WINTER CAMP VOI2020

Đỗ Phan Thuận

Email/Facebook: thuandp.sinhvien@gmail.com
Bộ môn Khoa Học Máy Tính, Viện CNTT & TT,
Trường Đại Học Bách Khoa Hà Nội.

Ngày 13 tháng 11 năm 2020

Outline

Contest 6: PRODUCTIVITY, CVRPOPT, ARCHERY

Contest 5: SPIRIT, ICBUS, CLOPAIR

Contest 4: WATCHING, FARM, FARM

Contest 3: THREEJUG, FACILITY, CANDIES

Contest 2: CNTDIV, STRALT, TELMOV

Contest 1: ALICEADD, SQUARE, TAXI, BCADIF

Contest 6: PRODUCTIVITY, CVRPOPT, ARCHERY

- PRODUCTIVITY
- CVRPOPT
- ARCHERY

Contest 5: SPIRIT, ICBUS, CLOPAIR

Contest 4: WATCHING, FARM, FARM

Contest 3: THREEJUG, FACILITY, CANDIES

Contest 2: CNTDIV, STRALT, TELMOV

Contest 1: ALICEADD, SQUARE, TAXI, BCADIF

PRODUCTIVITY

Một dây chuyền sản xuất có N vị trí làm việc đánh số từ 1 đến N. Có N công nhân để xếp vào làm việc trên các vị trí này. Biết s_{ij} là năng suất làm việc của công nhân i trên vị trí làm việc j của dây chuyền ($i, j = 1,2,..,N$). Cho trước một cách bố trí công nhân đứng làm việc trên các vị trí của dây chuyền, năng suất của dây chuyền theo cách bố trí đã cho sẽ là năng suất của công nhân có năng suất thấp nhất trong số tất cả các công nhân trên dây chuyền.
Yêu cầu: Tìm cách bố trí N công nhân vào làm việc trên N vị trí của một dây chuyền sản xuất sao cho năng suất của dây chuyền là lớn nhất và một công nhân chỉ làm đúng một công việc, một công việc chỉ được làm bởi đúng một công nhân.

Thuật toán 100 điểm

$N \leq 1000$.

- Chia nhị phân giá trị năng suất tối thiểu của công nhân.
- Với mỗi giá trị năng suất tối thiểu, áp dụng thuật toán cặp ghép với các cạnh có năng suất lớn hơn hoặc bằng năng suất tối thiểu.

CVRPOPT

A fleet of K identical trucks having capacity Q need to be scheduled to delivery pepsi packages from a central depot 0 to clients $1, 2, \ldots, n$. Each client i requests $d[i]$ packages. The distance from location i to location j is $c[i, j]$, $0 \leq i, j \leq n$.. A delivery solution is a set of routes: each truck is associated with a route, starting from depot, visiting some clients and returning to the depot for deliverying requested pepsi packages such that:

- Each client is visited exactly by one route
- Total number of packages requested by clients of each truck cannot exceed its capacity
- Each truck must visit at least one client

Goal

- Find a solution having minimal total travel distance

Thuật toán 100 điểm

Mỗi phương án là một cách phân chia các điểm cho các xe và sắp thứ tự thăm cho các điểm đó. Vì vậy để duyệt hết các phương án ta có thể làm như sau:

- Duyệt hết các cách phân chia N điểm cho K xe. Cũng chính là liệt kê các xâu K-phân độ dài N
- Với mỗi cách chia các điểm cho các xe, ta sẽ xử lý riêng cho từng xe. Mỗi xe có một tập các điểm phải đi qua và cần tìm thứ tự đi qua để cực tạ tiểu chi phí. Đây chính là bài toán TSP trên tập các điểm đó.

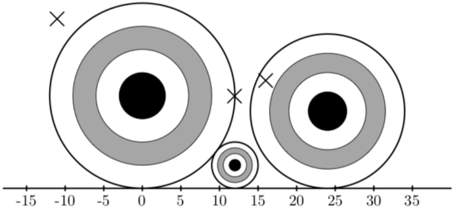
ARCHERY

Ta coi mục tiêu bắn có thể được biểu diễn dưới dạng mặt phẳng 2 chiều, trong đó $y = 0$ là mặt đất. Các mục tiêu có dạng vòng tròn, và tất cả các mục tiêu đều chạm mặt đất. Điều đó có nghĩa, nếu trung tâm của mục tiêu là (x, y) $(y > 0)$, thì bán kính của nó bằng y, để nó chạm vào dòng $y = 0$. Không có hai mục tiêu đồng thời có mặt tại tại bất kỳ thời điểm nào giao nhau (nhưng có thể tiếp xúc nhau).
Ban đầu không có mục tiêu bắn nào. Việc tham gia cuộc thi này có thể được mô tả là n sự kiện gồm 2 loại: hoặc sự kiện mục tiêu mới xuất hiện hoặc sự kiện vận động viên bắn mũi tên vào một điểm. Để đạt được mục tiêu, vận động viên phải bắn đúng bên trong vòng tròn (chạm vào đường biên không tính), khi đó mục tiêu đó sẽ bị xóa đi và vận động viên được thưởng một điểm.

Thuật toán 100 điểm

$1 \leq n \leq 2 \cdot 10^5$

- Với mỗi điểm, kẻ đường thẳng đi qua và vuông góc với trục x, đường thẳng này cắt tối đa $\log_2$(tọa độ tâm lớn nhất) (10^9) do tính chất hình học.
- Áp dụng cây IT quản lý các đoạn chiếu bởi hình tròn xuống trục.
- Độ phức tạp $O(n \log n \log c)$ ($\log c$ vì chia theo bán kính, $\log n$ vì xét tất cả $\log n$ đường tròn trong phạm vi). Phép kiểm tra một điểm có nằm trong đường tròn hay không chỉ mất $O(1)$.



Contest 6: PRODUCTIVITY, CVRPOPT, ARCHERY

Contest 5: SPIRIT, ICBUS, CLOPAIR

- SPIRIT
- ICBUS
- CLOPAIR

Contest 4: WATCHING, FARM, FARM

Contest 3: THREEJUG, FACILITY, CANDIES

Contest 2: CNTDIV, STRALT, TELMOV

Contest 1: ALICEADD, SQUARE, TAXI, BCADIF

SPIRIT

Công suất hiện tại là a, để khôi phục khả năng hoạt động của robot cần tăng công suất lên thành b. Để thay đổi công suất pin của robot, từ Trái đất có thể truyền hai loại tín hiệu: X và Y. Tín hiệu loại X cho phép tăng công suất hiện tại lên 1, tín hiệu loại Y cho phép tăng công suất hiện tại lên 2.
Nếu công suất pin tại một thời điểm nào đó là bội của số nguyên c thì robot sẽ hỏng hoàn toàn và không tương tác với tín hiệu điều khiển nữa.
Yêu cầu: Cho trước các số nguyên a, b, c, hãy xác định số lần gửi tín hiệu tối thiểu để khôi phục được khả năng hoạt động của robot.

Thuật toán 100 điểm

$1 \leq a < b \leq 10^9, 2 \leq c \leq 10^9$, a không chia hết cho c, và b không chia hết cho c.
Chia 2 trường hợp rồi mỗi trường hợp tính theo công thức:

- các số từ a đến *first* và từ *last* đến b không là bội của c;
- bội của c nằm trong khoảng $[first, last]$;
- c chia hết cho 2 –> nhảy bước 2 theo số lẻ
- c lẻ: $res = (last - first)/c * (c+1)/2$ –> nhảy 2 trong mỗi đoạn con giữa 2 bội của c, và có $(last - first)/c$ đoạn

ICBUS

Quốc gia Backoi có N thành phố, mỗi thành phố có một hệ thống xe chạy liên tỉnh khác nhau. Một xe có thể chạy từ thành phố i sang thành phố j nếu như có đường nối trực tiếp giữa hai thành phố này. Các con đường ở đây đều là đường 2 chiều. Mỗi hệ thống xe liên tỉnh có một số luật như sau:

- Hành khách muốn sử dụng hệ thống xe của thành phố i thì bắt buộc phải bắt xe tại thành phố i.
- Giá vé xe của thành phố i là đồng hạng C_i bất kể quãng đường bao xa.
- Hệ thống xe của thành phố i chỉ cho phép chạy tối đa qua D_i thành phố.

Quân là một hành khách muốn đi từ thành phố 1 đến thành phố N. Hãy giúp Quân tìm cách đi sao cho tổng chi phí là thấp nhất.

Thuật toán 100 điểm

$2 \leq N \leq 5000; N - 1 \leq K \leq 10000$

1. Loang tạo lại đồ thị trọng số mới giữa các đỉnh theo hệ thống tính giá vé xe bus rồi áp dụng thuật toán Dijkstra tìm đường di ngắn nhất trên đồ thị mới.
2. Hoặc áp dụng thuật toán Dijkstra trực tiếp trên đồ thị ban đầu, vừa làm vừa tạo cạnh.
3. Lưu ý sử dụng FastI/O.

CLOPAIR

Cho N điểm trên mặt phẳng, hãy tìm một cặp điểm với khoảng cách ơclit nhỏ nhất giữa chúng. Biết rằng không có hai điểm nào trùng nhau và có duy nhất một cặt có khoảng cách nhỏ nhất.

Subtask

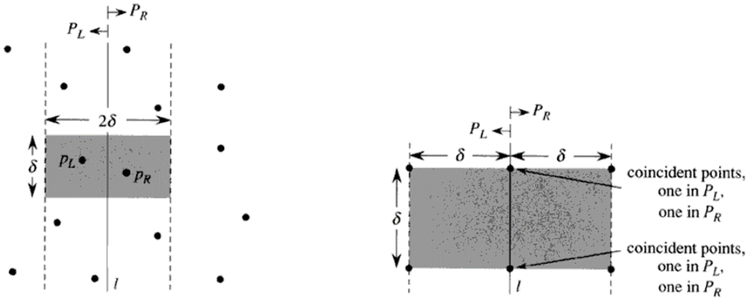
Sử dụng 2 vòng lặp duyệt qua tất cả các cặp điểm cho độ phức tạp $O(n^2)$.

Thuật toán 100 điểm

$2 \leq N \leq 50000$
Sử dụng phương pháp chia để trị. Tại mỗi thủ tục đệ qui đầu vào là tập điểm cần tìm cặp điểm gần nhất:

- CHIA : chia tập điểm thành 2 tập đều nhau (theo toạ độ x, mặt phẳng được chia thành 2 nửa).
- XỬ LÝ: gọi đệ qui cho mỗi tập vừa chia.
- KẾT HỢP: thuật toán đưa ra cặp điểm gần nhất.

Thuật toán 100 điểm



(a) (b)

Thuật toán 100 điểm

- CHIA: chia tập điểm thành 2 tập đều nhau (theo toạ độ x, mặt phẳng được chia thành 2 nửa): P_L, P_R (ĐPT $O(n)$).
- XỬ LÝ: gọi đệ qui cho mỗi tập vừa chia. Kết quả thu được 2 khoảng cách của các cặp điểm gần nhất bên nửa trái và nửa phải: (δ_L, δ_R).
- KẾT HỢP:
  - lấy giá trị nhỏ nhất của 2 khoảng cách thu được ở trên: $\delta = MIN\{\delta_L, \delta_R\}$ (độ phức tạp O(1)).
  - kiểm tra trong các cặp điểm mà một điểm $\in P_L$ và một điểm $\in P_R$) có khoảng cách nhỏ hơn δ không:
    - Tạo một mảng Y' các điểm nằm trong giới hạn 2δ theo trục dọc ở giữa trục chia, sắp xếp Y' tăng dần theo toạ độ y (độ phức tạp $O(n \log n)$ hoặc $O(n)$).
    - Với mỗi điểm trông Y', so sánh điểm đó với 7 điểm tiếp theo (độ phức tạp là $O(7n)$). Nếu tìm được khoảng cách $\delta' < \delta$, cập nhật lại δ theo δ'.

$T(n) = O(1)$, nếu $n \leq 3$.

$T(n) = 2T(n/2) + O(n \log n)$.

Độ phức tạp tổng cộng là $O(n \log^2 n)$.

Các cải tiến

- Tiền xử lý: X là tập điểm đã sắp xếp toàn bộ các điểm theo toạ độ x. Khi đó mỗi lần chia một nửa tập theo toạ độ x thì vẫn giữ được các điểm sắp xếp tăng dần theo x ở mỗi tập.
- Tiền xử lý: Y là tập điểm đã sắp xếp toàn bộ các điểm theo toạ độ x. Điều này giúp cho xác định được tập điểm Y' trong mỗi thủ tục đệ qui sắp xếp theo toạ độ y với độ phức tạp $O(n)$.
- Độ phức tạp $O(n \log n)$:

  $T(n) = O(1)$, nếu $n \leq 3$.
  $T(n) = 2T(n/2) + O(n)$.
- Chỉ cần so sánh với 5 điểm tiếp theo thay vì 6 điểm.
- Mở rộng sang bài toán 3 chiều.

Contest 6: PRODUCTIVITY, CVRPOPT, ARCHERY

Contest 5: SPIRIT, ICBUS, CLOPAIR

Contest 4: WATCHING, FARM, FARM

- WATCHING
- FARM
- COBOX

Contest 3: THREEJUG, FACILITY, CANDIES

Contest 2: CNTDIV, STRALT, TELMOV

Contest 1: ALICEADD, SQUARE, TAXI, BCADIF

WATCHING

Vào thời điểm 0, Bờm sẽ bật TV và bắt đầu xem loạt phim "Avengers" trên kênh A. Nếu bất cứ lúc nào trên kênh truyền hình mà Bờm đang xem có quảng cáo bắt đầu, thì Bờm sẽ chuyển sang kênh kia và xem kênh đó. Nếu Bờm chuyển kênh và cũng có một quảng cáo đang diễn ra vào lúc này, thì anh ấy sẽ không chuyển kênh với hy vọng rằng quảng cáo sẽ sớm kết thúc trên kênh này. Vào thời điểm t, Bờm sẽ tắt TV và đi ngủ.
Cho biết lịch chiếu quảng cáo cụ thể và thời lượng của các quảng cáo trên hai kênh, hãy xác định xem Bờm sẽ xem mỗi bộ phim bao nhiêu đơn vị thời gian.

Thuật toán 40 điểm

Đúng với $n \leq 1000, k, t, a_i, b_j \leq 10^6$.
Sử dụng 2 vòng lặp mỗi vòng trên một kênh và kiểm tra điều kiện chuyển kênh để tính vào tổng thời gian xem mỗi kênh.
Độ phức tạp $O(n^2)$.

Thuật toán 100 điểm

n, m, t và k $(1 \leq n, m \leq 10^5, 1 \leq t \leq 10^{18}, 1 \leq k \leq 10^9)$

1. Sử dụng hai con trỏ chạy tịnh tiến từ trái sang phải trên hai mảng lưu lịch chiếu của hai bộ phim. Độ phức tạp $O(n)$.
2. Hoặc một vòng lặp trên một kênh và kênh còn lại tìm kiếm nhị phân. Độ phức tạp $O(n \log n)$.

FARM

Một trang trại trồng và cung cấp rau sạch ra thị trường cần lập kế hoạch sản xuất cho giai đoạn từ ngày 1 đến ngày n với tổng lượng hạt giống có để gieo trồng là Q. Do đặc tính thời vụ, nên khi gieo trồng 1 đơn vị hạt giống vào ngày i thì sẽ thu được một sản lượng là a_i. Kế hoạch sản xuất sẽ bao gồm các đợt gieo trồng, mỗi đợt sẽ cần tính toán gieo trồng một lượng hạt giống là bao nhiêu và vào ngày nào. Do đặc tính sinh trưởng và thu hoạch của rau nên 2 đợt trồng liên tiếp cách nhau ít nhất K ngày: cụ thể nếu đợt thứ nhất bắt đầu gieo trồng vào ngày thứ i thì đợt gieo trồng tiếp theo sẽ chỉ có thể thực hiện từ ngày $i + K$ trở đi. Ngoài ra, số đơn vị hạt giống gieo trồng trong mỗi đợt không vượt quá hằng số P cho trước.
Hãy tính toán kế hoạch sản xuất sao cho tổng sản lượng rau thu được là lớn nhất.

Thuật toán 50 điểm

$n, Q, P \leq 100$.
Độ phức tạp $O(n^4)$.

Thuật toán 70 điểm

$n, Q, P \leq 1000$.
Qui hoạch động 3 chiều theo n, Q, P. Độ phức tạp $O(n^3)$.

Thuật toán 100 điểm

$1 \le n \le 10^4, 1 \le K \le 10, 1 \le Q, P \le 10^4$;
$a_1, \ldots, a_n (1 \le a_i \le 10^3)$

- Bài toán có thể biến đổi thành tìm tập con có tối đa Q/P phần tử với tổng lớn nhất. Do đó có thể giảm bới một chiều qui hoạch động theo chỉ số P.
- Gọi $dp(i,j)$ là sản lượng lớn nhất làm đến ngày i và đến đợt j. $dp(i,j) = MAX$ (

  $dp(i-1,j)$, nếu ngày i không trồng,
  $dp(i-k,j-1) + a[i])$, nếu trồng ngày i thì phải cách ra K ngày, $1 \le j \le R = Q/P$ là số đợt tối đa.)
- Sau đó truy vết lại để tính tổng sản lượng với mỗi đợt không quá P và tổng không quá Q.

COBOX

Cho n chiếc hộp được đánh số từ 1 đến n. Hộp thứ i có chiều dài a_i , chiều rộng b_i . Hộp i có thể đặt vào trong hộp j nếu i chưa bị chứa bởi hộp nào khác, j đang không chứa hộp nào khác và $a_i < a_j$, $b_i < b_j$. Cần tìm cách lồng các hộp vào nhau sao cho số hộp không bị lồng vào bất kỳ hộp nào là ít nhất. Nếu có nhiều cách lồng các hộp đều là tốt nhất, in ra cách bất kỳ

Thể loại bài

- Bài toán cần tìm Phân rã bè cực tiểu (Minimum Clique Cover - MCC) trên đồ thị comparability.
- Đồ thị comparability là loại đồ thị thoả mãn với bất kỳ bộ 3 đỉnh u, v, w nếu có cung uv và cung vw thì phải có cung vw.
- Phân rã bè cực tiểu MCC = tập độc lập cực đại MIS
- Bài toán dãy con tăng dài nhất chính là bài toán tập độc lập cực đại trên đồ thị mà mỗi đỉnh tương ứng với 1 số, mỗi cung tương ứng với số bên trái nhỏ hơn số bên phải. Ví dụ dãy: 5 3 7 thì đồ thị tương ứng có 3 đỉnh và 2 cung 57 và 37.

Subtask 1

$n \leq 5000$
$O(n^2)$.

Subtask 2

$a_i = b_i$.
Qui hoạch động đơn giản giống bài dãy con tăng dài nhất. Độ phức tạp $O(n \log n)$.

Thuật toán 100 điểm

$1 \leq n \leq 10^5$, $1 \leq a_i \leq 10^9$.

- Sắp xếp tăng theo a.
- Bài toán trở thành: Tìm cách phân hoạch dãy b thành ít nhất các dãy con tăng. Chính là độ dài dãy con không tăng dài nhất. Độ phức tạp $O(n \log n)$.

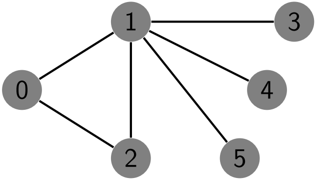
Có thể áp dụng cây BIT2D cho hai chiều của hình hộp cho độ phức tạp $O(n \log^2 n)$. Có thể mất một số test quá thời gian.

Một số bài toán quen thuộc trên đồ thị

- Đây là những bài toán cơ bản rất hay được sử dụng để ra bài trong các kỳ thi
- Thường xuyên được ẩn chứa kín trong phát biểu bài toán
- Xét một số ví dụ

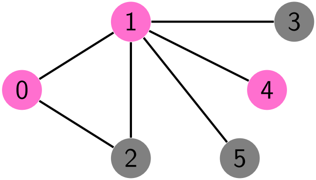
Bài toán phủ đỉnh - MVC

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một phủ đỉnh là một tập con các đỉnh S, sao cho với mỗi cạnh $(u, v) \subset E$, hoặc u hoặc v (hoặc cả hai) thuộc S



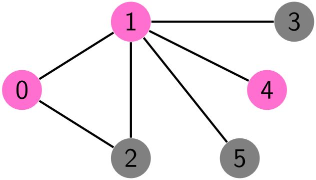
Bài toán phủ đỉnh - MVC

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một phủ đỉnh là một tập con các đỉnh S, sao cho với mỗi cạnh $(u, v) \subset E$, hoặc u hoặc v (hoặc cả hai) thuộc S



Bài toán phủ đỉnh - MVC

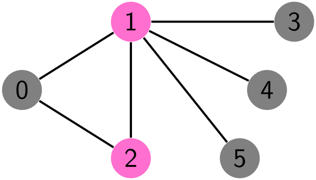
- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một phủ đỉnh là một tập con các đỉnh S, sao cho với mỗi cạnh $(u, v) \subset E$, hoặc u hoặc v (hoặc cả hai) thuộc S



- Hãy tìm một phủ đỉnh có lực lượng nhỏ nhất

Bài toán phủ đỉnh - MVC

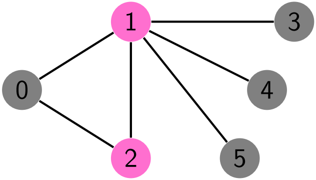
- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một phủ đỉnh là một tập con các đỉnh S, sao cho với mỗi cạnh $(u, v) \subset E$, hoặc u hoặc v (hoặc cả hai) thuộc S



- Hãy tìm một phủ đỉnh có lực lượng nhỏ nhất

Bài toán phủ đỉnh - MVC

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một phủ đỉnh là một tập con các đỉnh S, sao cho với mỗi cạnh $(u, v) \subset E$, hoặc u hoặc v (hoặc cả hai) thuộc S

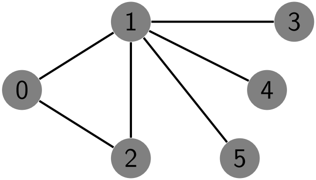


- Hãy tìm một phủ đỉnh có lực lượng nhỏ nhất

- Đây là bài toán NP-khó đối với đồ thị tổng quát

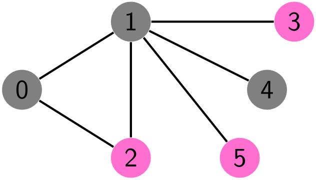
Bài toán tập độc lập - MIS

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tập độc lập là một tập con các đỉnh S, sao cho không có hai đỉnh u, v nào trong S kề với nhau trong G



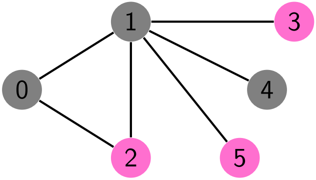
Bài toán tập độc lập - MIS

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tập độc lập là một tập con các đỉnh S, sao cho không có hai đỉnh u, v nào trong S kề với nhau trong G



Bài toán tập độc lập - MIS

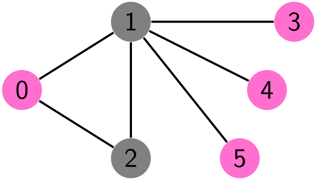
- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tập độc lập là một tập con các đỉnh S, sao cho không có hai đỉnh u, v nào trong S kề với nhau trong G



- Hãy tìm một tập độc lập có lực lượng lớn nhất

Bài toán tập độc lập - MIS

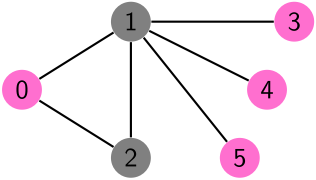
- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tập độc lập là một tập con các đỉnh S, sao cho không có hai đỉnh u, v nào trong S kề với nhau trong G



- Hãy tìm một tập độc lập có lực lượng lớn nhất

Bài toán tập độc lập - MIS

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tập độc lập là một tập con các đỉnh S, sao cho không có hai đỉnh u, v nào trong S kề với nhau trong G

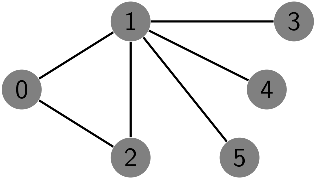


- Hãy tìm một tập độc lập có lực lượng lớn nhất

- Đây là bài toán NP-khó đối với đồ thị tổng quát

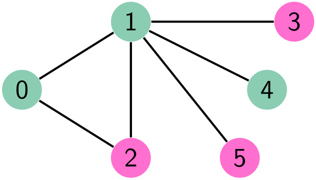
Mối quan hệ giữa MVC và MIS

- Hai bài toán trên có mối liên quan chặt chẽ với nhau
- Một tập con của các đỉnh là một phủ đỉnh khi và chỉ khi tập bù của nó là tập độc lập



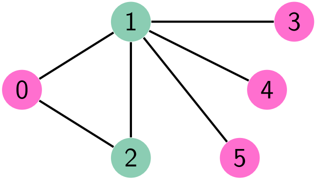
Mối quan hệ giữa MVC và MIS

- Hai bài toán trên có mối liên quan chặt chẽ với nhau
- Một tập con của các đỉnh là một phủ đỉnh khi và chỉ khi tập bù của nó là tập độc lập



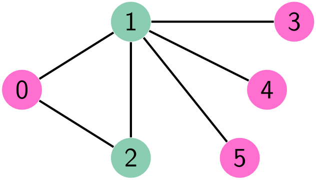
Mối quan hệ giữa MVC và MIS

- Hai bài toán trên có mối liên quan chặt chẽ với nhau
- Một tập con của các đỉnh là một phủ đỉnh khi và chỉ khi tập bù của nó là tập độc lập



Mối quan hệ giữa MVC và MIS

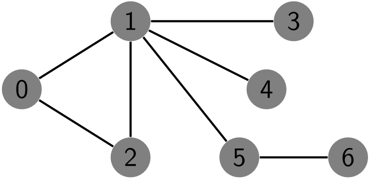
- Hai bài toán trên có mối liên quan chặt chẽ với nhau
- Một tập con của các đỉnh là một phủ đỉnh khi và chỉ khi tập bù của nó là tập độc lập



- Lực lượng của một phủ tập có kích thước nhỏ nhất cộng với lực lượng của tập độc lập có kích thước lớn nhất bằng tổng số đỉnh của đồ thị

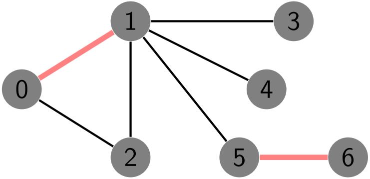
Bài toán ghép cặp

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một cặp ghép là một tập con các cạnh F sao cho mỗi đỉnh kề với tối đa một cạnh trong F



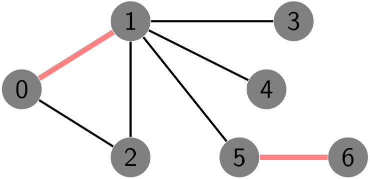
Bài toán ghép cặp

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một cặp ghép là một tập con các cạnh F sao cho mỗi đỉnh kề với tối đa một cạnh trong F



Bài toán ghép cặp

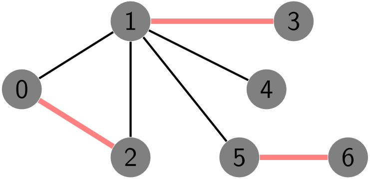
- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một cặp ghép là một tập con các cạnh F sao cho mỗi đỉnh kề với tối đa một cạnh trong F



- Hãy tìm một cặp ghép có lực lượng lớn nhất

Bài toán ghép cặp

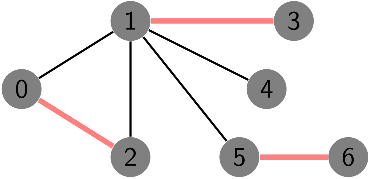
- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một cặp ghép là một tập con các cạnh F sao cho mỗi đỉnh kề với tối đa một cạnh trong F



- Hãy tìm một cặp ghép có lực lượng lớn nhất

Bài toán ghép cặp

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một cặp ghép là một tập con các cạnh F sao cho mỗi đỉnh kề với tối đa một cạnh trong F

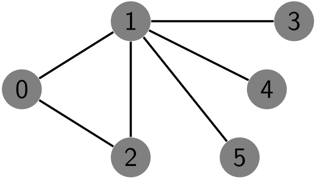


- Hãy tìm một cặp ghép có lực lượng lớn nhất

- Có thuật toán $O(|V|^4)$ cho đồ thị tổng quát, nhưng cài đặt khá phức tạp

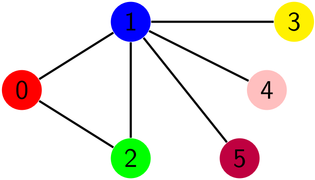
Bài toán tô màu đồ thị

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tô màu của đồ thị là một cách gán các màu vào các đỉnh sao cho các đỉnh kề nhau không cùng màu



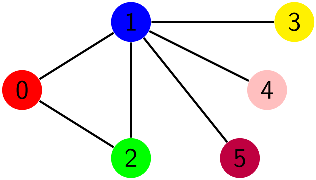
Bài toán tô màu đồ thị

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tô màu của đồ thị là một cách gán các màu vào các đỉnh sao cho các đỉnh kề nhau không cùng màu



Bài toán tô màu đồ thị

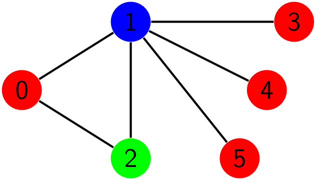
- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tô màu của đồ thị là một cách gán các màu vào các đỉnh sao cho các đỉnh kề nhau không cùng màu



- Hãy tìm một cách tô màu sao cho sử dụng ít màu khác nhau nhất

Bài toán tô màu đồ thị

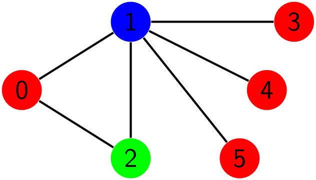
- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tô màu của đồ thị là một cách gán các màu vào các đỉnh sao cho các đỉnh kề nhau không cùng màu



- Hãy tìm một cách tô màu sao cho sử dụng ít màu khác nhau nhất

Bài toán tô màu đồ thị

- Cho đồ thị vô hướng không trọng số $G = (V, E)$
- Một tô màu của đồ thị là một cách gán các màu vào các đỉnh sao cho các đỉnh kề nhau không cùng màu



- Hãy tìm một cách tô màu sao cho sử dụng ít màu khác nhau nhất

- Đây là bài toán NP-khó trên đồ thị tổng quát

Contest 6: PRODUCTIVITY, CVRPOPT, ARCHERY

Contest 5: SPIRIT, ICBUS, CLOPAIR

Contest 4: WATCHING, FARM, FARM

Contest 3: THREEJUG, FACILITY, CANDIES

- THREEJUG
- FACILITY
- CANDIES

Contest 2: CNTDIV, STRALT, TELMOV

Contest 1: ALICEADD, SQUARE, TAXI, BCADIF

THREEJUG

Có 3 bình dung tích A, B, C (lít) với lượng nước ban đầu tương ứng là a, b, c (lít). Mỗi bước được phép đổ đúng d lít từ một bình i sang một bình j khác với điều kiện lượng nước hiện có trong bình i lớn hơn hoặc bằng d và sau khi đổ hết d lít từ bình i sang bình j thì nước trong bình j không bị tràn ra ngoài. Hãy tìm dãy ít nhất các bước đổ nước sao cho lượng nước còn lại ở 1 trong 3 bình đúng bằng T.

Thuật toán 60 điểm

Đúng với $A, B, C \leq 1000$.
Sử dụng phương pháp tìm kiếm theo chiều rộng BFS loang các trạng thái đổ có thể từ 3 bình. Do A, B, C đủ nhỏ nên có thể lưu hết được các trạng thái trong hàng đợi.

Thuật toán 100 điểm

$0 \leq A, B, C, a, b, c, d, T \leq 10^5$.
Bài toán đưa về giải 3 phương trình riêng biệt và đưa ra nghiệm nhỏ nhất trong 3 phương trình đó:
$a + dx = T$
$b + dy = T$
$c + dz = T$
Với mỗi phương trình, sử dụng phương pháp tham lam để thử đổ xem có thoả mãn điều kiện đề bài không.

FACILITY

Một công ty cung cấp dịch vụ cho thuê kho chứa hàng. Công ty nhận được n đơn đặt thuê kho hàng của khách hàng $1, \ldots, n$, mỗi đơn thuê của khách hàng i sẽ bao gồm:

- s_i: ngày bắt đầu thuê
- d_i: số ngày cần thuê
- r_i: số tiền khách hàng i thuê phải trả cho công ty

Tại mỗi thời điểm, kho hàng của công ty chỉ có thể phục vụ cho 1 đơn thuê duy nhất, đồng thời khi một khách hàng kết thúc sử dụng kho hàng thì công ty cần có K ngày để bảo trì kho trước khi cho một khách hàng khác thuê: cụ thể, khách thứ nhất kết thúc thuê vào ngày thứ x thì khách thứ hai chỉ có thể thuê sau ngày thứ $x + K$. Hãy giúp công ty lựa chọn các khách để cho thuê sao cho tổng số tiền thu được là lớn nhất.

Thuật toán 30 điểm

Đúng với 30% số điểm có $n, K \leq 10$. Duyệt toàn bộ.

Thuật toán 80 điểm

Đúng với $n, K, s_i, d_i, r_i \leq 5 \times 10^4$

- Sort lại theo thời điểm bắt đầu s
- Qui hoạch động: gọi $dp(i)$ là tổng số tiền thu được lớn nhất cho thuê đến ngày i:

dp(i) = MAX(

$dp(i-1)$ (nếu không cho thuê ngày i),

$dp(j) + r(i)$ với $j < i$, nếu cho thuê ngày i và ngày gần nhất cho thuê cách i ít nhất K ngày.

)

Độ phức tạp $O(n^2)$.

Thuật toán 100 điểm

$1 \le n \le 10^6, 1 \le K \le 10^9$
Do $dp(k)$ tăng dần nên:

- Sử dụng phương pháp tìm kiếm nhị phân hoặc BIT để tìm k tốt nhất trong đoạn trước i. Độ phức tạp $O(n \log n)$.
- Sử dụng cấu trúc dữ liệu Deque tìm $dp(k)$ tốt nhất trong hàng đợi hai đầu tịnh tiến dần theo i. Độ phức tạp $O(n)$.
- Lưu ý: Với những dữ liệu vào có số lượng lớn ký tự ví dụ khoảng $c \times 10^5$ đến $c \times 10^6$ thì nên sử dụng phương pháp đọc ghi dữ liệu nhanh (FastI/O).

Phương pháp đọc ghi dữ liệu nhanh FastI/O

```


template <typename T> inline void read(T &x){
    x = 0; char c;
    while (!isdigit(c = getchar()));
    do
        x = x * 10 + c - '0';
    while (isdigit(c = getchar()));
}

template <typename T> inline void write(T x){
    if (x > 9) write(x / 10);
    putchar(x % 10 + 48);
}

int main() {
    read(n);
    write(n);
}
```

CANDIES

Kẹo được ban tổ chức chia thành các gói để phân phát cho thí sinh. Số cái kẹo trong mỗi gói kẹo luôn luôn là số Fibonacci, và số lượng gói kẹo mỗi loại mà ban tổ chức có là vô hạn. Giả sử có n thí sinh tham gia cuộc thi. Sau cuộc thi, các thí sinh sẽ được xếp hạng từ 1 đến n (không có hai thí sinh nào cùng hạng). Thí sinh hạng thứ i sẽ nhận được một số gói kẹo sao cho tổng lượng kẹo trong các gói đúng bằng $n - i + 1$. Ban tổ chức đã chọn cách phát kẹo sao cho đối với mỗi thí sinh số gói kẹo nhận được là ít nhất. Có một vấn đề là việc chia kẹo rất tốn thời gian, vì thế những người nhận nhiều hơn hoặc bằng k gói kẹo sẽ được nhận kẹo của mình vào ngày hôm sau.

Yêu cầu: Hãy tính tổng số gói kẹo của những người được nhận kẹo vào ngày hôm sau.

Thuật toán 40 điểm

$n, k \leq 10^5,\ T \leq 100.$
Nhận xét quan trọng: $k > 80$ thì kết quả là 0
Chuẩn bị trước mảng kết quả $n \times 80$. Số gói kẹo cần dùng cho thí sinh n kẹo có thể tính bằng tham lam, trừ dần cho số Fib gần nhất. $F[n, k]$ là số học sinh hạng $\leq n$ được chia kẹo có số gói $\geq k$.

Thuật toán 70 điểm

Đúng với $n > 10^5$, $T \leq 100$.

- Xử lý riêng từng test. Qui hoạch động.
- Mỗi số $n \leq 10^{15}$ tương ứng 1-1 với một dãy nhị phân độ dài 80 mà không có hai bit 1 liên tiếp, tương ứng là tổng các số Fibonacci thứ hạng tương ứng với vị trí các bít 1.
- Bit i giá trị 1 0 là gói kẹo thứ i được chọn hay không.
- 2 bit 1 liên tiếp nghĩa là ghép thành 1 số Fib cao hơn nên không có 2 bit 1 liên tiếp.

Thuật toán 70 điểm

- Gọi dp(i, j, ok) là số xâu nhị phân thỏa mãn, sao cho đến vị trí thứ i đã có j bit 1, và ok = true / false là đã nhỏ hơn hẳn xâu nhị phân tương ứng của n hay chưa. Mỗi test sẽ mất cỡ $80 \times 80 \times 2$ phép tính.
- biến ok là để kiểm soát các xâu sinh ra có thứ tự từ điển $\leq n$.
- $ok = 1$ thì phần đầu xâu đã nhỏ hơn n rồi, nên có thể thêm bit 1 ở đây.
- $ok = 0$ thì phần đầu 2 xâu đang bằng nhau, nên để thêm bit 1 thì $a[i]$ phải =0.

Thuật toán 100 điểm

$1 \leq T \leq 10^5$, $1 \leq n, k \leq 10^{15}$.

- Chuyển về bài toán thứ tự từ điển: Trong số các xâu nhị phân độ dài 80, không có hai bit 1 liên tiếp, và có ít nhất k bit 1, thì xâu n đứng thứ mấy và tổng số bit 1 của các xâu trước nó là bao nhiêu?
- Gọi $dp(i, j) = (a, b)$ với a là số xâu và b là tổng số bit 1 của các xâu đó, sao cho đến vị trí i thì cần thêm ít nhất j bit 1. Độ phức tạp cho bước chuẩn bị hàm $dp()$ là $80 \times 80 \times 2$, và cho mỗi test là 80 để truy vết trong mảng $dp()$.
- Do chuẩn bị trước mảng $dp()$ và chỉ truy vết riêng cho mỗi test trong mảng có sẵn nên ta bỏ được chiều ok đánh dấu vết cho mỗi test như trong thuật toán 70 điểm.

Contest 6: PRODUCTIVITY, CVRPOPT, ARCHERY

Contest 5: SPIRIT, ICBUS, CLOPAIR

Contest 4: WATCHING, FARM, FARM

Contest 3: THREEJUG, FACILITY, CANDIES

Contest 2: CNTDIV, STRALT, TELMOV

- CNTDIV
- STRALT
- TELMOV

Contest 1: ALICEADD, SQUARE, TAXI, BCADIF

CNTDIV

Cho số nguyên dương N, xét số nguyên dương $T = N \times (N+1) \times (N+2)$. Yêu cầu hãy đếm số ước của T^2 mà nhỏ hơn T và không phải ước của T.

Thuật toán 25 điểm

Đúng cho các test giới hạn số N và số lượng bộ test Q bé, khi đó làm trực tiếp, tính từng ước của T và T^2 trong mỗi test.

Thuật toán 100 điểm

- Sàng nguyên tố đến 10^6 để tính các ước nguyên tố $T = 2^{p_2} * 3^{p_3} * \ldots * k^{p_k}$
- Gọi $cntT$ là số ước của T mà $< T$, gọi $cntTT$ là số ước của T^2 mà $<T$ −> $res = cntTT - cntT$
- Lưu ý 1: số ước của T^2 mà $\leq T$ bằng số ước của T^2 mà $> T$
- Lưu ý 2: số ước của T^2 gấp đôi số ước của T

STRALT

1. Chuỗi S có dạng $n(C)$, trong đó n là số tự nhiên nằm ngay trước dấu ngoặc tròn, được biến đổi thành chuỗi D thu được bằng cách lặp liên tiếp n lần chuỗi C. Ví dụ, với chuỗi $5(ab)$ ta có $n = 5$ và thu được dãy $D = ababababab$.
2. Chuỗi S có dạng $[*C]$ được biến đổi thành một chuỗi palindrom (nghĩa là chuỗi đối xứng) có độ dài chẵn, thu được bằng cách ghép chuỗi C với chuỗi ngược của C. Ví dụ, với chuỗi $[*abc]$, chuỗi palindrom thu được có độ dài chẵn là $abccba$.
3. Chuỗi S có dạng $[C*]$ được biến đổi thành một chuỗi palindrom có độ dài lẻ, thu được bằng cách ghép dãy C với chuỗi ngược của C mà bỏ đi ký tự đầu tiên. Ví dụ, với chuỗi $[abc*]$, chuỗi palindrom thu được có độ dài lẻ là $abcba$.

Một chuỗi được coi là đã được giải nén nếu nó chỉ bao gồm các chữ cái viết thường của bảng chữ cái tiếng Anh.

Yêu cầu: Cho chuỗi đã nén S, hãy giúp Bob xác định số lần biến đổi thuộc 3 kiểu trên, cùng với chuỗi T ban đầu trước khi nén của chuỗi S.

Thuật toán

Sử dụng Stack và xử lý xâu để tách từng dạng ra xử lý lần lượt theo độ ưu tiên từ trong ra ngoài đúng với nguyên lý LIFO của Stack: vào sau ra trước.

TELMOV

Cô kỹ sư Alice đang sống ở trong thiên hà VNOI2020. Trong thiên hà này có N hành tinh khác nhau và M kênh vận chuyển hai chiều dạng (x, y, t) cho phép bạn di chuyển từ hành tinh x đến hành tinh y (hoặc ngược lại) trong t giây.
Nhưng Alice nhận thấy phương pháp vận chuyển này rất kém hiệu quả nên đã phát triển một thiết bị cho phép bạn dịch chuyển từ hành tinh x đến bất kỳ hành tinh y nào khác trong P giây với điều kiện bạn có thể đến hành tinh y đó từ hành tinh x chỉ sử dụng tối đa L kênh vận chuyển.
Thiết bị này hiện mới là bản thử nghiệm nên không thể được sử dụng quá K lần. Alice đang ở hành tinh 1 và muốn biết thời gian tối thiểu để đến hành tinh N.
Yêu cầu: Viết chương trình tính thời gian tối thiểu cần thiết để đến được hành tinh N bắt đầu từ hành tinh 1.

Thuật toán 24 điểm

Đúng với các test thoả mãn điều kiện không được sử dụng phép dịch chuyển và tất cả các kênh vận chuyển đều có thời gian $= 1$. Do đó ta chỉ cần sử dụng thuật toán loang BFS đơn giản để giải quyết.

Thuật toán 50 điểm

Đúng với các test thoả mãn điều kiện không sử dụng phép dịch chuyển. Do đó bài toán chính là bài toán tìm đường đi ngắn nhất trên đồ thị có trọng số và ta chỉ cần sử dụng thuật toán Dijkstra đơn giản để giải quyết.

Thuật toán 50 điểm

Đúng với các test thoả mãn điều kiện không sử dụng phép dịch chuyển. Do đó bài toán chính là bài toán tìm đường đi ngắn nhất trên đồ thị có trọng số và ta chỉ cần sử dụng thuật toán Dijkstra kết hợp với Hàng đợi ưu tiên PQ để giải quyết.

Thuật toán 66 điểm

Đúng với các test thoả mãn điều kiện số lượng hành tinh nhỏ hơn hoặc bằng 300. Do đó có thể chia bài toán thành hai bước

1. Xây dựng lại đồ thị bằng cách bổ sung thêm các cạnh đi được bởi cách dịch chuyển thoả mãn các ràng buộc về số lần sử dụng kênh vận chuyển.
2. Áp dụng thuật toán tìm đường đi ngắn nhất trên đồ thị mới xây dựng.

Thuật toán 100 điểm

- Quan sát thấy các điều kiện hạn chế $L, K \leq 10$ là rất nhỏ. Vì vậy ta có thể lưu lại được các trạng thái dịch chuyển trong quá trình xây dựng đường đi ngắn nhất bởi thuật toán Dijkstra vào mảng ba chiều $10000 \times 10 \times 10$.
- Sửa hàm đánh giá tối ưu của thuật toán Dijkstra thành $F(x, y, z)$, nghĩa là thời gian đến x nhỏ nhất sử dụng y lần dịch chuyển và qua z cạnh tính từ lần dịch chuyển gần nhất. Chuyển trạng thái dựa vào điều kiện không quá K lần dịch chuyển giới hạn cho tham số y và sử dụng tối đa L kênh vận chuyển giới hạn cho tham số z.

Contest 6: PRODUCTIVITY, CVRPOPT, ARCHERY

Contest 5: SPIRIT, ICBUS, CLOPAIR

Contest 4: WATCHING, FARM, FARM

Contest 3: THREEJUG, FACILITY, CANDIES

Contest 2: CNTDIV, STRALT, TELMOV

Contest 1: ALICEADD, SQUARE, TAXI, BCADIF

- ALICEADD
- SQUARE
- TAXI
- BCADIF

ALICEADD

- Cho hai số a và b, hãy viết chương trình bằng C/C++ tính số $c = a + b$
- Lưu ý giới hạn: $a, b < 10^{19}$ dẫn đến c có thể vượt quá khai báo `long long`

Thuật toán

- Chỉ cần khai báo a, b, c kiểu `unsigned long long`, trường hợp tràn số chỉ xảy ra khi a, b có 19 chữ số và c có 20 chữ số

1. Tách $a = a1 \times 10 + a0$
2. Tách $b = b1 \times 10 + b0$
3. Tách $a0 + b0 = c1 \times 10 + c0$
4. In ra liên tiếp $a1 + b1 + c1$ và $c0$

SQUARE

Xét dãy số sau:

$0, 0+1, 0+1+3, 0+1+3+5, \ldots, 0+1+3+\ldots+(2n-1), \ldots$.

Đây là dãy được tạo bởi tổng vài số tự nhiên lẻ đầu tiên và các số hạng của dãy đều là số chính phương (tức là bình phương của một số nguyên): $0, 1, 4, 9, \ldots, n^2, \ldots$.

Tổng quát hóa dãy này bằng cách thay số 0 ở đầu bởi một số nguyên k, như vậy ta được dãy:

$k, k+1, k+1+3, k+1+3+5, \ldots, k+1+3+\ldots+(2n-1), \ldots$.

Tuy nhiên khác với trường hợp $k=0$ ở trên, dãy này chỉ có một vài số hạng là số chính phương.

Yêu cầu: Cho trước số nguyên k, cần tìm số nguyên không âm nhỏ nhất sao cho bình phương của nó xuất hiện trong dãy số trên.

Thuật toán

Bài toán đưa về tìm x dương nhỏ nhất mà $k = (x + y)(x - y)$

TAXI

Crab vừa rộng mô hình dịch vụ sang chuyển phát hàng hóa khi xe đang rảnh. Có n gói hàng, gói thứ i muốn chuyển từ vị trí i đến vị trí $i + n$. Cần lập lịch cho xe xuất phát từ vị trí 0, chuyển hết các gói hàng và quay lại vị trí xuất phát. Sức chứa của xe là đủ lớn, do đó gói hàng thứ i sẽ được chuyển nếu ít nhất một lần, lộ trình của xe có đi qua i trước khi đi qua $i + n$. Ví dụ với $n = 3$, lộ trình sau là thỏa mãn: $0 - 1 - 2 - 1 - 5 - 3 - 6 - 4 - 0$
Cho biết độ dài tuyến đường đi lại giữa mọi cặp vị trí, hãy tìm lộ trình của taxi có tổng độ dài các tuyến đường đi qua là nhỏ nhất. Lưu ý, các tuyến đường trong thành phố là đường một chiều nên khoảng cách từ x đến y có thể khác với khoảng cách từ y đến x, và có thể đường đi ngắn nhất x và y không phải là đường đi trực tiếp giữa chúng. Nếu có nhiều lộ trình thỏa mãn có cùng độ dài nhỏ nhất, in ra một lộ trình bất kỳ

Thuật toán

1. Floyd để đường đi thoả mãn bất đẳng thức tam giác
2. QHĐ bitmask hoặc duyệt nhánh cận giống bài toán người du lịch

Bài toán người du lịch (hay người bán hàng)

Bài toán người du lịch là bài toán NP-khó kinh điển nhưng có rất nhiều ứng dụng trong thực tế, đặc biệt ngày nay với ứng dụng của khoa học dữ liệu và phân tích tài chính.

Mỗi khi một hành trình của người bán hàng kết thúc, các dữ liệu sẽ được phân tích bởi các thuật toán trong khoa học dữ liệu, áp dụng vào ngành phân tích tài chính, từ đó có thể 'hoc máy' các kết quả lịch sử để áp dụng vào kế hoạch phát triển tiếp theo.

Bài toán người du lịch

- Cho một đồ thi n đỉnh và giá trị trọng số $c_{i,j}$ trên mỗi cặp đỉnh i, j. Hãy tìm một chu trình đi qua tất cả các đỉnh của đồ thị, mỗi đỉnh đúng một lần sao cho tổng các trọng số trên chu trình đó là nhỏ nhất
- Đây là bài toán NP-khó, vì vậy không tồn tại thuật toán tất định thời gian đa thức nào hiện biết để giải bài toán này
- Thuật toán duyệt toàn bộ đơn giản duyệt qua toàn bộ các hoán vị các đỉnh cho độ phức tạp là $O(n!)$, nhưng chỉ có thể chạy được đến $n \leq 11$
- Liệu có thể làm tốt hơn với phương pháp qui hoạch động?

Bài toán người du lịch

- Không mất tính tổng quát giả sử chu trình bắt đầu và kết thúc tại đỉnh 0

- Gọi $\mathrm{tsp}(i, S)$ cách sử dụng ít chi phí nhất để đi qua toàn bộ các đỉnh và quay trở lại đỉnh 0, nếu như hiện tại hành trình đang ở tại đỉnh i và người du lịch đã thăm tất cả các đỉnh trong tập S

- Bước cơ sở: $\mathrm{tsp}(i, \text{tập mọi đỉnh}) = c_{i,0}$
- Bước chuyển đệ qui:
  $\mathrm{tsp}(i, S) = \min_{j \notin S} \{ \ c_{i,j} + \mathrm{tsp}(j, S \cup \{j\}) \ \}$

Biểu diễn tập hợp bằng Bitmask

- Cho một số lượng nhỏ ($n \leq 30$) phần tử
- Gán nhãn bởi các số nguyên $0, 1, \ldots, n-1$
- Biểu diễn tập hợp các phần tử này bởi một biến nguyên 32-bit
- Phần thử thứ i trong tập được biểu diễn bởi số nguyên x nếu bit thứ i của x là 1
- Ví dụ:
  - Cho tập hợp $\{0, 3, 4\}$
  - `int x = (1<<0) | (1<<3) | (1<<4);`

Biểu diễn tập hợp

- Tập rỗng:

  `0`

- Tập có một phần tử:

  `1<<i`

- Tập vũ trụ (nghĩa là tất cả các phần tử):

  `(1<<n)-1`

- Hợp hai tập:

  `x|y`

- Giao hai tập:

  `x&y`

- Phần bù một tập:

  `~x & ((1<<n)-1)`

Biểu diễn tập hợp

- Kiểm tra một phần tử xuất hiện trong tập hợp:

```
if (x & (1<<i)) {
    // yes
} else {
    // no
}
```

Biểu diễn tập hợp

- Tại sao nên làm như vậy mà không dùng set<int>?
- Biểu diễn đỡ tốn khá nhiều bộ nhớ (nén 32,64,128 lần)
- Tất cả các tập con của tập n phần tử này có thể biểu diễn bởi các số nguyên trong khoảng $0 \ldots 2^n - 1$
- Dễ dàng lặp qua tất cả các tập con
- Dễ dàng sử dụng một tập hợp như một chỉ số của một mảng

Bài toán người du lịch

```
const int N = 20;
const int INF = 100000000;
int c[N][N];
int mem[N][1<<N];
memset(mem, -1, sizeof(mem));

int tsp(int i, int S) {
    if (S == ((1 << N) - 1))  return c[i][0];

    if (mem[i][S] != -1) {
        return mem[i][S];
    }
    int res = INF;
    for (int j = 0; j < N; j++) {
        if (S & (1 << j))
            continue;
        res = min(res, c[i][j] + tsp(j, S | (1 << j)));
    }
    mem[i][S] = res;
    return res;
}
```

Bài toán người du lịch

- Như vậy lời giải tối ưu có thể được đưa ra như sau:
- `printf("%d\n", tsp(0, 1<<0));`

Bài toán người du lịch

- Độ phức tạp tính toán?

Bài toán người du lịch

- Độ phức tạp tính toán?

- Có $n \times 2^n$ khả năng input
- Mỗi input được tính trong thời gian $O(n)$, giả thiết mỗi lời gọi đệ qui chỉ mất $O(1)$
- Thời gian tính tổng cộng là $O(n^2 \times 2^n)$
- Như vậy có thể tính nhanh được với n lên đến 20

Bài toán người du lịch

- Độ phức tạp tính toán?
- Có $n \times 2^n$ khả năng input
- Mỗi input được tính trong thời gian $O(n)$, giả thiết mỗi lời gọi đệ qui chỉ mất $O(1)$
- Thời gian tính tổng cộng là $O(n^2 \times 2^n)$
- Như vậy có thể tính nhanh được với n lên đến 20
- Làm thế nào để đưa ra được chính xác hành trình của người du lịch?

Bài toán người du lịch - Truy vết bằng đệ qui

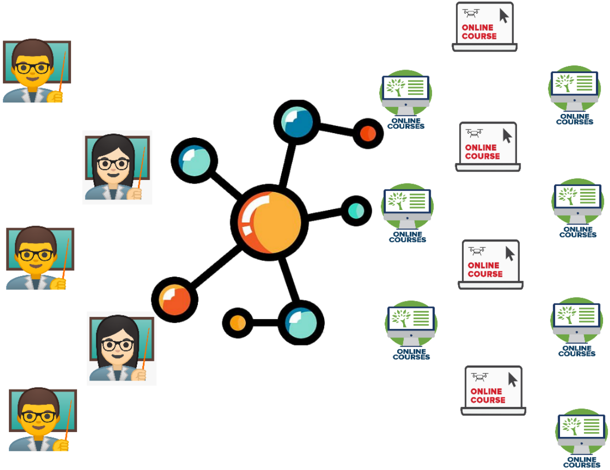
```
void trace_tsp(int i, int S) {
    printf("%d ", i);
    if (S == ((1 << N) - 1))  return;

    int res = mem[i][S];
    for (int j = 0; j < N; j++) {
        if (S & (1 << j))
            continue;

        if (res == c[i][j] + mem[j][S | (1 << j)]){
            trace_tsp(j, S | (j << j));
            break;
        }
    }
}
```

Bài toán người du lịch - Truy vết bằng vòng lặp

```
int answer = tsp(0, 1);
printf("%d\n", answer);
stack<int> s;
s.push(0);
for (int i = 0, S = 1, k = 0; k < n-1; ++k) {
   for (int j = 0; j < n; ++j){
     if ((S & (1 << j))
        && (mem[i][S] == c[i][j] + mem[j][S | (1 << j)])
            s.push(j);
            i = j;
            S = S | (1 << j);
     }
   }
}
while (!s.empty()) {
    printf("%d ", s.back());
    s.pop();
}
```

BCADIF

- Có n môn học và m giáo viên, mỗi giáo viên có danh sách các môn có thể dạy.
- Có danh sách các môn học không thể để cùng một giáo viên dạy do trùng giờ.
- Load của một giáo viên là số môn phải dạy của giáo viên đó.
- **Yêu cầu:** Tìm cách xếp lịch cho giáo viên sao cho Load tối đa của các giáo viên là tối thiểu.


ONLINE COURSE
ONLINE COURSES
ONLINE COURSES
ONLINE COURSE
ONLINE COURSES
ONLINE COURSES
ONLINE COURSE
ONLINE COURSES
ONLINE COURSES
ONLINE COURSE
ONLINE COURSES

Thuật toán

- Sử dụng thuật toán vét cạn, duyệt toàn bộ môn học, xếp giáo viên dạy môn học đó.
- Sử dụng thuật toán nhánh cận:
  - Chọn môn học chưa có người dạy có số giáo viên dạy ít nhất để phân công trước.
  - Nếu phân công cho giáo viên A môn X, mọi môn học trùng lịch với môn X không thể được dạy bởi giáo viên A sau này.
  - Nếu maxCurrentLoad hiện tại lớn hơn minLoad tối ưu thu được trước đó thì không duyệt nữa.

Nhánh cận tốt hơn

- Sử dụng thuật toán nhánh cận:
  - Chọn môn học chưa có người dạy có số giáo viên dạy ít nhất để phân công trước. Sử dụng Hàng đợi ưu tiên **PQ** để lưu tải của các giáo viên, mỗi bước chọn sẽ duyệt nhanh được giáo viên từ tải thấp đến tải cao.
  - Nếu phân công cho giáo viên A môn X, mọi môn học trùng lịch với môn X không thể được dạy bởi giáo viên A sau này. Lưu ý ở bước phân công xong và bước quay lui cần phải cập nhật lại PQ.
  - Nếu maxCurrentLoad hiện tại lớn hơn minLoad tối ưu thu được trước đó thì không duyệt nữa.