KEYENCE Programming Contest 2019 / キーエンス プログラミング コンテスト 2019 解説

IH19980412, satashun

2019/01/13

A: Beginning

各数字が何回登場するかを配列に持っておきます。
$1, 4, 7, 9$ が一度ずつ登場するなら YES、そうでないなら NO を出力すれば良いです。

解答例: https://atcoder.jp/contests/keyence2019/submissions/3974382

B: KEYENCE String

文字列長は高々 100 なので、取り除く「連続した部分文字列」を全通り試すことができます。
各々の場合において、取り除かれない文字を元の文字列の順番で結合して得られた文字列が”keyence” になれば、この文字列がキーエンス型文字列であることがわかります。
逆に、全ての場合でそうならなかったとすると、この文字列はキーエンス型文字列ではありません。

解答例: https://atcoder.jp/contests/keyence2019/submissions/3974383

C: Exam and Wizard

まず、 $A_i < B_i$ なる i については、必ず $A_i \neq C_i$ になるため、この問題の本質は、 $A_i \geq B_i$ なる A_i で、$A_i \neq C_i$ なるものの個数を最小化することです。

この問題は、$A_i < B_i$ なる i について、 $B_i - A_i$ の総和を S と呼ぶことにすると、 $A_i \geq B_i$ なる i について、 $A_i - B_i$ をできるだけ少なく選び、総和を S 以上にする問題と同値です。

よって、最初に S を計算した上で、$A_i \geq B_i$ なる i について、 $A_i - B_i$ を大きい順に見ていき、今までの総和が S 以上になるまで num 個足したとすると、答えは ($A_i < B_i$ なる i の個数) + num になります。

注意すべきケースとして、$A_i < B_i$ なる i がない (答えは 0) や、 $A_i \geq B_i$ なる i についての $A_i - B_i$ の総和が S に満たない (答えは -1) などが挙げられます。

D: Double Landscape

まず、A_i と B_j の中で同じ値がある場合答えは 0 です。以下、A_i と B_j はどちらも全要素が異なるものとして考えます。グリッドにおいて、x が置かれうる場所について、x が大きい順に考察します。

- x が A_i と B_j の両方に現れる場合
  x の場所は一意に定まります。
- x が A_i と B_j の片方に現れる場合
  x が A_i に現れている場合のみ述べます。(逆の場合も対称性よりほぼ同じです)
  x の場所としてあり得るものの個数は、$(B_j \geq x$ なる j の個数) です。
- x が A_i と B_j に現れない場合
  x の場所としてあり得るものの個数は、$(A_i \geq x$ なる i の個数) $\times$ $(B_j \geq x$ なる j の個数) です。
  (ただし、そのうち $N \times M - x$ 個は x よりも大きい値が埋めています)

以上の値を全て掛け合わせれば答えが求まります。計算量は $\mathcal{O}(NM)$ です。

E: Connecting Cities

基本的には最小全域木を求める問題だと思えば良いですが、今のままだと使う辺の候補が $\mathcal{O}(N^2)$ 本で、このままでは絶望的です。

やり方は何通りかあると思いますが、ここでは辺の候補を $\mathcal{O}(N)$ 本に絞り、それらを $\mathcal{O}(N \log N)$ 時間で求める方法を述べます。

N 個の都市に、規模が小さい順に新たに 1 から N の番号をつけることにします。 (規模が等しい場合は適当に順番を決めて良いです)
すると、以下の事実が成り立ちます:

- 番号 i の都市と、その左にある番号 1 以上 $i-1$ 以下の都市を結ぶ道路は、最も建設コストが少ないもののみ考えればよい。
- 番号 i の都市と、その右にある番号 1 以上 $i-1$ 以下の都市を結ぶ道路は、最も建設コストが少ないもののみ考えればよい。

一つ目の証明は、番号 i の都市を v 、v の左にある番号 i 未満の都市と v を結ぶ道路の内、最も建設コストが少ないものが v と u を結ぶものとし、最小全域木において実際に使った道路は v と w を結ぶものであったとすると $(u \neq w)$
u と v と w を通るサイクルにおいて、必ず v と w を結ぶ道路がコスト最大であることが容易に示せるので、v と u を結ぶ道路以外考えなくても最小全域木が得られることがわかります。二つ目の証明も対称性より同様に示せます。

さて、ある区間上の建設コスト最小の都市を見つけるのは segment tree により 1 回あたり $\mathcal{O}(\log N)$ 時間で求めることができ、辺の候補は $\mathcal{O}(N)$ 本になっているので、計算量は $\mathcal{O}(N \log N)$ になります。

辺の候補を絞ったあとは、クラスカル法を用いれば最小全域木を求めることができます。この計算量も $\mathcal{O}(N \log N)$ なので、この問題を全体で $\mathcal{O}(N \log N)$ 時間で解くことができました。

F: Paper Cutting

まず、スコアについて考えてみましょう。それぞれの長方形ごとに何回カウントされるかを足す方法が考えられますが、このままでは計算量が大きすぎます。4 頂点の代わりに、左下の座標が (i, j) であるような破片がカウントされる回数をまとめて数えることを考えます。

明らかに、このような破片は $x = i$ と $y = j$ で切る前は存在しません。そして、その後どのような操作をしてもこのような破片は常に 1 つだけ存在します。対称性より、基本的に i, j の値に依らず同じ回数だけスコアに寄与します。

しかし、$i = 0$ の場合と $j = 0$ の場合は元から切られていると考えることが出来るので、場合分けして数えることにします。

$H + W$ 本のうち K 本を順番に切る方法が何通りあるか考えてみましょう。これは $H + W$ までの整数からなる順列のうち前 K 個の場合の数を考えればよく、後ろ $(H + W - K)$ 個 の順番を無視して $(H + W)!/(H + W - K)!$ 通りとわかります。各破片がカウントされる回数の期待値を求め、最後に操作列の個数をかけることでスコアの和を求めることができます。K 番目までの操作列についての以下の確率を考える上で、 $K + 1$ 番目以降も順列になっていると考えても構いません。

$i \neq 0, j \neq 0$ のとき、$x = i$ の後に $y = j$ が選ばれる場合、スコアは $y = j$ が何回目に選ばれるかに依るので、これを t 回目とします。このような事象の確率は、$1/(H + W) * (t - 1)/(H + W - 1)$ で、スコアに $K + 1 - t$ 寄与します。$y = j$ の後に $x = i$ が選ばれる場合も同じです。このような (i, j) の組は HW 通りあります。

$i = 0, j \neq 0$ または $i \neq 0, j = 0$ のとき、t 回目に 0 でない方で切るとして、この確率は $1/(H + W)$ であって、スコアに $K + 1 - t$ 寄与します。このような (i, j) の組は $H + W$ 通りです。

$i = 0, y = 0$ のとき、初めから存在するので常に K 寄与します。

以上で $O(K)$ でこの問題が解けました。

KEYENCE Programming Contest 2019 Editorial

IH19980412, satashun

2019/01/13

A: Beginning

Check if all of $1, 4, 7, 9$ appear in the input.
Example: https://atcoder.jp/contests/keyence2019/submissions/3974382

B: KEYENCE String

Try all substrings.
Example: https://atcoder.jp/contests/keyence2019/submissions/3974383

C: Exam and Wizard

Please wait for a while!

D: Double Landscape

Please wait for a while!

E: Connecting Cities

We want to compute the MST of the graph, but a straightforward algorithm doesn't work because there are $\mathcal{O}(N^2)$ edges. We first enumerate candidates of MST edges, and then compute the MST of the graph with only those edges. There are two ways to find candidates:

Divide and conquer.

Let's divide the array into two halves. Only consider edges between the two halves. When are we interested in the edge between i and j such that $0 \leq i < N/2$ and $N/2 \leq j < N$?

The cost of the edges can be written as $f(i)+g(j)$, where $f(i) = A_i - Di$ and $g(j) = A_j + Dj$. Let i_0, j_0 be the indices that minimize the values of $f(i_0), g(j_0)$. We claim that the edge between i and j can be a candidate only when $i = i_0$ or $j = j_0$. Otherwise, the three edges $(i, j_0), (i_0, j), (i_0, j_0)$ are cheaper than the edge (i, j), so this edge can't be included in the MST. Thus, we limit the number of edges between the two halves to $O(N)$.

If we apply divide-and-conquer with the observation above, the total number of candidate edges will be $O(NlogN)$, and this solution works in $O(Nlog^2N)$.

Sort by A_i.

For simplicity, assume that the values of A_i are pairwise distinct.

Consider a particular city (call it x). We can prove the following sbout the edges that connect this city and smaller city (cities that satisfy $A_i < A_x$):

- Among edges that connects x and all smaller cities to the left of x, we should only consider the cheapest edge.
- Among edges that connects x and all smaller cities to the right of x, we should only consider the cheapest edge.

Let's prove the first claim. Suppose that among edges that connects x and all smaller cities to the left of x, the chapest one is (x, y). Then, for each other z that satisfies $z < x$ and $A_z < A_x$, both edges (x, y) and (y, z) are cheaper than (x, z). Thus, (x, z) never becomes the MST edge. The second claim is similar.

This way, the candidates will be $O(N)$, and this solution works in $O(NlogN)$.

F: Paper Cutting

We'll compute the expected score when we choose K cuts uniformly at random. (The answer is this expected value times $N(N-1)...(N-K+1)$, where $N = H + W$).

First, let's find a simpler way to compute the score. For each point (i, j), we count the number of times a rectangle is counted whose lower-left corner is at (i, j). Call this number $f(i, j)$. The score is the sum of $f(i, j)$ for all $(i, j)(0 \leq i \leq W, 0 \leq j \leq H)$.

How to compute the expected value of $f(i, j)$? For simplicity, assume that $i \neq 0, j \neq 0$.

In order for $f(i, j)$ to be nonzero, the cuts at $x = i$ and $y = j$ must be made (otherwise (i, j) can't be a lower-left corner). Once these two cuts are made, each time we make a cut, the value of $f(i, j)$ increases by one.

Let $N = H + W$. The probability that both of the two cuts are made is:

$$\frac{\binom{K}{2}}{\binom{N}{2}} \tag{1}$$

and for each line except for the two lines made above (call this line L), the probability that both of the two cuts and the cut at L are made, and L is the last cut of these three cut is (this means that the value $f(i, j)$ is increased by one by line L):

$$\frac{\binom{K}{3}}{\binom{N}{3}} \times \frac{1}{3} \tag{2}$$

Thus the expected value of $f(i, j)$ is:

$$\frac{\binom{K}{2}}{\binom{N}{2}} + \frac{\binom{K}{3}}{\binom{N}{3}} \times \frac{1}{3} \times (N-2) \tag{3}$$

Similarly, we can compute the values of $f(i, j)$ for cases $i = 0$ and $j = 0$. (These cases should be handled separately because the lines $x = i$ or $y = j$ may exist from the beginning in these cases.)

The bottleneck is the computation of $N(N-1)...(N-K+1)$, and this solution works in $O(K)$ time.