng thái của bài toán này quá phức tạp, có thể gồm các kí tự mà không liên tiếp nhau ở xâu ban đầu. Để biết được trạng thái của bài toán cần phải biết toàn bộ những thao tác đã thực hiện cùng với thứ tự thực hiện của chúng, điều này là quá cồng kềnh.
- Khi thực hiện một phép xóa xâu con, hai phần còn lại của xâu sẽ có mối quan hệ qua lại với nhau chứ không tách biệt ra hoàn toàn. Ví dụ như khi xóa xâu con "bb" trong xâu "aabbaa". Khi xóa một xâu con ở giữa xâu, hai phần mới của xâu lại tiếp tục tương tác với nhau tạo thành một bài toán mới chứ không phân rã ra thành bài toán con nào.

Tóm lại là, các thao tác có mối quan hệ chặt chẽ với nhau, như là một nút thắt và ta sẽ rất khó để gỡ rối.

> **Remark.** Hãy tưởng tượng hình ảnh của một nút thắt dây giày. Nếu cố gắng gỡ phần nút ở giữa thì sẽ rất khó để có thể gỡ ra được. Tuy nhiên chỉ cần kéo một đầu dây thì nút sẽ được mở ra. Trong một số trường hợp, khi mô hình bài toán phức tạp, ta sẽ cố gắng tìm ra "đầu dây" để giải quyết bài toán. Đó thường là những phần tử đầu hoặc cuối mảng, hay phần tử lớn nhất, nhỏ nhất, hay là nút lá trong cây... Dựa vào những yếu tố đặc biệt ấy, ta có thể dần dần giảm phạm vi của bài toán và cuối cùng giải được toàn bộ bài toán.

Bài toán này là một ví dụ điển hình cho ý kiến trên. Ta thấy rằng khi xóa một xâu con nằm ở giữa, sẽ dẫn đến sự tương tác giữa 2 phần còn lại của xâu. Thế nhưng nếu ta xóa xâu con nằm ở đầu xâu, thì chỉ có duy nhất 1 phần còn lại của xâu, ta không gặp rắc rối với việc những bài toán mới lại tương tác với nhau. Như vậy ta đã tách bài toán ra thành một bài toán con nhỏ hơn. Bởi vậy, ta sẽ tập trung chú ý vào xâu bị xóa mà nằm ở đầu. Một xâu con nằm ở đầu bị xóa khi và chỉ khi kí tự đầu tiên của xâu bị xóa. Vì vậy hãy quan sát khi nào kí tự đầu tiên của xâu bị xóa qua ví dụ sau đây :

Ở ví dụ trên, ta sẽ xóa các xâu theo thứ tự sau : $bb, cc, aaaaaaa, dd$ để có kết quả tối ưu. Ở một xâu kí tự bất kì, kí tự đầu tiên chắc chắn sẽ bị xóa vào một thời điểm nào đó, và nó sẽ được xóa chung với một số kí tự khác (như ta thấy ở ví dụ, nó không nhất thiết phải liên tiếp trong xâu ban đầu) trong lời giải tối ưu. Và để xóa được kí tự đầu tiên, ta phải xóa hết các khoảng trống ở giữa (trong ví dụ, đó là xâu bb, cc). Đó chẳng phải là những bài toán con ư? Nó giống như là bạn đã hoàn toàn tạo ra một xâu mới bb và giải bài toán ban đầu với nó. Sau khi xóa xong xâu con nằm ở đầu, ta lại tiếp tục với một bài toán con nữa (ở ví dụ thì là xâu dd). Điều đặc biệt của những bài toán con này là : nó cũng là một xâu con liên tiếp của xâu kí tự ban đầu! Đây là một dấu hiệu rõ ràng, không thể chối cãi cho một bài toán quy hoạch động!

Gọi tập hợp vị trí những kí tự được xóa chung với kí tự đầu tiên trong lời giải tối ưu là X. Tác giả muốn nhấn mạnh rằng, với một xâu kí tự bất kì, X **luôn tồn tại và cố định**, giống như trong ví dụ, tập hợp X là những kí tự có dấu màu xanh lá ở trên đầu. Để dễ hiểu, bạn đọc có thể tưởng tượng như sau : Có n cái cốc úp xuống, trong n cái cốc ấy có m cốc có tiền, nhiệm vụ của bạn là tìm m cốc ấy.

Để tìm ra được X, ta chắc chắn phải biết được kí tự đứng liền sau vị trí đầu tiên là ở đâu (Theo test ví dụ, đó là vị trí 4). Vì ta chưa biết vị trí của nó ở đâu, ta sẽ duyệt k qua hết mảng và giả sử nó là vị trí thứ hai. Để k là vị trí thứ hai, ta phải xóa hoàn toàn xâu con $s[2..k-1]$). Tiếp theo ta sẽ phải tìm vị trí thứ 3 của X, nhưng việc đó là không cần thiết bởi vì vị trí thứ 3 là vị trí thứ hai của $X_{[k..n]}$ (Cùng định nghĩa với X, nhưng là X của xâu con $[k..n]$). Nếu như ta tìm được vị trí thứ hai cho mọi xâu con, ta có thể giải được với toàn bộ bài toán. Chính vì vậy nên trong công thức truy hồi ở mục **3.2**, ta cập nhật $f(i,j)$ cho $f(i+1,k-1)+f(k,j)$. Ý nghĩa của $f(i+1,k-1)$ là để xóa đi khoảng trống giữa vị trí đầu tiên và vị trí thứ hai trong X, $f(k,j)$ là để tiếp tục tìm những vị trí còn lại trong X.

Chắc hẳn bạn đọc sẽ hỏi, thế còn ta +1 cho một lần thực hiện thao tác ở đâu? Chúng ta sẽ +1 khi đã tìm được đến vị trí cuối cùng trong X (trong ví dụ là vị trí 11, tìm được ở đây nghĩa là $i = 11$). Khi đó ta không cần tìm vị trí thứ hai của X nữa, mà +1 vào chi phí và giải bài toán còn lại (Là xâu dd trong ví dụ). Điều đó giải thích cho $f(i+1,j)+1$.

3.4 Tổng kết

Những thứ đúc kết được từ bài toán trên :

- Khi đối mặt với một bài toán có cấu trúc phức tạp, cố gắng bám víu vào những yếu tố đặc biệt của bài toán có thể sẽ mang đến hiệu quả không ngờ.
- Một trong những cách nhìn quan trọng trong việc giải bài đó là nhìn thẳng vào đáp án, như cái cách mà tác giả đã định nghĩa tập X cùng với cách đi tìm nó.
- Cách tìm X khá thú vị, thay vì tìm vị trí thứ 2, 3, 4,... ta chỉ cần tìm vị trí thứ hai của mọi xâu con! Từ đó có thể dẫn đến một cách làm quy hoạch động. Cách làm này còn có thể áp dụng được vào nhiều bài khác.

4 Điều hướng giao thông

4.1 Tóm tắt đề

Cho một đơn đồ thị vô hướng, liên thông. Tìm một cách định hướng các cạnh sao out-degree của mỗi đỉnh là một số chẵn.

4.2 Phân tích bài toán

Đây là một bài toán constructive trên đồ thị tổng quát. Đồ thị tổng quát chắc hẳn là dạng khó nhất bởi vì những cạnh của nó không nhất thiết phải theo quy luật nào cả. Các cạnh có thể tùy ý liên quan với nhau làm cho việc điều hướng các cạnh khi nhìn vào tổng thể đồ thị rất phức tạp. Nếu nhìn vào toàn bộ đồ thị, thật sự sẽ rất khó để ta có thể xây dựng lời giải vì chúng ta không biết bắt đầu từ đâu và như thế nào.

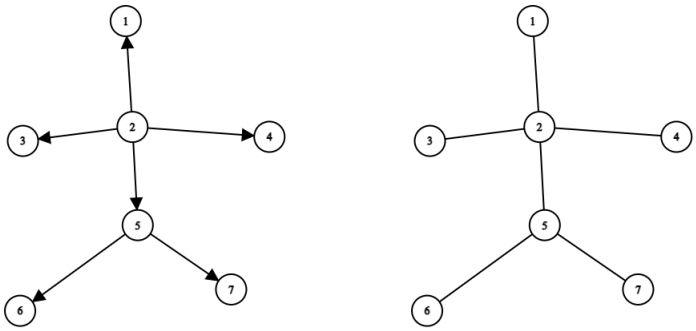
> **Remark.** Đối với các bài toán constructive trên đồ thị tổng quát, bởi vì mô hình của đồ thị vô cùng phức tạp, ta không nhất thiết phải xây dựng kết quả hoàn chỉnh ngay lập tức. Thay vào đó ta có thể giải quyết với những trường hợp "dễ" rồi mới bắt đầu với những trường hợp khó hơn. Đó là bởi vì đồ thị tổng quát rất phức tạp, nếu ta cứ hì hục suy nghĩ trên đồ thị tổng quát thì sẽ rất rối và không mang lại nhiều kết quả. Hãy tưởng tượng giống như bạn đang xây một ngôi nhà, điều đầu tiên là bạn phải xây cái móng trước, chứ không thể cùng lúc xây luôn cả ngôi nhà! Trong khi suy nghĩ những trường hợp dễ, bạn sẽ nhận ra được những tính chất thú vị của bài toán.

Một trong những cách tiếp cận thường thấy sẽ là bắt đầu với việc tìm cách giải quyết cho cây khung. Chúng ta nghĩ đến cây bởi vì nó có cấu trúc không quá phức tạp, các cạnh có một quy luật nhất định chứ không lộn xộn như đồ thị tổng quát, và cũng như tên gọi của nó, cây khung đã chứa mọi đỉnh của đồ thị, là cái "móng" nhà. Chính vì vậy, ta hãy thử giải bài toán này nhưng thêm một điều kiện rằng đồ thị có cấu trúc cây

4.3 Giải quyết bài toán với trường hợp cây

Để thuận tiện trong việc trình bày lời giải, ta gọi p_u là nút cha của u, nút gốc là 1. Ta tạo một mảng f với ý nghĩa sau đây : Nếu cạnh (p_u, u) được điều hướng thành cạnh có hướng từ p_u sang u thì $f[u] = 1$, ngược lại thì $f[u] = 0$. Mảng f cũng được định nghĩa cho nút gốc 1 : Tưởng tượng nút gốc có một nút cha ảo và có cạnh nối với nó, khi đó f được định nghĩa với nút gốc. Tuy nhiên để có một cách điều hướng thỏa mãn, $f[1]$ buộc phải bằng 1 vì nếu ngược lại, ta cần phải có cạnh ảo mới giải quyết được bài toán. Việc làm của ta là cần xây dựng mảng f sao cho cách điều hướng đó thỏa mãn yêu cầu đề bài.

> **Remark.** Hãy nhớ lại hình ảnh nút thắt dây giày ở bài trước. Ta nên bắt đầu với những yếu tố đặc biệt với hy vọng thu nhỏ bài toán. Đối với cây, đó là những nút lá.



Ta nhận thấy rằng, với v là một nút lá thì $f[v] = 1$. Điều này hiển nhiên vì chỉ có duy nhất 1 cạnh nối với nó. Tiếp theo ta sẽ nhìn đến những nút u kề với lá. Bởi vì những cạnh đi xuống nút con của u đã được định hướng, ta hoàn toàn có thể xác định được hướng của cạnh (p_u, u). Hướng của cạnh đó chỉ có thể hoặc là đi lên hoặc đi xuống chứ không còn đường nào khác (ví dụ cạnh (2, 5) chỉ có thể đi xuống).Với cách nhìn từ những nút lá, ta đã nhận ra rằng : khi ta đã tìm được f cho những nút con, ta chắc chắn sẽ tìm được f cho nút hiện tại. Bởi vì vậy, ta sẽ thực hiện giải bài toán từ thấp lên cao, nút nào càng thấp thì giải quyết trước, rồi sau đó mới giải quyết những nút cao hơn sau. Điều này có thể dễ dàng thực hiện bởi thuật toán DFS. Sau khi ta thực hiện xong, việc cuối cùng cần làm là là kiểm tra xem $f[root] = 1$ không.

Chi tiết hơn về cách xây dựng $f[u]$ từ những nút con : Đếm xem out-degree của u hiện tại là bao nhiêu (chỉ xét những cạnh đã điều hướng, là những cạnh nối với nút con). Nếu như outdegree hiện tại là chẵn, thì $f[u] = 1$, ngược lại $f[u] = 0$.

$$f[u] = 1 \oplus f[v_1] \oplus f[v_2] \oplus ... \oplus f[v_k] \text{ với } v_i \text{ là những nút con của } u. \textbf{ (1)}$$

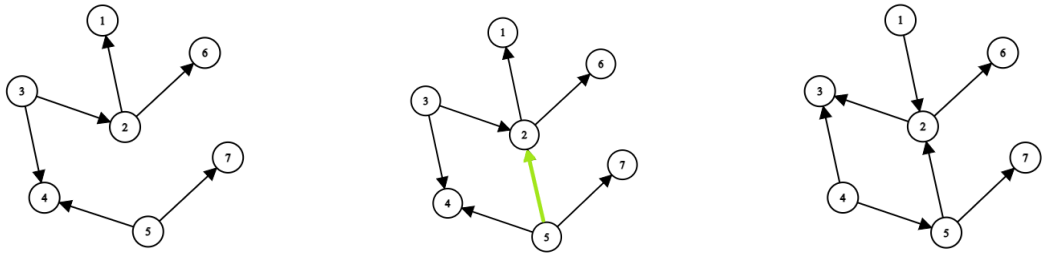
$\oplus$ là phép toán XOR, đây là một phép toán logic, trong C++ thì là dấu mũ. Các em có thể tìm hiểu thêm trên mạng.

Bằng cách này, chúng ta đã giải được trường hợp cây của bài toán. Có thể có bạn sẽ hỏi rằng : Nhỡ như chúng ta không tìm thấy lời giải, nhưng thật ra là có mà ta không tìm thấy thì sao? Nếu nhìn kĩ, ta có thể thấy rằng với mỗi cây, cách định hướng các cạnh là cố định, duy nhất. Điều đó là bởi vì cấu trúc của cây, vị trí các nút lá quyết định tất cả, ta không đưa ra bất kì quyết định gì mà chỉ theo những gì ta bắt buộc phải làm.

Remark. Nếu như quan sát kĩ công thức, ta thấy rằng $f[root] = 1$ khi và chỉ khi số đỉnh là lẻ. Bạn có thể tưởng tượng mỗi nút có một hộp kẹo, ban đầu mỗi hộp có 1 viên kẹo. Mỗi lần thực hiện **(1)**, bạn lấy tất cả viên kẹo ở những nút con để bỏ vào hộp hiện tại. Nếu cứ làm như vậy thì cuối cùng, hộp ở root sẽ có n viên kẹo. Tức là, $f[root] = 1 \oplus 1 \oplus 1... \oplus 1$ (có n lần $\oplus$ 1). Để $f[root] = 1$, n phải là số chẵn!

4.4 Giải quyết bài toán với trường hợp đồ thị tổng quát

Chúng ta mong muốn điều hướng được tất cả các cạnh. Thì điều đầu tiên chúng ta nên nghĩ tới là xử lí đối với cây khung trước. Đó là việc ta đã giải quyết ở 4.3. Bây giờ, từ cách điều hướng đối với cây khung, chúng ta sẽ lần lượt xử lí đối với từng cạnh không nằm trong cây khung. Hãy quan sát điều gì sẽ xảy ra nếu ta thêm một cạnh ngoài vào cây khung :



Lúc này, khi thêm cạnh $(5, 2)$, out-degree của đỉnh 5 đã trở thành số lẻ. Để làm thỏa mãn bài toán, ta cần phải toggle $f[5]$ (toggle nghĩa là từ 0 thành 1, từ 1 thành 0). Theo hiệu ứng di truyền, ta phải toggle các giá trị của $f[u]$ với u thuộc đường đi từ 5 lên gốc. Điều đó đồng nghĩa với việc đảo chiều tất cả các cạnh nằm trên đường đi từ 5 đến 1 (Cạnh ảo từ gốc lên cha cũng được đảo). Lúc này tất cả các đỉnh đều thỏa mãn tính chất của đề, riêng đỉnh gốc thì f chưa bằng 1. Tuy nhiên nếu ta thêm một cạnh mới vào đồ thị nữa thì f của gốc lại bằng 1. Ta nhận thấy rằng nếu ta thêm một cạnh bất kì vào đồ thị mới, chiều của nó không quan trọng bởi vì thứ thật sự thay đổi bài toán chính là trạng thái mới của $f[root]$, còn f của những đỉnh khác thì ta có thể điều chỉnh sao cho phù hợp với điều kiện đề bài. Sau khi thêm một cạnh xong, ta lại có thể tiếp tục thêm một cạnh khác, không liên quan gì đến những cạnh trước, việc chúng ta cần làm là toggle trạng thái của f từ đỉnh ra lên gốc. Tác giả muốn nhấn mạnh rằng, mỗi lần thêm một cạnh, $f[root]$ lại được toggle và là cái duy nhất ảnh hưởng đến việc điều hướng có khả thi hay không.

Tóm lại việc chúng ta cần làm là : Lựa chọn một cây khung bất kì, tìm ra mảng f đối với cây khung đó. Sau đó thêm lần lượt các cạnh vào, chiều của nó không quan trọng. Giả sử cạnh có chiều từ u đến v. Với mỗi cạnh, ta cần toggle giá trị f của các đỉnh từ u lên gốc. Bài toán có kết quả nếu như sau khi thêm mọi cạnh, $f[root] = 1$. Sau đó ta in ra chiều của các cạnh dựa trên f. Việc toggle các đỉnh, không nên làm đối với mỗi lần thêm cạnh mà chỉ cần đánh dấu những vị trí mà từ đó cần update lên gốc, sau đó ta chạy một lần DFS duy nhất để toggle toàn bộ mọi đỉnh cần toggle. Tư tưởng cách làm giống như khi giải prefix sum. Xem chi tiết hơn trong code của tác giả.

Đối với việc chọn cây khung, ta có thể lựa chọn một cây khung bất kì vì dù chọn cái nào đi nữa thì $f[root]$ sau khi xử lí với cây khung cũng không đổi (phụ thuộc vào số đỉnh của cây). Bởi vì ta cần $f[root]$ phải bằng 1 sau khi thêm tất cả các cạnh nên :

- Nếu n chẵn ($f[root] = 0$) thì số cạnh ngoài thêm vào phải là số lẻ
- Nếu n lẻ ($f[root] = 1$) thì số cạnh ngoài thêm vào phải là số chẵn

Từ đó ta suy ra được cách làm của ta cho ra lời giải khi tổng số cạnh là số chẵn. Ta không thể làm tốt hơn được nữa bởi vì khi tổng số cạnh lẻ, hiển nhiên bắt buộc phải có 1 đỉnh out-degree lẻ.

Phân tích được đến đây nghĩa là ta đã hoàn toàn thấu hiểu được bài toán.

4.5 Tổng kết và mở rộng

- Đây tiếp tục là một bài toán có mô hình khá phức tạp, và tác đã trình bày một hướng suy nghĩ bắt nguồn từ những trường hợp đơn giản và những yếu tố đặc biệt. Tác giả hy vọng bạn đọc có thể đón nhận được cách tiếp cận này.
- Đừng ngại nháp! Khi làm bài này, nhờ ngồi nháp và thử thêm từng cạnh từng cạnh vào trong đồ thị, tác giả mới nhận ra được những tính chất hay ho ở **4.4**.

- Thực ra khi làm bài này, không cần phải suy nghĩ kĩ càng như cách tác giả trình bày mà chỉ cần suy nghĩ theo hướng trừu tượng và trực giác. Nhưng bởi vì đây là một bài viết, tác giả cần phải trình bày rõ ràng, đúng đắn nên sẽ hơi dài dòng.
- Cách suy nghĩ từ cây khung và xét những cạnh ngoài là một cách làm được áp dụng rất rộng rãi trong các bài đồ thị tổng quát. Có một chủ đề về nó tên là cây DFS. Đây là một cấu trúc rất hay của DFS mà ít được để ý đến. Tác giả sẽ để nguồn để các bạn tìm hiểu thêm ở đây

5 Sáng kiến vành đai và con đường

5.1 Tóm tắt đề

Cho một đơn đồ thị vô hướng có trọng số. Bạn cần tìm một tuyến đường từ S đến T và một tuyến đường từ U đến V sao cho :

- Tuyến đường từ S đến T là một trong những tuyến đường ngắn nhất từ S đến T
- Độ dài của tuyến đường từ U đến V là nhỏ nhất, với điều kiện : không tính độ dài của những con đường thuộc tuyến đường từ S đến T

5.2 Tính chất quan trọng về đường đi ngắn nhất

Rất nhiều bạn khi học về đường đi ngắn nhất thường "đốt cháy giai đoạn", thừa nhận khá nhiều tính chất lí thuyết. Chính vì vậy nên khi áp dụng, các bạn chỉ có thể làm được những bài cơ bản chứ không thể giải những bài khó. Chủ đề về đường đi ngắn nhất thường thấy trên mạng cũng ít nói đến những lí thuyết quan trọng. Sau đây tác giả xin trình bày một tính chất cực kì thú vị và hay ho về đường đi ngắn nhất.

Note. **Theorem :** Cho một đồ thị vô hướng có trọng số **dương** cùng hai đỉnh S và T. Tập hợp những cạnh nằm trên ít nhất một đường đi ngắn nhất từ S đến T sẽ tạo thành một đồ thị mới. Đồ thị mới đó có cấu trúc của một DAG. Khi đó một đường đi từ S đến T là đường đi ngắn nhất **khi và chỉ khi** nó là một đường đi trên DAG. Có nghĩa là mọi con đường trên DAG là một đường đi ngắn nhất trên đồ thị ban đầu từ S đến T. Ngược lại, một đường đi ngắn nhất trên đồ thị ban đầu từ S đến T là một đường đi trên DAG từ S đến T

Để tìm được DAG đó, ta chạy Dijkstra cho S và T. Sau đó, một cạnh (u, v) trọng số w nằm trong DAG khi và chỉ khi $distS[u] + distT[v] + w = distS[v]$ với $distS[x]$ là đường đi ngắn nhất từ S đến x.

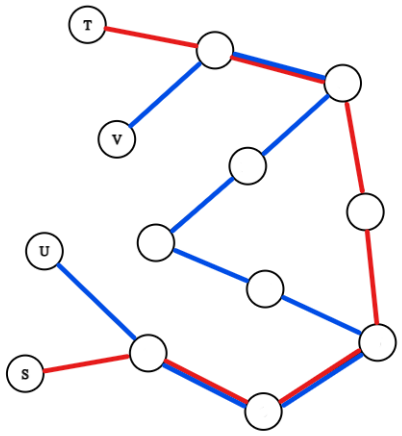
Định lí này rất giá trị trong tin học. Khi ta làm việc với tập hợp những đường đi ngắn nhất từ S đến T, thay vì lúc nào cũng khư khư khái niệm đường đi ngắn nhất, ta chỉ cần làm việc với những đường đi trên một DAG. Từ một đồ thị có cấu tạo phức tạp, cạnh có trọng số liên quan đến đường đi ngắn nhất,... ta đã chuyển bài toán về một đồ thị đơn giản hơn rất nhiều, các cạnh có quy luật và không liên quan gì đến đường đi ngắn nhất nữa.

Ngoài ra còn có những biến thể khác của định lí này như Shortest path tree, Shortest path DAG nhưng rộng hơn DAG mà tác giả trình bày,... được sử dụng khá nhiều trong những bài toán đường đi ngắn nhất.

5.3 Phân tích bài toán

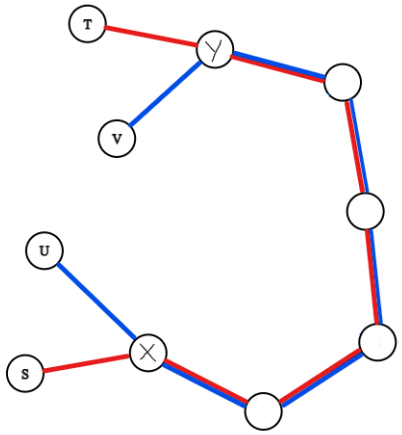
Remark. Hầu như trong tất cả những bài tìm lời giải tối ưu, việc quan sát mô hình tối ưu của kết quả là vô cùng quan trọng, giúp chúng ta đưa ra những nhận xét để giải bài toán.

Trước khi suy nghĩ mọi thứ, ta phải đặt những tính chất quan trọng của đề lên ưu tiên. Ở đây đó tính chất đường đi ngắn nhất từ S đến T .Khi mới tiếp cận bài toán, ta sẽ nhìn nhận kết quả có mô hình như sau :



Cạnh xanh là thuộc tuyến đường từ U đến V, cạnh đỏ là thuộc tuyến đường từ S đến T. Ta thấy rằng đường đi màu xanh sẽ gặp đường đi màu đỏ ở một số con đường, có thể không liên tục.

Khi nhìn vào mô hình đáp án như thế, ta dễ dàng nhận thấy cách để cải thiện kết quả : một khi chúng ta đã đi vào tuyến đường xe buýt (màu đỏ), ta chỉ nên ra khỏi tuyến đường xe buýt một lần duy nhất, khi nào còn sử dụng được thì cứ tiếp tục sử dụng (vì không tính phí) chứ không nên thay thế bằng những con đường khác ngoài tuyến xe buýt. Với nhận xét đó, ta có thể lựa chọn con đường tối ưu hơn như sau :



Ta thấy rằng với nhận xét trên, mô hình bài toán đã đơn giản hơn rất nhiều. Ta có thể phát biểu lại bài toán ban đầu nhu sau : Tìm min của $distU[X] + distV[Y]$ với mọi X, Y sao cho X và Y cùng nằm trên một trong những đường đi ngắn nhất từ S đến T.

Thế nhưng bây giờ, điều quan trọng là làm sao để tìm những cặp (X, Y) như vậy? Nếu như bạn biết được định lí **5.2**, mọi chuyện sẽ đơn giản hơn nhiều. Để tìm được (X, Y), ta cần phải biết được hình hài của những đường đi ngắn nhất từ S đến T là như thế nào. Điều đó đã được thể hiện rất rõ qua định lí nêu trên. Ta có thể dễ dàng suy ra được rằng một cặp (X, Y) thỏa mãn khi và chỉ khi có đường đi từ X đến Y trên DAG được nhắc đến trong định lí.

Với nhận xét trên, bài toán lại càng đơn giản hơn nữa, ta có thể phát biểu như sau : Cho một DAG, tìm min của $distU[X] + distV[Y]$ với mọi X, Y sao cho từ X có thể đến được Y. Lưu ý là bây giờ, $distU$ chỉ là một cái mảng thôi, không còn liên quan đến đường đi ngắn nhất nữa. Để giải được bài toán mới, ta duyệt

qua mọi X, tìm giá trị nhỏ nhất của $distV$ của những đỉnh mà X đến được. Đây là một bài toán cực kì cơ bản và có thể giải bằng quy hoạch động trên DAG. Lưu ý rằng X và Y có thể tráo vai trò cho nhau nên ta phải thực hiện một lần nữa, nhưng giải quyết với những cặp (Y, X)

5.4 Tổng kết

- Bài toán này giới thiệu một tính chất vô cùng quan trọng của đường đi ngắn nhất, và là "hồi chuông cảnh tỉnh" cho những bạn bỏ qua học lí thuyết trong môn đồ thị.
- Một lần nữa, bài toán này đã thể hiện góc nhìn trực diện vào mô hình của lời giải. Tác giả thực sự rất thích cách tiếp cận bài toán như thế này!
- Đối với tác giả, bài toán này là ranh giới giữa việc "dở đồ thị" và "khá đồ thị"!

6 Đa giác và phép chia cắt thú vị

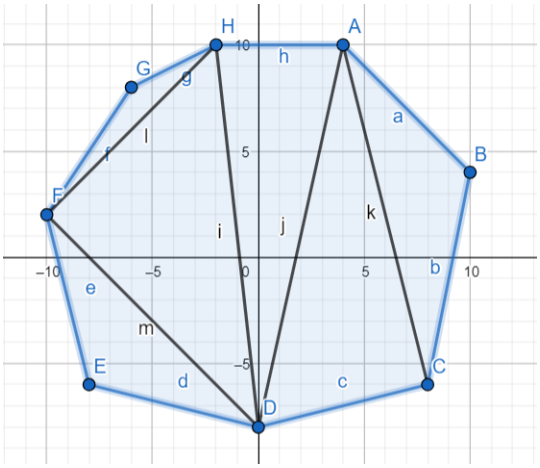
6.1 Tóm tắt đề

Cho một đa giác lồi gồm n đỉnh tọa độ nguyên. Đếm xem có bao nhiêu cách chia đa giác thành những tam giác, sao cho không có điểm nào bên trong nằm trên cạnh, và số điểm trong mỗi tam giác là một số chẵn.

6.2 Phân tích bài toán

Đối với những bài toán đếm kiểu như thế này, gần như ta không thể tránh được một lời giải dùng quy hoạch động. Nếu như là một người chưa rành về quy hoạch động mà tiếp cận bài toán này (ý là tác giả đó), ta sẽ hì hục cố gắng tạo ra một hàm quy hoạch động $f(i, j)$ gì đó, rồi sau đó dựa vào việc vẽ đường chéo để chia đa giác ra làm hai nửa. Tuy nhiên cách làm đó không hợp lí bởi vì : đúng là khi làm như vậy, đa giác sẽ được chia thành hai nửa, tức là hai bài toán con, nhưng ta không cách nào biểu diễn được tập hợp những đỉnh của đa giác. Bạn hãy thử nháp và vẽ các đường chéo mà xem!

> **Remark.** Đối với những bài toán quy hoạch động nói riêng và những bài toán tin học nói chung, việc quan sát những mô hình, cấu trúc của mục tiêu mà mình đang hướng đến vô cùng quan trọng. Điều đó đã được thể hiện ở bài C, và ở cả bài E. Bản chất của các công thức quy hoạch động theo góc nhìn của tác giả là: Để tìm ra cấu trúc của thứ mà mình cần tìm (có thể là lời giải tối ưu, hoặc một mô hình thỏa mãn tính chất gì đó, ở bài này thì là một cách chia đa giác thỏa mãn) ta cần tìm một phần nhỏ của nó (ví dụ như vị trí thứ hai trong X ở bài C, hay với bài này là một tam giác trong $n-2$ tam giác của đa giác), để tìm phần còn lại ta sẽ dựa vào những bài toán con đã tính xong. Bạn đọc hãy ngẫm nghĩ xem! Hầu như các bài quy hoạch động đều hoạt động như vậy.



Một cách chia đa giác thỏa mãn sẽ gồm có $n-2$ tam giác. Mỗi thành phần của nó là một tam giác. Ta thấy rằng mỗi cạnh của đa giác là một cạnh của duy nhất một tam giác. Như tác giả đã nói, khi ta nghĩ đến cách làm quy hoạch động, hãy nghĩ đến việc tìm thử một thành phần của mục tiêu mà mình hướng đến. Khi các bạn cứ hì hục chia bài toán con bằng những đường chéo thì nó chưa tốt lắm bởi vì : Đúng là một cách chia đa giác sẽ gồm những đường chéo nhưng thứ đóng vai trò quan trọng hơn trong bài này đó chính là những tam giác.

Bây giờ ta sẽ đi thử một phần nhỏ của một cách chia thỏa mãn, tức là một tam giác. Ta sẽ thử đối với những tam giác đơn giản trước, tức là những tam giác có ít nhất một cạnh là cạnh của đa giác, giả sử như tam giác HAD. Lúc này bài toán ban đầu sẽ chia ra làm hai phần, mỗi đa giác con là một đoạn liên tiếp các đỉnh của đa giác ban đầu, mỗi đa giác có đúng một cạnh không thuộc cạnh của đa giác (Cạnh i và cạnh j).

Ta tiếp tục chia những đa giác con thành những bài toán con nhỏ hơn. Bởi vì những cạnh như i, j mà để tiếp tục xuất hiện trong đa giác mới thì sẽ rất phức tạp, bởi vì nó không thuộc cạnh của đa giác ban đầu, trực giác mách bảo ta rằng tam giác cần chọn tiếp theo phải có một cạnh thuộc kiểu của i, j. Ta nhận ra một điều thú vị rằng khi ta chia như vậy, các đa giác con tiếp tục được chia nhỏ ra nữa, đồng thời, tập hợp những đỉnh của bài toán con luôn là một đoạn liên tiếp các đỉnh của đa giác (điều mà ta không làm được với cách chia bằng đường chéo). Tóm lại, bằng cách thử những tam giác, ta đã có thể chia bài toán ban đầu thành những bài toán con mà ta có thể biểu diễn được. Việc tiếp theo ta cần làm là tìm cách ghép các bài toán con lại với nhau. Ta có thể dễ dàng rút ra cách kết hợp hai bài toán con là : Số cách chia của đa giác hiện tại = Tích của số cách chia đa giác của hai bài toán con. Bạn đọc cũng đừng quên rằng khi chọn một tam giác con, ta phải chú ý số điểm nằm trong tam giác đó là số chẵn và không có điểm nào nằm trên cạnh của tam giác!

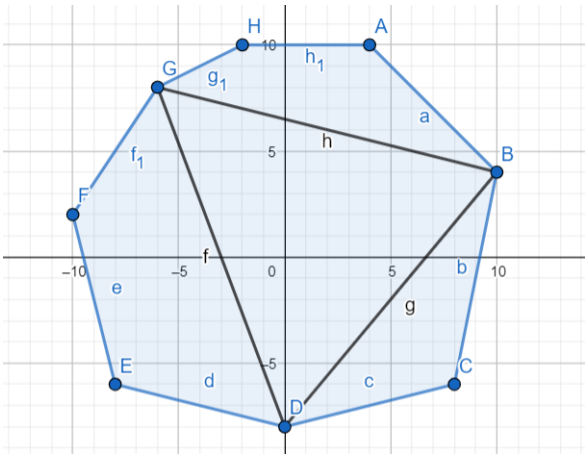
Tiếp theo ta sẽ cố gắng thể hiện ý tưởng trên thành một công thức quy hoạch động cụ thể. Gọi $f(i,j)$ là số cách chia đa giác gồm những đỉnh từ i đến $j (i < j)$ thành những tam giác con thỏa mãn. Kết quả của toàn bộ bài toán sẽ là $f(1,n)$. Công thức thể hiện ý tưởng trên như sau :

$$f(i,j) = \sum_{\substack{k=i+1 \\ \triangle p_i p_k p_j \text{satisfy}}}^{j-1} f(i,k) \cdot f(k,j)$$

Tất nhiên công thức trên vẫn còn thiếu những bài toán cơ bản (base-case), và sẽ để lại để bạn đọc tự suy nghĩ. Để tính được công thức trên, ta cần biết được cách để tính số điểm nằm trong một tam giác.

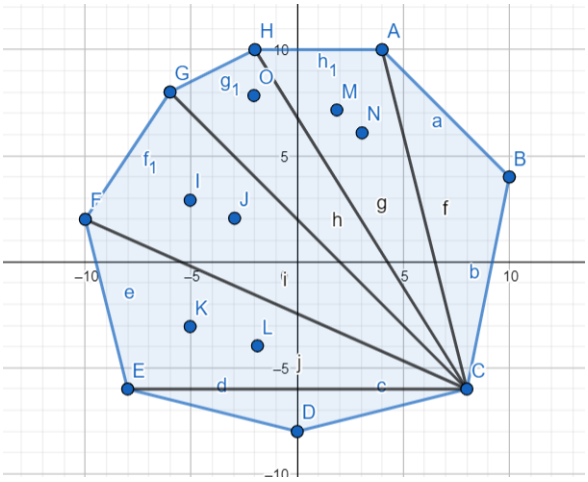
Ngoài ra ta cần phải kiểm tra có điểm nào nằm trên một cạnh của tam giác mà ta lựa hay không. Đây sẽ là bài tập cho bạn đọc vì trình bày nó sẽ tạo ra sự lằng nhằng không cần thiết.

6.3 Tính số điểm nằm trong một tam giác thuộc đa giác



Mặc dù bài toán ban đầu nhìn có vẻ liên quan rất nhiều đến hình học, tuy nhiên anh cho rằng mấu chốt của bài này vẫn là quy hoạch động, còn phần hình học chỉ là để bổ trợ.

Để tính được số điểm nằm trong tam giác GBD, ta chuyển thành bài toán tìm số điểm nằm trong đa giác $GFED, GHAB, BCD$, là những đa giác với tập đỉnh là một đoạn con của đa giác. Tư tưởng chính của cách làm là chia đa giác cần tính ra thành các tam giác nhỏ đã được tính trước, từ những tam giác nhỏ đó cần xây dựng một mảng tiền tố để trả lời những câu hỏi một cách nhanh chóng, hiệu quả.



Những đa giác mà ta cần tìm tổng số điểm nằm ở trong, sẽ có dạng các đỉnh thuộc một đoạn $[i, j]$. Ta sẽ lần lượt giải quyết với những đa giác có i là một đỉnh cố định nào đó. Ở ví dụ trên, ta sẽ giải quyết với những đa giác có i là đỉnh C. Sau này tác giả sẽ gọi là "đỉnh đầu mút" cho dễ nói chuyện nha!

Ví dụ đối với đa giác trên, giả sử ta cần tìm số điểm nằm trong đa giác $CDEFG$ (Về sau sẽ kí hiệu là $[C-G]$), ta cần phải biết được đáp án đối với tam giác CDE, CEF, CFG (Bằng cách nào thì ta sẽ nói sau). Sau khi xây dựng xong đáp án với từng tam giác con, ta phải xây dựng một mảng cộng dồn, gọi là $P[C][X]$ (Tác giả viết in hoa cho dễ hình dung đỉnh, chứ thực ra nó phải là chỉ số) là tổng đáp án của những tam giác con có đỉnh nằm bên trái cạnh CX. Khi đó, giả sử ta cần tìm số đỉnh nằm trong đa giác $[C-H]$, đáp án sẽ là $P[C][H]$. Việc dùng chỉ số trong mảng cộng dồn có thể hơi lằng nhằng với những đỉnh nằm qua ranh giới n và 1, tuy nhiên ta có thể tạm bỏ qua việc đó, quan trọng là ta hiểu bản chất của vấn đề trước.

Bây giờ ta sẽ tìm cách giải quyết với mỗi tam giác con. Thay vì với mỗi tam giác con ta tìm số điểm nằm trong nó, hãy làm ngược lại. Với mỗi điểm, ta hãy tìm xem nó thuộc tam giác con nào. Để làm được việc đó, ta tìm xem điểm đó nằm bên trái đường chéo nào đầu tiên. Ở ví dụ trên, điểm I nằm bên trái đường chéo CG đầu tiên, vì vậy nó thuộc tam giác CGF. Xử lí với mọi điểm, ta có được đáp án với mọi tam giác con. Việc đó có thể giải quyết bằng thuật toán chặt nhị phân, ta cần chặt xem đường chéo nào sẽ là đường chéo đầu tiên nằm bên phải điểm ta đang xét.

Độ phức tạp : $\mathcal{O}(NMlogN)$.

Thực ra độ phức tạp trên đã đủ để AC, nhưng tác giả muốn trình bày cách làm còn tối ưu hơn nữa!

Với mỗi điểm X nằm trong đa giác, nếu ta xét đỉnh đầu mút (đề cập ở trên) theo thứ tự tăng dần, thì đầu mút còn lại của đường chéo mà ta cần tìm với điểm X chỉ có thể tăng dần. Bởi vì vậy ta không cần phải chặt nhị phân, thay vào đó ta chỉ cần lưu lại đỉnh đã xét với mỗi điểm, sau đó cứ tăng dần đỉnh cần xét để kiểm tra. Với mỗi điểm, số lần tăng của đỉnh kiểm tra là $\mathcal{O}(N)$ nên toàn bộ phần xử lí hình học chỉ là $\mathcal{O}(NM)$

Tổng độ phức tạp của bài toán : $\mathcal{O}(N^3 + NM)$

6.4 Tổng kết

- Trong bài này các tác giả đã trình bày thêm một cách tạo công thức quy hoạch động với một bài có cấu trúc phức tạp bằng cách nhìn thẳng vào đáp án
- Tác giả đã trình bày một kĩ năng khá thú vị trong hình học. Kĩ năng này còn được sử dụng trong thuật toán tìm cây khung trên đồ thị mặt phẳng gọi là Delaunay Triangulation (Được nhắc đến trong tài liệu giáo khoa chuyên tin quyển 2 phần cây khung).

7 Bài tập đề nghị

C. Xâu mất

- Bài Baloon.
- COCI 2010 - Zuma

D. Điều hướng giao thông

- Wizard's Tour

E. Sáng kiến vành đai và con đường

- Jzzhu and Cities
- Increasing Cost
- Balkan 12 Shortest Path (Có thể hơi quá khó, nhưng hay).
- Legacy

F. Đa giác và phép chia cắt thú vị

- The Child and Polygon.
- CEOI12 - Race

8 Lời kết

- Bạn đọc khi nghiên cứu tài liệu này sẽ không tránh khỏi sự hiểu nhầm ý của tác giả. Nếu thấy có gì thắc mắc, hãy viết lại câu hỏi, suy nghĩ kĩ bản thân có muốn hỏi hay không, rồi sau đó hãy chia sẻ ý kiến với tác giả trong những buổi gặp mặt online sắp tới.
- Các bài toán tác giả cung cấp nhằm mục đích mang đến những góc nhìn mới, tính chất mới lạ đến với bạn đọc. Có thể bạn đọc sẽ thấy bài toán hơi khó khăn nhưng đừng ngại mở lòng và thử những ý tưởng mới!
- Trong việc học phải lấy tự học làm cốt! Với việc cung cấp những bài toán mới lạ, tác giả muốn nhắc nhở rằng bộ môn tin học luôn chứa đựng những điều thú vị và hấp dẫn. Đừng ngại bước ra khỏi vùng an toàn, hãy tự mình chinh phục những điều mới mẻ! Hãy thực sự tận hưởng việc học, học vì bản thân mình thật sự muốn khám phá ra những điều mình chưa biết để luôn tạo động lực thúc đẩy bản thân phát triển.