A. Cable Car

Đề bài yêu cầu thêm một trong K đường đi cho trước để đường đi từ S đến T là ngắn nhất. Giả sử ta chọn thêm vào đường đi từ (2 chiều) u đến v với trọng số là w. Khi đó, đường đi ngắn nhất từ S đến T sẽ trở thành:

$$\min(dist(S, T), dist(S, u) + w + dist(u, T), dist(S, v) + w + dist(u, T)),$$

với dist(u, v) là đường đi ngắn nhất từ u đến v trong đồ thị ban đầu.

Như vậy, ta cần tính dist(S, u) và dist(u, T) với mọi đỉnh u một cách hiệu quả. Điều này có thể thực hiện được bằng cách sử dụng thuật toán Dijkstra từ đỉnh S và đỉnh T.
Lưu ý: để tính dist(u, T), ta có thể đảo ngược chiều của các cạnh trong đồ thị ban đầu rồi chạy Dijkstra từ T.
Thuật toán chạy trong thời gian $O((n+m) * \log(n+m) + k)$.

B. Conservation Area

Với bài toán này, ta có thể giả sử n không nhỏ hơn 3 (với $n < 3$ ta có thể xét riêng rất dễ dàng). Xét bộ 3 điểm A_i, A_j, A_k đôi một khác nhau bất kì, ta định nghĩa C(i, j, k) là đường tròn nhỏ nhất chứa cả 3 điểm A_i, A_j, A_k. Ta có:

+ C(i, j, k) là đường tròn ngoại tiếp tam giác (A_i, A_j, A_k) nếu tam giác đó nhọn.
+ C(i, j, k) là đường tròn nhận cạnh góc tù là đường kính nếu tam giác đó tù.

Ta sẽ chứng minh kết quả bài toán chính là giá trị lớn nhất của C(i, j, k) (tạm gọi là P). Thật vậy, gọi R là đường tròn nhỏ nhất chứa tất cả các điểm, dễ thấy R >= P. Như vậy ta cần chứng minh P chứa tất cả các điểm.
Giả sử P = C(i, j, k) và tồn tại x sao cho P không chứa A_x. Xét 2 trường hợp:

1. Tam giác (A_i, A_j, A_k) tù tại k => P nhận (A_i, A_j) là đường kính. Vì A_x nằm ngoài P nên tam giác (A_i, A_j, A_x) nhọn => C(i, j, x) là đường tròn ngoại tiếp của (A_i, A_j, A_x). Dễ thấy C(i, j, x) > P, vô lí.
2. Tam giác (A_i, A_j, A_k) nhọn => P là đường tròn ngoại tiếp. Ta cũng có thể chứng minh được một trong 3 đường tròn C(i, j, x), C(j, k, x), C(k, i, x) có bán kính lớn hơn P.

Như vậy, ta có thể duyệt qua tất cả các bộ 3 điểm có thể để tính kết quả trong $O(n^3)$.
Ngoài ra, có một cách giải hiệu quả hơn cho bài toán này là thuật toán Emo Welzl, chạy trong độ phức tạp kì vọng là O(N).

C. DJ Music Mixer

Ta bắt đầu với công thức Quy hoạch động: F(n, k) là số cách chọn ra n bản nhạc với tổng thời gian là k:

$$F(n, k) = sum\{ F(n-1, k-i) * A[i] \}, 0 <= i <= k$$

Trong đó, A[i] là số bản nhạc có thời lượng bằng i.
Nếu ta định nghĩa một đa thức G_n(x) = F(n,0) + F(n,1) * x + F(n,2) * x^2 + … thì G_n(x) = G_(n-1)(x) * G_1(x). Do đó, G_n(x) = G_1(x)^n, với G_1(x) = A[0] + A[1] * x + A[2] * x^2 + …
Ta có thể tính nhanh lũy thừa k của một đa thức bậc n với độ phức tạp O(M(n) * log(k)), với M(n) là độ phức tạp để nhân 2 đa thức bậc n. Sử dụng Number Theoretic Transform cho M(n) = O(n * log(n)). Như vậy độ phức tạp của thuật toán là O(n * log(n) * log(k)), trong đó n = 50000.

D. Drawing Desk

Với K = 1, bài toán trở thành tìm một điểm bị chứa bởi nhiều hình chữ nhật nhất. Đây là một bài toán cổ điển, giải bằng phương pháp sweep-line kết hợp cấu trúc dữ liệu Segment Tree, cụ thể như sau:

+ Ta coi mỗi cột là một “thời điểm”, duy trì một Segment Tree để tìm ô bị chứa bởi nhiều hình chữ nhật nhất tại từng thời điểm.
+ Tại thời điểm 0 (cột 0), tất cả các ô đều có 0 hình chữ nhật chứa chúng.
+ Với mỗi hình chữ nhật có góc trái trên là (x, y), góc phải dưới là (u, v), ta coi nó như 2 truy vấn cập nhật Segment Tree: cập nhật tăng đoạn (y, v) lên 1 đơn vị ở thời điểm x, và giảm đoạn (y, v) đi 1 đơn vị ở thời điểm (u + 1).
+ Với mỗi thời điểm, thực hiện tất cả cập nhật, sau đó tìm ô có giá trị lớn nhất. Đáp số chính là giá trị lớn nhất trong mọi thời điểm.

Lời giải trên có độ phức tạp O((N + M) * log(N)), với M là số hình chữ nhật, N là kích thước bảng.
Với K bất kì, ta có thể coi góc trên trái (x, y) là (x - K + 1, y - K + 1) và giải giống như khi K = 1.

E. Flower Festival

Trước hết, điều kiện để bảng có tính đối xứng cả trục ngang và trục dọc là:

$$A[i][j] = A[n+1-i][j] = A[i][m+1-j] = A[n+1-i][m+1-j]$$

Với mọi i = 1..n, j = 1..m.
Do đó, với mỗi ô (i, j), ta chỉ cần xét 4 ô như trên rồi tìm giá trị xuất hiện nhiều nhất, gán tất cả các ô còn lại bằng giá trị đó. Lưu ý, các ô đối xứng với (i, j) có thể không đủ 4 ô, tuy nhiên điều đó không ảnh hưởng tới thuật toán.

F. Numberland

Thay vì biến đổi từ 1 thành S, ta sẽ biến đổi từ S về 1 với 2 phép biến đổi: giảm S đi 1 đơn vị và hoán vị các chữ số của S.

Nhận thấy để giảm số chữ số của S, ta luôn phải đưa S về dạng 10^k rồi giảm 1 đơn vị. Do đó, ta chỉ quan tâm đến số bước ít nhất để đưa S về dạng 10^k (dễ dàng tính được số bước để giảm từ 10^k về 1 là 10 * (1 + 2 + … + k) - k).
Để đưa S về dạng 10^k, ta xét 3 trường hợp sau:

+ Chữ số cao nhất của S bằng 1: Với những chữ số khác 0 còn lại, ta đưa chúng về hàng đơn vị, giảm về 0 rồi làm với chữ số tiếp theo.
+ Chữ số cao nhất của S khác 1 nhưng tồn tại một chữ số khác bằng 1: Làm tương tự nhưng ta sẽ đưa chữ số 1 về hàng cao nhất trong một bước hoán vị nào đó.
+ Không có chữ số nào của S bằng 1: Đưa chữ số ở hàng đơn vị của S về 1, hoán vị về hàng cao nhất rồi làm tương tự như trên.

G. Robot

Không mất tính tổng quát, giả sử ta đi đến vòng tròn (x1, y1, r1) để tắt TV trước, sau đó đến vòng tròn còn lại để tắt điều hòa (trường hợp còn lại giải tương tự). Dễ thấy cách tối ưu nhất là:
i) Đi đến một điểm A trên đường tròn 1.
ii) Từ điểm A, đi theo đường nối giữa A với tâm của đường tròn 2, đến điểm B trên đường tròn 2.
Như vậy, kết quả sẽ là CA + BA - r2, với C là điểm xuất phát, B = (x2, y2) là tâm đường tròn 2. Như vậy, bài toán trở thành: cho 2 điểm B, C và đường tròn (O, r1), tìm điểm A trên đường tròn sao cho BA + CA nhỏ nhất.
Giới hạn bài toán cho biết B và C nằm ngoài đường tròn O. Do vậy, điểm A tối ưu phải nằm trong cung nhỏ BC của đường tròn O. Mặt khác, khi A chạy trên cung nhỏ BC, có thể thấy rằng hàm F(A) = AB + AC là một hàm có đạo hàm luôn tăng. Như vậy ta có thể sử dụng kĩ thuật chia tam phân để tìm điểm A tối ưu.

H. Save My Files!

Bài toán yêu cầu tìm hoán vị P nhỏ nhất đứng sau hoán vị A theo thứ tự từ điển thỏa mãn: P(P(i)) = i với mọi i = 1..n. Cách giải thông thường của dạng bài này là: cố định vị trí i đầu tiên sao cho P(i) = k > A(i), sau đó tìm hoán vị P nhỏ nhất theo thứ tự từ điển thỏa mãn yêu cầu đề bài và có i vị trí đầu là A(1), A(2), …, A(i-1), k.
Để tìm hoán vị thỏa mãn nhỏ nhất với một tiền tố cho trước, ta làm như sau:

+ Xét một vị trí i đã được gán, nếu P(i) = j khác i thì ta gán P(j) = i. Nếu mâu thuẫn với các vị trí đã được gán trước đó thì kết luận không dựng được.
+ Với các vị trí j chưa được gán ở bước 1, ta gán P(j) = j.

Độ phức tạp thuật toán sẽ là O(N^3) (O(N) cho mỗi bước kiểm tra).

I. Space-Time Travel

Ta cần tính giá trị của:

$$\Sigma \{i = 1..N\} \{j = 1..M\} (i - j) * |B_j - A_i|$$

$$= \Sigma \{B_j < A_i\} (i - j) * (A_i - B_j) - \Sigma \{B_j > A_i\} (i - j) * (A_i - B_j)$$

$$= S - T$$

Xét giá trị của S:

$$S = \Sigma \{B_j < A_i\} (i - j) * (A_i - B_j)$$

$$= \Sigma \{B_j < A_i\} (i * A_i + j * B_j) - \Sigma \{B_j < A_i\} (i * B_j + j * A_i)$$

Để tính S, ta có thể lập một vài mảng tổng dồn:

+ S1[x] = Tổng các i * A_i với $A_i > x$.
+ S2[x] = Tổng các j * B_j với $B_j < x$.
+ S3[x] = Tổng các i với $A_i > x$.
+ S4[x] = Tổng các j với $B_j < x$.

Vì các giá trị A_i, B_j đều nằm trong khoảng $[1..10^4]$ nên việc tính các mảng này có thể thực hiện dễ dàng trong độ phức tạp $O(N + M + \max\{A_i\})$.

Việc tính T có thể thực hiện tương tự như tính S.

J. Treasure Box

Giả sử k là số nguyên không âm lớn nhất sao cho phần nguyên dưới của 2 phép chia A / 10^k và B / 10^k bằng nhau, thì đáp án sẽ là k.

Thật vậy: theo giả thiết, giả sử $A = Q * 10^k + r$, $B = Q * 10^k + s$ ($0 <= r, s < 10^k$) thì mọi số nằm giữa A và B đều có dạng $Q * 10^k + t$, với $0 <= t < 10^k$. Do đó, 2 số bất kì nằm giữa A và B không thể khác nhau quá k chữ số.

Mặt khác, vì k là số lớn nhất thỏa mãn điều kiện trên nên $A = Q_1 * 10^{k-1} + r_1$, $B = Q_2 * 10^{k-1} + r_2$, trong đó $Q_1 < Q_2$. Ta có thể chọn $X = Q_2 * 10^{k-1} - 1$, $Y = Q_2 * 10^{k-1}$ thì $A <= X < Y <= B$ và X khác Y đúng k chữ số.