DISCO presents ディスカバリーチャンネル コードコンテスト 2017 本戦 解説

chokudai, yosupo

平成 29 年 11 月 2 日

A: 正方形のチップ 2

重なる可能性のある正方形を全部試して、それぞれ円に含まれているかを判定すればよいです。

例えば円の中心を、200mm : $(100, 100)$ と 300mm : $(150, 150)$ に置けば、試すべき正方形の座標は $(K \times i \sim K \times (i+1), K \times j \sim K \times (j+1)), (0 \leq i, j < 300/K)$ です (問題文の図の、左下を原点に合わせたとイメージしてください)。

正方形が円に含まれているかどうかは、4 隅がすべて円に含まれているかどうかを調べればよいです。

B: GCD ロボット

$\gcd(Z, a_i) = g_i$ とします。

この問題で聞かれているのは、

すべての i について $\gcd(X, a_i) = g_i$ となる X のうち、最小のものです。

まず、$\gcd(X, a_i) = g_i$ より、X は g_i の倍数である必要があります。

つまり、少なくとも X は $Y = \mathrm{lcm}(g_1, g_2, ..., g_N)$ の倍数です。

また、ここから Z も Y の倍数であることがわかるので、少なくとも $\gcd(Y, a_i) \leq \gcd(Z, a_i) = g_i$ であり、また Y の定義より $\gcd(Y, a_i) \geq g_i$ であるため、$\gcd(Y, a_i) = g_i$ です。

以上より、Y は答えの条件を満たし、さらに最小であるためこれが答えです。

別解

gcd, lcm 系の操作は、整数を (素因数 2 の個数, 素因数 3 の個数, 素因数 5 の個数, ...) という無限次元のベクトルに変換すると見通しがよくなることが多いです。

実は、gcd はこのベクトルの各要素について min を取る操作であり、lcm は max を取る操作です。

つまりこの問題は、あるベクトルの列 a_i とベクトル X が与えられ、以下の条件を満すベクトル Z を探せ

- すべての i, j について、$\min(X_j, (a_i)_j) = \min(Z_j, (a_i)_j)$

という問題になります。

明らかにこの条件はベクトルの各要素に対し独立なので、要素ごとに考えると、結局 $Z_j = \max(\min(X_j, (a_i)_j))$ とすれば良いことがわかります。

よってこの操作を元の整数に戻すと、$Z = \mathrm{lcm}(\gcd(X, a_i))$ となります。

C: グラフいじり

この問題では、単純なサイクル (=同じ頂点や辺を 2 度以上使わないサイクル) だけ考えています。しかし、実は単純とは限らないサイクル (=同じ頂点や辺を 2 度以上使うサイクル) も考えても答えは変わりません。これは、単純とは限らないサイクルはいくつかの単純なサイクルに分割できることから示せます。

頂点 u から頂点 v の、長さ d の辺を辺 (u, v, d) と書くことにします。

まず、辺の長さを変えることは考えず、与えられたグラフが条件を満たすかを判定するアルゴリズムを考えます。

条件を満たすならば、辺 (u, v, d) があると、v から u へのパスの長さは必ず $-d$ であるはずです。よって、逆向きの辺 $(v, u, -d)$ をグラフに足してもよいです。なぜならば、もしこの辺を使う長さ非 0 のサイクルがあるなら、そこを元からあるパスで置き換える事を考えると、どうせ元のグラフでも条件を満たさないことがわかるからです。

ここで dfs 木を考えます。この問題は有向グラフですが、先述の操作で逆辺を足すことで、dfs 木の性質はまるで無向グラフのようになります。つまり、dfs 木に含まれない辺 (u, v, d) について、u と v は (どちらが先祖かはわかりませんが) 先祖と子の関係になります。

よって、dfs 木を一つとります、根は頂点 1 とします。

この dfs 木での頂点 i までの距離を h_i とします。すると、全ての辺 (u, v, d) について、$d = h_v - h_u$ となっている必要があることがわかります。

逆に、これらを満たしていれば、グラフは条件を満たします。

以上より、与えられたグラフが条件を満たすかどうかは $O(E)$ で判定する事が出来ます。

辺 (u, v) の長さを弄ることを考えます。もしこの辺が dfs 木に含まれるとややこしいことになりますが、制約より v から全ての頂点に到達できるため、辺 (u, v) を取り除いてもこのグラフは連結です。

よって、辺 (u, v) と、これに対応する逆辺を使わないように dfs 木を作り、同様の判定を行えば良いです。

よって、弄る辺を決めたら $O(E)$ で判定できるため、この問題は $O(E^2)$ で解けました。

ところで、長さが 0 でないサイクルを見つけたとします。するとこのサイクルの辺のうちどれかは必ず長さを変えないといけません。サイクルの長さは高々$O(V)$ なので、もしこのようなサイクルを見つけられれば、$O(VE)$ で解けます。

まず与えられたグラフが条件を満たすか判定し、もし辺 (u, v) でエラーが起きたら、この辺と、dfs 木での (u, v) をつなぐパスは必ず長さ非 0 のサイクルになります。

よって、これでサイクルを見つけることができます。

実際の実装ではパスの復元はすこし面倒なため、エラーが起きた辺と、dfs 木の辺すべてを調べるようにすると実装が楽です。

余談

競プロで「無向グラフの全てのサイクル」と出てきた時は、とにかく dfs 木を作るのが王道パターンです。なぜなら、全域木を取り、(全域木に含まれない辺 1 本)+全域木上のパス、を列挙すると、これらのうちいくつかの xor を取ることで、グラフのサイクルが列挙できるからです (興味のある人は cycle basis, サイクル基底などで検索してみてください)。そして全域木のなかでも、無向グラフならば dfs 木の性質は非常に良いため、よく使われます。

A*や最小費用流の Primal-Dual 法で使うポテンシャルという概念を使えば、結局のところこの問題の条件は、「頂点に適切なポテンシャルを割り振り、辺の長さを全て 0 にすることが出来るか」と言い替えることが出来ます (先述の h_i がポテンシャルになります)。

D: なめらかな木

まず $O(2^N N^2)$ 状態の bit DP を考えます。

木の頂点に $1, 2, ..., N$ と順番に書いていくこと考えます。

ここで i を書くときに保持しておかなくてはいけない状態はなんでしょうか。

まず、木の頂点それぞれについて、その頂点に値を書いたか書いてないかは持たないといけなそうです。$(O(2^N))$

そして、2 個前まで、つまり $i-2, i-1$ を書いた位置も持っておかないといけなそうです。$(O(N^2))$

3 個以上前の位置は持つ必要がありません。(これから i 以上の数を書き込んでいくため、$i-3$ 以下は全て離れすぎていており、同一視して良い)

遷移は、まだ値が書かれていない頂点を 1 個選んで、その周りに 3 個以上前、つまり $i-3$ 以下が書かれていなければそこに書き込み遷移します。この遷移からも $i-3$ 以下を同一視して良いことがわかります。

これで $O(2^N N^2)$ 状態、$O(N)$ 遷移の bit DP ができました。

ここで、枝刈りを考えます。

まず、頂点の次数はすべて 4 以下と仮定して良いです (そうでないなら答えは 0 です。)

$i-3$ 以下が書き込まれた頂点と、値がまだ書かれていない頂点が隣り合っていた場合、その時点で枝刈りしてよいです。

なぜならば、その値がまだ書かれていない頂点には、どうせ未来永劫値が書き込めないため、結局答えが 0 になるからです。

これは非常に強い枝狩りです。なぜならば、木から $i-2, i-1$ が書かれた頂点を取り除いたあとの森を考えると、連結成分ごとに「すべて値が書かれている」か「すべて値が書かれていない」かのどちらかにならないといけないからです。

また、頂点の次数は 4 以下であるため、連結成分の個数は必ず 7 個以下です。

よって状態数は $N \times (N-1) \times 2^7 = 313600$ で抑えることができます。

上記の bit DP をハッシュマップを使ってメモ化し、この枝刈りをいれれば十分高速にこの問題を解くことができます。

なお、頑張ればハッシュマップを使わず実装することも可能です。

E: 足のばし

まず、木の直径ということで、中心を考えたくなります。

一般に木の中心を考える問題では、中心が頂点の場合と辺の場合の場合分けが発生し大変なのですが、頂点倍加というテクニックを使えばこれを回避できることが多いです。

頂点倍加とは、各辺 (u, v) を、新しい頂点 e を使い、辺 (u, e) と辺 (e, v) に置き換えるテクニックです (増える頂点は $n-1$ 個なため、正確には倍ではないですが…)。

これは、各辺の中心に新しい頂点を足す操作であり、これを行うと、頂点が中心の場合だけ考えれば良くなることが多いです。

この問題では、このテクニックを使うと、「辺の長さを 2 伸ばす」を K 回行う問題になり、直径/2、つまり半径の最小値を求める問題になります。本当は、辺 (u, e) と辺 (e, v) を同時に 1 伸ばすという操作も考えなくてはいけないのですが、これは考えなくていいことが示せます。

これにより、頂点が中心の場合のみ考えればよくなります (「辺の長さを 2 伸ばす」という操作だけでは、辺を中心にすることはできません)。

では、各頂点について、その頂点が中心である場合はどうなるかを考えます。なお、頂点が葉の場合は例外で、そもそも中心になれないため考えません。

選んだ頂点を根とした、根つき木にします。まず、この時点での最も深い葉の深さを R とすると、半径は必ず R 以上になります。そして半径を R にするには、

$$(R \times (\text{葉の個数}) - (\text{最初の時点での葉までの距離の総和}))/2$$

回まで操作できることがわかります。また、半径を $R+2x$ にするには、

$$((R+2x) \times (\text{葉の個数}) - (\text{最初の時点での葉までの距離の総和}))/2 =$$
$$(R \times (\text{葉の個数}) - (\text{最初の時点での葉までの距離の総和}))/2 + x \times (\text{葉の個数})$$

回まで操作できることがわかります。なお、半径を $R+2x+1$ にすることはできません。

$dp[i] =$ (最終的な半径が i となる時の操作回数の最大) として、この配列 dp の値を求めることを考えます。

これは、どの頂点を選ぶかにかかわらず葉の個数は一定であることを使えば、まず各頂点について、最初の時点での半径 R と $X = (R \times (\text{葉の個数}) - (\text{最初の時点での葉までの距離の総和}))/2$ を求め、$dp[R] = \min(dp[R], X)$ とし、最後に $i = 1, 2, ...$ と順に $dp[i+2] = \min(dp[i+2], dp[i] + \text{葉の個数})$ とすれば、正しく値が求まることがわかります。

これらの情報を得るためには、すべての頂点について、最初の時点での葉までの距離の総和や最大を求める必要があります。ですがこれは全方位木 DP で線形で求めることができます。

元の問題に戻ります。

上の $dp[i]$ を使えば、$dp[i] \geq K$ なる最小の i を求める問題になります。ここで、i はどのぐらいまで考える必要があるでしょうか。実は、最終的な半径というのは $O(K)$、つまり 10^{18} オーダーになってしまうため、とてもじゃないですが十分な長さの $dp[i]$ を計算することは出来ません。

ですが、$dp[i]$ は、$n \leq i+2$ ならば、必ず $dp[i+2] = dp[i] + (\text{葉の個数})$ となっているため、$i < n$ についてのみ $dp[i]$ を求めれば、それより後については簡単に求めることができます。よって、$O(N+Q)$ で全てのクエリを処理できます。

以上より、この問題は $O(N+Q)$ で解くことができました。