ABC #040 解説

2016 年 6 月 18 日

A - 赤赤赤赤青

x 番目にある青いブロックを 1 番目に持ってくる場合と n 番目に持ってくる場合の両方を考える。

- 1 番目に持ってくるとき、必要な操作の回数は $x-1$ 回である。
- n 番目に持ってくるとき、必要な操作の回数は $n-x$ 回である。

よって答えは $x-1$ と $n-x$ のうち小さい方となるので、両方計算して小さい方を出力すればよい。

こういった ± 1 が絡む計算式については n や x といった文字のまま考えるだけでなく、実際のケース (たとえば入出力例) を使って検算することでミスを減らすことができる。

B - □□□□□

入出力例 1 のようにタイルを余らせても正方形に近い形にしたほうがよい場合もあれば、入出力例 2 のように多少縦横の長さに差が出てもタイルを多めに使ったほうがよい場合もある。そのため、答えを n の簡単な式で表すことは難しそうである。

そこで、作る長方形の縦の長さ h と横の長さを w をすべて試して最良のものを選ぶことを考える。$h = 1, 2, \ldots$ と順に考えていき、$h \times w$ が n を超えない範囲で w の値をすべて考えればよい。

このとき、それぞれの h に対して w の取りうる値はおよそ $\frac{n}{h}$ 通りである。よって全体で試すべきパターンは

$$\frac{n}{1} + \frac{n}{2} + \cdots + \frac{n}{n} = n\left(1 + \frac{1}{2} + \cdots + \frac{1}{n}\right)$$

通り程度である。

この式の右辺にある自然数の逆数の和は調和数と呼ばれており、その大きさは $\ln n$ (n の自然対数) ほどになることが知られている。(調和数などで検索してみるとよい)

したがって試すパターン数は $n \ln n$ 程度となり、$n \leq 10^5$ ならば十分な速さとなる。

なお、h を決めた際、w はできるだけ大きい方がよいことを利用すれば試すパターン数は n 程度になる。さらに、$h \leq w$ となる場合だけを考えればよいことも利用すれば、試すパターン数は $\sqrt{n}$ 程度になる。(どちらも今回の問題を解くのには必要ない)

C - 柱柱柱柱柱

基本的な動的計画法 (DP; Dynamic Programming) を用いる問題である。N が大きいので移動パターンをすべて試すことは現実的でなく、また入出力例 2 からわかるように常にコストの安い選択をすればよいわけでもない。

$dp[]$ という配列を用意し、$dp[i]$ が「1 本目の柱から i 本目の柱へ移動するまでの合計コストの最小値」となるように計算をしたい。まず明らかに $dp[1] = 0$ である。1 より大きな i については次のように考えられる。

- $i-2$ 本目の柱から一気に i 本目の柱へ飛んだ場合、そのときの最小コストは $dp[i-2] + |A_{i-2} - A_i|$ である。
- そうではなく、$i-1$ 本目の柱から i 本目の柱へ来た場合、そのときの最小コストは $dp[i-1]+|A_{i-1}-A_i|$ である。

よって、i を小さい方から $dp[i]$ の値を求めていき、それぞれの計算は上の 2 パターンのうちより小さい方の値をとればよい。最終的に $dp[N]$ が求める答えとなる。

D - 道路の老朽化対策について

いくつかの頂点と、その間を結ぶいくつかの辺からなる構造をグラフという。この問題では N 個の都市をぞれぞれ頂点、M 本の道をそれぞれ辺としてグラフの問題と見ることができる。さらに、この問題では道路は双方向に通行可能 (一方通行ではない) で、辺にそれぞれ値がついているので、重みつきの無向グラフについての問題である。

また、無向グラフにおいて「ある頂点から辺を辿って行き来できるような頂点の集合」を連結成分と言う。したがって、この問題はグラフの言葉を使えば以下のように言い換えられる。

「N 個の頂点と M 本の辺からなる重みつき無向グラフが与えられる。Q 個の質問に答えよ。それぞれの質問は『重みが w より大きい辺だけを考えたとき、頂点 v が属する連結成分の大きさは？』である」

このように問題をグラフなどの言葉を使って言い換えることは、問題をグラフに関するさまざまなアルゴリズムや過去に解いたことのある問題などと結びつけるのに役立つ。

50 点解法

無向グラフにおいてある頂点 v の属する連結成分を求めるには、深さ優先探索や幅優先探索などのグラフ探索アルゴリズムを用いればよい (これら基本的なグラフ探索アルゴリズムについては検索すれば多くの解説がある)。

こういった探索では最悪の場合すべての頂点と辺を見ることになるので、ひとつの質問に答えるために $\mathcal{O}(N+M)$ 程度の時間が必要である。

よって全体で $\mathcal{O}(Q(N+M))$ 時間がかかることになり、50 点が得られる。

満点解法

■Union-Find　グラフの連結成分などを管理するひとつの方法として、素集合データ構造 (Disjoint-set Data Structure) がある。日本のプログラミングコンテスト周辺では、このデータ構造のことをサポートする操作の名前を使って **Union-Find** と呼ぶことが多い。Union-Find の詳しい実装などについては過去に解説が存在する (http://www.slideshare.net/chokudai/union-find-49066733) ので参考にするとよい。

グラフの連結成分管理の観点から見ると、Union-Find に可能なのは以下の操作である。

- Union: ふたつの連結成分をつなげる。
- Find: ある頂点の属する連結成分の代表となる点を求める。

Find で代表となる点を求めることで、たとえば 2 点が同じ連結成分に属するかどうかを調べることができる。しかし今回の問題では連結成分の大きさを求める必要があるため、機能を追加する必要がある。

はじめにすべての点が独立した連結成分になっている状態では、どの頂点もその連結成分の大きさは 1 である。その後、Union 操作によって異なる 2 つの連結成分がつなげられるときに、その大きさを足しあわせて持つことにすればよい。

具体的には、$S[Find(v)]$ を「頂点 v の属する連結成分の大きさ」とし、最初はすべて 1 で初期化しておく。その後、頂点 u, v の属する連結成分どうしをつなげるとき、$Find(u) \neq Find(v)$ ならば、$S[Find(u)] + S[Find(v)]$ を新たな $Find(u), Find(v)$ の値に対応する S の値とすればよい。

■アルゴリズム　Union-Find の重要な制約として、連結成分をつなげることはできるがその逆はできない、というものがある。したがって、Union-Find をただ使うだけでは、各質問ごとに毎回 Union-Find を構築する必要があり高速化ができない。

そこで、辺の重みが大きい順 (道が新しい順) に質問に答えていくことを考える。すると、ある質問に答えたあと次の質問に答えるとき、対象となるグラフは辺が増えることはあっても減ることはない。よってひとつの Union-Find を使って、辺の重みが大きい順に連結成分をつなげつつ質問に答えていくことができる。

こうすれば Union-Find の操作回数が全体で $\mathcal{O}(M + Q)$ 回程度になり、Union-Find の一回の操作は非常に高速なので満点が得られる。

具体的な実装では辺と質問をまとめて新しい順にソートし、順番に見ていくようにするとよい。ただし、辺と質問で年が同じ場合は、先に質問に答えたあとに辺を見るようにする必要があるので注意すること。