ABC #107 / ARC #101 Editorial

writer: sugim48

2018 年 8 月 25 日

A: Train

答えは $N+1-i$ です.

- C++ のコード例: https://beta.atcoder.jp/contests/abc107/submissions/3070852
- Java のコード例: https://beta.atcoder.jp/contests/abc107/submissions/3070854
- Python 3 のコード例: https://beta.atcoder.jp/contests/abc107/submissions/3070856

B: Grid Compression

まず，黒いマスが含まれる行および列をマークします．その後，マークされた行とマークされた列が交差する位置の $a_{i,j}$ のみを出力すればよいです.

- C++ のコード例: https://beta.atcoder.jp/contests/abc107/submissions/3070967
- Java のコード例: https://beta.atcoder.jp/contests/abc107/submissions/3070968
- Python 3 のコード例: https://beta.atcoder.jp/contests/abc107/submissions/3070969

C: Candles

連続する K 本のろうそくに火を付けるのが最適です．N 本のうち連続する K 本のろうそくを選ぶ方法は $N-K+1$ 通りしかないので，全探索することにします．連続する K 本のろうそくのうち，最も左のものを l とし，最も右のものを r とします．l と r を訪れれば，自然と残りのろうそくを訪れることになります．よって，「座標 0 を出発し，座標 x_l と座標 x_r をともに訪れるための最短時間」が求めるべき値です．これは次の 2 通りの値の最小値です:

- $l \to r$ の順に訪れる場合: $|x_l| + |x_r - x_l|$
- $r \to l$ の順に訪れる場合: $|x_r| + |x_r - x_l|$

連続する K 本のろうそくを選ぶ方法を全探索し，この値の最小値を求めれば，それが答えです．時間計算量は $\mathrm{O}(N)$ です.

D: Median of Medians

まず，中央値の性質を調べてみましょう．長さ M の整数列 b の中央値を x とします．このとき，x は次の性質を満たします:

- b 中に x 以上の要素が $\left\lceil \frac{M}{2} \right\rceil$ 個以上含まれる．
- x は，上の性質を満たす整数の中で最大である．

よって，中央値を求める際には二分探索が有効であると分かります．すなわち，前者の性質を満たすような最大の整数を二分探索で求めればよいです．

以上の方針をこの問題に適用すると，次の問題を解ければよいことになります:

> $m_{l,r}$ $(1 \leq l \leq r \leq N)$ のうち x 以上の要素は何個か？

上述の中央値の性質を用い，問題をさらに言い換えます:

> a の連続部分列 $a[l,r]$ $(1 \leq l \leq r \leq N)$ のうち，x 以上の要素を $\left\lceil \frac{r-l+1}{2} \right\rceil$ 個以上含むものは何通りか？

この言い換えにより，a の各要素は x 以上か否かという情報のみが必要であることが分かります．そこで，a の各要素を，x 未満ならば `A` へ，x 以上ならば `B` へ置き換えてしまいます．すると，問題は次のように言い換えられます:

> a の連続部分列 $a[l,r]$ $(1 \leq l \leq r \leq N)$ のうち，(`A` の個数) $\leq$ (`B` の個数) を満たすものは何通りか？

見通しをより良くするために，さらに `A` を -1 へ，`B` を $+1$ へ置き換えると，問題は次のように言い換えられます:

> a の連続部分列 $a[l,r]$ $(1 \leq l \leq r \leq N)$ のうち，要素の総和が 0 以上のものは何通りか？

連続部分列の要素の総和は，累積和を用いると簡単に扱えます．(± 1 へ変換後の) a の累積和の列を S とします．すなわち，各 $0 \leq i \leq N$ について $S_i = \sum_{j=1}^{i} a_i$ とします．すると，$a[l,r]$ の要素の総和は $S_r - S_{l-1}$ と計算できます．よって，最終的な問題は次のようになります:

> 組 (l,r) $(0 \leq l < r \leq N)$ のうち，$S_l \leq S_r$ を満たすものは何通りか？

これは，数列 S の転倒数を求める問題とほぼ同じです．非常に典型的なので，「転倒数 アルゴリズム」などで検索すると多くの解説が見つかります．この問題は $\mathrm{O}(N \log N)$ 時間で解くことができます．よって，二分

探索のステップ数を掛け合わせると，元の問題は $\mathrm{O}(N \log N \log A)$ 時間で解くことができます．ここで，A は a の要素の最大値です．

E: Ribbons on Tree

「辺に少なくとも 1 本のリボンが張られている」という条件よりも，「辺にリボンが張られていない」という条件の方が扱いやすいので，包除原理を用いることにします．包除原理を用いると，答えは次のように計算できます:

$$\sum_{F \subseteq E} (-1)^{|F|} (F \text{ のどの辺にもリボンが張られないようなペアの分け方の通り数})$$

ここで，E は与えられた木の辺集合とし，F は E の部分集合全体を動きます．

まず，F をひとつ固定したとき，「F のどの辺にもリボンが張られないようなペアの分け方の通り数」を求めてみましょう．木から F の各辺を取り除き，木を $|F|+1$ 個の連結成分へ分割します．すると，各ペアは同一の連結成分内の頂点からなる必要があります．よって，各連結成分の頂点数をそれぞれ $n_1, n_2, \ldots, n_{|F|+1}$ とすると，通り数は $g(n_1) \cdot g(n_2) \cdot \cdots \cdot g(n_{|F|+1})$ となります．ここで，$g(\cdot)$ は次のように定義されます:

$$g(n) := \begin{cases} (n-1) \cdot (n-3) \cdot \cdots \cdot 3 \cdot 1 & (n : \text{偶数}) \\ 0 & (n : \text{奇数}) \end{cases}$$

当然 F を全探索することはできないので，代わりに根付き木上で DP をします．DP の状態は，「今の部分木」「今の連結成分の頂点数」に加え，「これまでに取り除いた辺の本数の偶奇」が必要です．よって，$dp(\cdot,\cdot,\cdot)$ を次のように定義します:

$$dp(v,x,p) := \left(\begin{array}{l} \text{頂点 } v \text{ を根とした部分木において，} v \text{ を含む連結成分の頂点数が } x \text{ で，} \\ \text{部分木内で取り除いた辺の本数の遇奇が } p \text{ の場合の，ペアの分け方の通り数} \end{array} \right)$$

この dp を葉から根へ向かって順に更新し，最後に根 r における dp を参照することで，答えが求まります．

部分木どうしの dp をマージするとき，常に x を 0 から N まで回してしまうと，各マージごとに $\mathrm{O}(N^2)$ 時間が掛かります．マージは全体で $N-1$ 回行われるので，合計で $\mathrm{O}(N^3)$ 時間が掛かり，TLE してしまいます．ここで，頂点数 X の部分木の dp においては，x は X まで考慮すればよいことを利用します．すると，頂点数 X の部分木と頂点数 Y の部分木の dp をマージするのに掛かる時間は $\mathrm{O}(X \cdot Y)$ に抑えられます．これは定数倍改善でしかないように思えますが，実は計算量を本質的に改善し，全体で $\mathrm{O}(N^2)$ 時間となります．これは十分に高速です．計算量解析は省きますが，「木 DP 二乗」などで検索すると解説が見つかります．

F: Robots and Exits

まず，最も左の出口より左に置かれているロボットたちと，最も右の出口より右に置かれているロボットたちは無視してもよいです．

ある時点におけるロボットの変位の履歴を考えます．この履歴の最小値を x とし，最大値を y とします．最初，$(x,y) = (0,0)$ です．すぬけ君は，(x,y) から $(x+1,y)$ または $(x,y+1)$ のいずれかを自由に選び，変化させることができます．これは，すぬけ君が二次元平面上の原点から，右または上へ距離 1 ずつ移動していくことと等価です．

この言い換えのもとで，各ロボットがいずれの出口を通るかを考えます．ロボット i について，すぐ左の出口までの距離を a_i とし，すぐ右の出口までの距離を b_i とします．すると，x が a_i に達するタイミングと

y が b_i に達するタイミングを比較し，前者が早ければロボット i はすぐ左の出口を通り，後者が早ければロボット i はすぐ右の出口を通ります．すなわち，二次元平面上に点 (a_i, b_i) をプロットしたとき，すぬけ君がこの点の下側を通るか，左側を通るかによって，ロボット i の通る出口が変わります．

ABC #107 / ARC #101 Editorial

writer: sugim48

2018 年 8 月 25 日

A: Train

The answer is $N + 1 - i$.

- C++ Code Example: https://beta.atcoder.jp/contests/abc107/submissions/3070852
- Java Code Example: https://beta.atcoder.jp/contests/abc107/submissions/3070854
- Python 3 Code Example: https://beta.atcoder.jp/contests/abc107/submissions/3070856

B: Grid Compression

Mark all rows/columns that contain at least one black cell, and print intersections of marked rows and marked columns.

- C++ Code Example: https://beta.atcoder.jp/contests/abc107/submissions/3070967
- Java Code Example: https://beta.atcoder.jp/contests/abc107/submissions/3070968
- Python 3 Code Example: https://beta.atcoder.jp/contests/abc107/submissions/3070969

C: Candles

Clearly, we should light K consecutive candles. Let l, r be the leftmost/rightmost indices of candles we light. Since there are only $N - K + 1$ pairs of (l, r), let's try them all.

For a fixed (l, r), we should compute the minimum time required to visit both x_l and x_r (then all other candles between them will be visited too). There are two cases:

- Visit in the order $l \to r$: $|x_l| + |x_r - x_l|$
- Visit in the order $r \to l$: $|x_r| + |x_r - x_l|$

The answer is the minimum of these values. This solution works in $O(N)$ time.

D: Median of Medians

Let b be a sequence of length M. If the median of b is x,

- b contains at least $\lceil \frac{M}{2} \rceil$ elements that are greater than or equal to x.
- x is the largest possible value that satisfies the condition above.

Let's fix a parameter x. We want to solve the following problem:

> Count the number of pairs (l, r) $(1 \leq l \leq r \leq N)$ such that $m_{l,r} \geq x$.

If we can solve this, by doing a binary search on x (using the property of median mentioned above), we can solve the original problem.

For simplicity, we call an element "large" if the element is x or greater, and call it "small" otherwise. Notice that $m_{l,r} \geq x$ holds if and only if (the number of large elements in it) is greater than or equal to (the number of small elements in it). Thus, we replace each large element with 1 and each small element with -1. Now $m_{l,r} \geq x$ holds if and only if $a_l + \ldots + a_r \geq 0$.

To count such pairs, we use prefix sums. We want to count the number of pairs (l, r) such that $S_{l-1} \leq S_r$, where $S_i := \sum_{j=1}^{i} a_i$. Now the problem is reduced to a standard inversion-number problem.

This solution works in $\mathrm{O}(N \log N \log max\_a)$ time in total.

E: Ribbons on Tree

By inclusion-exclusion principle, we want to compute the following value:

$$\sum_{F \subseteq E} (-1)^{|F|} f(F)$$

Here, E is the edge set of the given tree, and $f(F)$ is the number of ways to split vertices in pairs such that no edge in F is decorated with ribbons.

For a fixed F, we can compute $f(F)$ as follows. Suppose that the number of vertices of the $|F|+1$ connected components we get by removing edges in F from the tree are $n_1, n_2, \ldots, n_{|F|+1}$. Then, $f(F) = g(n_1) \cdot g(n_2) \cdot \cdots \cdot g(n_{|F|+1})$, where $g(n)$ is defined as follows (this is the number of ways to split n things into pairs):

$$g(n) := \begin{cases} (n-1) \cdot (n-3) \cdot \cdots \cdot 3 \cdot 1 & (n : even) \\ 0 & (n : odd) \end{cases}$$

To compute the sum of $f(F)$ quickly, let's use DP.

Suppose that the tree is rooted in some way. We decide pairings from leaves, and we keep "current subtree", "the number of vertices in current component", and "the parity of removed edges".

Define $dp(v, x, p)$ as follows:

> Consider the subtree rooted at v. We remove some edges from the subtree and get some connected components. Then, within each component, we split vertices into pairs. Here, the parity of the number of removed edges must be p, and the number of vertices in the component with v (the root of the subtree) must be x. How many ways are there to do this?

Strictly speaking, we need a slight technical modification to this. We split vertices into pairs within each component, but as an exception, we don't make pairs for the vertices in the component with v (in other words, we don't multiply $g(x)$). (Otherwise you will have trouble updating this DP table).

This solution works in $O(N^2)$ time.

F: Robots and Exits

We'll update it later!