

ARC054 解説

DEGwer

2016年5月21日

A問題

問題概要: 長さ L の円形の動く歩道があり、速度 X で回っている。高橋君が速度 Y で歩けると、高橋君が位置 S から位置 D まで行くのにかかる最小の時間を求めよ。

解説:

位置 S から位置 D まで動く歩道の進む方向に測った距離を K とします。これは、 $S \leq D$ のときは $D - S$ に等しく、そうでないときは $D + L - S$ に等しいです。

X と Y の大小で場合分けをします。

$X < Y$ のとき、高橋君は動く歩道を順方向と逆方向のどちらに歩いても、前に進むことができます。順方向に進んだときにかかる時間は $\frac{K}{X+Y}$ 、逆方向に進んだときにかかる時間は $\frac{K}{Y-X}$ です。これらのうち小さいほうが答えです。

そうでないとき、高橋君は順方向に進んだときのみ前に進むことができます。このとき $\frac{K}{X+Y}$ の時間がかかり、これが答えです。

場合分けをしないと、 $\frac{K}{Y-X}$ の計算でゼロ除算が発生したり、負の値が出てきたりするので注意してください。

B問題

問題概要: コンピュータは2年に1.5倍のペースで高速になる。現在 P 年かかる計算を、適切なタイミングで始めて計算が終わるまでの時間を最小化せよ。

解説:

x 年後に計算を始めるとき、計算が終わるまでの時間は $f(x) = x + 2^{-x/1.5}P$ 年です。これは、凸関数と凸関数の和なので、凸関数です。よって、三分探索によって最小値を求めることができます。

また、 $f'(x)$ の零点を求め、そのときの値を出力しても大丈夫です。その場合、零点が0未満のときは $f(0) = P$ を出力する必要があります。

C問題

問題概要: 頂点を同じ集合の間に辺がないように 2 つに分けたとき、両側ともに頂点数が $N \leq 200$ となる二部グラフが与えられる。(以下この分割によってできた頂点集合を A, B と呼ぶ。) 完全マッチングの個数の偶奇を答えよ。

解説:

完全二部マッチングの個数の数え上げは、NP 困難な問題として知られています。しかし、その偶奇だけなら高速に求めることができます。

完全マッチングにおいて、 A の i 番目の要素とマッチングされている要素を B の $\sigma(i)$ 番目の要素とすると、 $\sigma(i)$ たちは 1 から N までの順列となります。つまり、求めたいのは、すべての順列 σ について、 A の i 番目の要素が B の $\sigma(i)$ 番目の要素にマッチングされているものの個数を合計したものの偶奇です。

A の i 番目の頂点と B の j 番目の頂点の間の辺の数を (i, j) 成分とする行列 M (入力で与えられる行列) を考えます。上記の値は、

$$\sum_{\sigma \in S_N} M_{1\sigma(1)} \dots M_{N\sigma(N)}$$

の偶奇に等しいです。ただし、 S_N は 1 から N までの順列すべての集合を表します。

さて、 M の行列式は、

$$\sum_{\sigma \in S_N} \text{sgn}(\sigma) M_{1\sigma(1)} \dots M_{N\sigma(N)}$$

とあらわされます。 $\text{sgn}(\sigma)$ は 1 または -1 の値をとりますが、これらは $\text{mod} 2$ ではどちらも 1 です。すなわち、これは $\text{mod} 2$ では先ほどの式と一致します。行列式を求めるのは $O(N^3)$ でできるので、 $O(N^3)$ でこの問題を解くことができます。

D 問題

問題概要: つぎの 3 種類の操作を繰り返してでできる数列のバブルソートの交換回数を求めよ。

- k のみからなる単項数列を作る
- 数列を k 回繰り返す
- 2 つの数列をつなげる

はじめの二つの操作の回数の和 (以下 N とおく) は 10^5 以下。

解説:

数列 X のバブルソートの交換回数は、 $i < j$ かつ $X_i > X_j$ なる組 (i, j) の個数に等しいので、これを数えることにする。

現在存在する数列 S に対し、以下のデータを持っておきます。

- S のバブルソートの交換回数 X_S
- S にどの数は何回現れたかをすべて列挙したものの D_S
- S を 2 回繰り返したとき、バブルソートの交換回数が $2X_S$ に比べ何回多いかを表す Y_S

これを持っておけば、単項数列を作る操作は $O(1)$ 、数列を繰り返す操作も $O(1)$ 、2 つの数列 S_1 と S_2 をつなげる操作は $O(D_{S_1}$ の要素数 $+ D_{S_2}$ の要素数) ででき、全体で $O(N^2)$ となり、部分点が得られます。

各 S に対し、 D_S を適切なデータ構造で管理すれば、以下のようなアルゴリズムで満点を得ることができます。

- 項 k のみからなる単項数列 S を作る操作において、空の $Trees_S$ を作り、 $X_S = 0, Y_S = 0$ としたうえで、 D_S に値 k を 1 個追加する。
- 数列 S を k 回繰り返す操作において、新しい (X_S, Y_S) をそれぞれ $(kX_S + \frac{k(k-1)}{2}Y_S, k^2Y_S)$ とおき、さらに D_S に格納されている各要素の個数をすべて k 倍する。
- 数列 S と数列 T をつなげる操作 (この操作でできる数列を L とする) において、 D_S と D_T のサイズを比べ、
 - $|D_S| < |D_T|$ ならば、 D_S の要素をすべて見る。値 k が t 個あるとき、 T の値 k 未満の要素の数を数えてそれを t 倍したものたちすべての和と、 X_S と X_T との和が X_L となる。さらに、値 k が t 個あるとき、 T の値 k 未満の要素と k より大きい要素の数を数えてそれを t 倍したものたちすべての和と、 Y_S と Y_T との和が Y_L となる。その後、 D_T に D_S の要素をすべて加え、それを D_L とする。
 - $|D_S| \geq |D_T|$ ならば、 D_T の要素をすべて見る。値 k が t 個あるとき、 S の値 k より大きい要素の数を数えてそれを t 倍したものたちすべての和と、 X_S と X_T との和が X_L となる。さらに、値 k が t 個あるとき、 S の値 k 未満の要素と k より大きい要素の数を数えてそれを t 倍したものたちすべての和と、 Y_S と Y_T との和が Y_L となる。その後、 D_S に D_T の要素をすべて加え、それを D_L とする。
- 最後にできた数列 S の X_S の値が答えである。

このアルゴリズムの正当性は以下の通りです。 X_S が S のバブルソートの交換回数、 Y_S が S を 2 回繰り返した数列において、1 個目の S から 1 個、2 個目の S から 1 個の要素をとってきたときにそれらの大小関係がひっくり返っているものの個数を表すことに注意します。

単項数列を作る操作において、 $X_S = Y_S = 0$ は明らかです。数列 S を k 回繰り返した数列 (ここにおけるひとつの S をブロックと呼ぶことにする) を S^k とおくと、 S^k での $S_i^k > S_j^k$ なる $i < j$ の個数は、

- i, j が同じブロックに属する場合、 kX_S 個
- そうでない場合、 $\frac{k(k-1)}{2}Y_S$ 個

となるので、この操作も正当です。

数列 S, T をつなげる操作において、 S と T をつなげた数列 L での $L_i > L_j$ なる $i < j$ の個数は、

- i, j 番目の要素がともに S の要素であった場合 X_S 個
- i, j 番目の要素がともに T の要素であった場合 X_T 個
- どちらでもない場合、 S の要素 S_i について、 $S_i > T_j$ なる j の個数すべての和、もしくは T の要素 T_j について、 $S_i > T_j$ なる i の個数すべての和 (これらは同じ値を表す)

であるので、上記のアルゴリズムは正当です。

さらに、数列をつなげる操作において、 D_S のサイズが小さいほうの数列を大きいほうの数列にマージしているので、単項数列として作った各数列において、その値が D_S に新しく入れられる回数は高々 $O(\log N)$ 回です。(次の列に入れられるまでに、自分が属する数列 S の D_S のサイズが2倍以上になります。)

よって、あとは「数列 S の k 以下の要素の個数を数える」「 D_S に値を追加する、すなわちある箇所の値をある値だけ増やす」「 D_S のすべての要素の個数を k 倍する」という操作ができればよいことになります。

最初の2つの操作は、各数列について segment tree を動的に構築することで可能になります。最後の操作は、実際に segment tree 上の値を更新せず、各 segment tree についてかわりに「segment tree に書かれている値を何倍すると本当の値になるか」を表す値 P_S を持つておくことにします。すると、

- 数列 S の値 k 以下の要素の個数を求めるときは、segment tree 上での位置 k より前のものの合計に P_S をかけたものを求める
- 位置 k の値を t だけ増やすときは、segment tree 上での位置 k に tP_S^{-1} を加算する
- D_S のすべての要素を k 倍するとき、 P_S を k 倍する

という操作を行うことで、全体で $O(N \log^2 N)$ となり、満点を得ることができます。