KUPC2020 autumn A

`= A. Academic Distance`

writer: etonagisa

2020 年 10 月 10 日

i 番目の教室と $i+1$ 番目の教室の移動距離を足し合わせればよいので、$\sum_{i=1}^{i=n-1} |x_i - x_{i+1}| + |y_i - y_{i+1}|$ が答えです。これは単純な For ループで実装が可能で、計算量は $O(n)$ です.

KUPC2020 autumn B

= B. Bunch of Paper

writer: etonagisa

2020 年 10 月 10 日

紙の番号の小さいほうから順に整数を選んでいくことを考えます。i 番目までの紙の整数の選び方を確定したとすると、$i+1$ 番目以降の紙での整数の選び方は、i 番目の紙で選んだ整数にのみ依存することがわかります。そこで次のような動的計画法を考えます。

$dp[i][j] = i$ 番目の紙までの整数の選び方を確定し、i 番目の紙では $v_{i,j}$ を選んで単調増加な数列を作る方法の数

定義から $dp[1][j] = 1 (1 \leq j \leq K)$ です。また $i > 1$ に対して、$dp[i][j] = \sum_{v_{i-1,l} \leq v_{i,j}} dp[i-1][l]$ が成り立ちます。この式を素直に実装すると計算量は $O(NK^2)$ となり、実行時間制限に間に合いません。

$dp[i-1]$ の値が全てわかっている状態から、$dp[i]$ の値を $O(K)$ で全て計算することを考えます。これができれば全体の計算量は $O(NK)$ となり間に合います。各紙に書かれている整数がソートされていることから、各 $j (1 \leq j \leq K)$ に対してある整数 l_j が存在し、$dp[i][j] = \sum_{l \leq l_j} dp[i-1][l]$ が成り立ちます。直観的には、l_j は $v_{i-1,l} \leq v_{i,j}$ が成り立つ最大の l です。さらにこの l_j は j について広義単調増加します。したがって j と l_j についてしゃくとり法を行うことで、$dp[i]$ の値が $O(K)$ で計算できました。

別解として、あらかじめ $dp[i-1]$ の累積和をとっておき、l_j を二分探索を使って毎回求めることでも正しい解が得られます。この場合の計算量は $O(NK \log K)$ ですが、十分間に合います。

KUPC2020 - C
Grid and Substrings 解説

= C. Character Grid

原案:zaki
解答:yamunaku, nikutto

Kyoto University

October 10, 2020

Contents

問題概要

- すいばかくんに正方形のグリッドをプレゼントしたい
- 各グリッドには英小文字を書き込む
- グリッドから得られる文字列が相異なるようにしたい
- $N = 13$ で満点

考察

- 長さ 3 以上の文字列は考えなくて良い
- 長さ 2 の英小文字から成る文字列は 26^2 個存在する
- 一辺の長さ N のグリッドからは $N(N-1)$ 個の長さ 2 の文字列が得られる
- $N = 18$ が最大
- 一つ手元で見つけたら埋め込んでしまえば OK

解答例 1

- 手で頑張る

a	b	b	c	c	d	d	e	e	f	f	g	g
c	a	d	b	e	c	f	d	g	e	h	f	i
z	e	a	f	b	g	c	h	d	i	e	j	f
d	z	g	a	h	b	i	c	j	d	k	e	l
y	f	z	i	a	j	b	k	c	l	d	m	e
c	y	h	z	k	a	l	b	m	c	n	d	o
x	e	y	j	z	m	a	n	b	o	c	p	d
d	x	g	y	l	z	o	a	p	b	q	c	r
w	f	x	i	y	n	z	q	a	r	b	s	c
c	w	h	x	k	y	p	z	s	a	y	b	u
v	e	w	j	x	m	y	r	z	u	a	v	b
d	v	g	w	l	x	o	y	t	z	w	a	x
u	f	v	i	w	n	x	q	y	x	z	y	a

解答例 2 (nikutto)

- 以下の条件を満たす整数列 a, b を見つける
  - $a_1, a_2, \ldots, a_{N-1}, b_1, b_2, \ldots, b_{N-1}$ は 1 以上 26 未満の distinct な整数
  - $(a_i + a_{i+1} + \ldots + a_j), 1 \leq i < j \leq N-1$ が 26 の倍数でない
  - $(b_i + b_{i+1} + \ldots + b_j), 1 \leq i < j \leq N-1$ が 26 の倍数でない
- 乱択で十分高速に見つかる
- ans[i][j] = 'a' + $(\Sigma_{p=0}^{i-1} a_p + \Sigma_{q=0}^{j-1} b_q)\%26$ で定める

解答例 3

- 一点更新山登り法で見つける
- $N = 18$ まで見つかる

(部分点) 解答例 4

- 枝刈り DFS で見つける
- 探索順・枝刈り規則を工夫しないと厳しいかも
- writer は $N = 12$ まで解けたので 180 点を獲得することができた

Problem D : Stick Conbination = D. Distribute the Bars

本解説で説明する方法は解法の一つにすぎず，他の解法も存在する．

$1, 3, 5, \cdots, 2N-1$ の棒の長さの和は N^2 であるため，M 本になるように棒を接着して棒の長さを揃えようとする場合，各棒の長さが N^2/M となるようにしなければならない．

N が素数の場合　手で小さいケースを構築しようとすると，N が素数であるとき構築できないことが推測できる．M は N^2 の約数であるから，N が素数であるとき，$M = 1, N, N^2$ となるが，問題の制約からどのケースも明らかに解が構築できない．

N が偶数の場合　残っている中で最も短い棒と長い棒の二つを順に接着していくことで，(各 $k(1 \leq k \leq N/2)$ について，長さ k と $2N-k$ の二つの棒を接着することで）$M/2$ 本の長さ $2N$ の棒を作ることができ，解となる．

$N = p \times p$ の場合 ($3 \leq p$)　例えば，$N = 25 = 5 \times 5$ であるとき，$1, 3, \cdots, 49$ の棒を以下のように A, B, C, D, E の p グループに分けて接着する．

$$\begin{pmatrix} 1 & 3 & 5 & 7 & 9 \\ 11 & 13 & 15 & 17 & 19 \\ 21 & 23 & 25 & 27 & 29 \\ 31 & 33 & 35 & 37 & 39 \\ 41 & 43 & 45 & 47 & 49 \end{pmatrix} = \begin{pmatrix} A & B & C & D & E \\ B & C & D & E & A \\ C & D & E & A & B \\ D & E & A & B & C \\ E & A & B & C & D \end{pmatrix}$$

このように $1, 3, \cdots, 2p^2-1$ を $p \times p$ の形に並べ，各行各列で各グループが丁度一つ存在するようにグループ分けをすると各グループの棒の長さの和が $1+19+27+35+43 = 3+11+29+37+45 = 5+13+21+39+47 = 7+15+23+31+49 = 9+17+25+33+41$ と等しくなる．

$N = p \times q$ の場合 ($3 \leq p \leq q$, p, q は奇数)　$N = p \times p$ の場合と N が偶数の場合の手順を組み合わせて構築する．$1, 3, \cdots, 2p^2-1$ までを $N = p \times p$ の場合の手法で p グループに棒を分ける．その後，グループに振り分けていない棒の中で最も短い棒と長い棒の二つを順に組にしていくことで，$p \times (q-p)/2$ 個の総長が等しい棒の組を作ることができ，これらの組を p グループに振り分けることができる．

例えば，$N = 21 = 3 \times 7$ であるとき，$1, 3, \cdots, 41$ の棒を以下のように A, B, C の p グループに分けて接着できる．

$$\begin{pmatrix} 1 & 3 & 5 \\ 7 & 9 & 11 \\ 13 & 15 & 17 \end{pmatrix} = \begin{pmatrix} A & B & C \\ B & C & A \\ C & A & B \end{pmatrix}$$

$$\begin{pmatrix} 19 & 21 & 23 & 25 & 27 & 29 \\ 41 & 39 & 37 & 35 & 33 & 31 \end{pmatrix} = \begin{pmatrix} A & B & C & A & B & C \\ A & B & C & A & B & C \end{pmatrix}$$

この例では (19,41),(21,39),(23,37),(25,35),(27,33),(29,31) の組をそれぞれ A,B,C,A,B,C と振り分けている.

E: Sequence Partitioning `= E. Efficient Partitioning`

まず、数値 x を固定したときに、f の最大値を x 以上にできるかどうかを考えます。

$dp[i]$ を、先頭の部分列 $[1, i]$ について同様に f を定義したとき (これを f_i とする) の f_i の最大値を x 以上にできるかどうかと定義し、$dp[0] = true$ と初期化します。

i を小さい順に 1 から N まで見ていく動的計画法を考えます。

いま、位置 i を見ているとします。$dp[i]$ が $true$ になるには、ある j $(j < i)$ が存在して、$dp[j] = true$ かつ $b_{j+1} + c_i + \sum_{j<k\leq i}(a_k) \geq x$ を満たす必要があります。

これは愚直に k を全探索すると $O(N)$ かかってしまいます。そこで p_i を $0 \leq j < i$ かつ $dp[j] = true$ を満たす j についての $b_{j+1} + \sum_{j<k\leq i}(a_k)$ の最大値と定義すると、$dp[i]$ が $true$ となることは $p_i + c_i \geq x$ が成立することと言い換えられます。また、p_{i+1} は p_i から更新できます。

以上より x を固定すると $O(N)$ で f の最大値が x 以上にできるかどうかを判定できることが分かったので、 x の値を二分探索することで答えとなる値を求めることができます。

KUPC2020 F: GRIDMST

= F. Find the MST for Grid

原案, 作成: yamunaku

1 考察

重みの等しい辺については、順序をつけて、優先的に選ぶものを決めます。順序づけをした後の最小全域木は、順序づけをする前の最小全域木の 1 つであることが示せます。この問題では、等しい重みの辺があるときは、次のように選ぶことにします。

- $(i-1,j)$ と (i,j) を結ぶ辺のスコアを $i \times 10^{18} + j \times 10^9 + 1$ とする。
- $(i,j-1)$ と (i,j) を結ぶ辺のスコアを $i \times 10^{18} + j \times 10^9$ とする。
- 等しい重みの辺については、スコアの小さなものを優先的に選ぶことにする。

最小全域木のアルゴリズムに従って、2 通りの考察を行うことができます。どちらの考察でも、次の結論を導くことができます。

最小全域木に含まれる辺の重みの和は、

$$\sum_{i=1}^{H-1}(A_i + B_1) + \sum_{j=1}^{W-1}(C_1 + D_j) + \sum_{i=2}^{H-1}\sum_{j=2}^{W-1}\min(A_{i-1} + B_j,\ C_i + D_{j-1})$$

である。

1.1 プリム法を用いた考察

頂点 $(1,1)$ からプリム法を適用することを考えます。数列 A, B, C, D の単調性から、プリム法の各段階では、頂点 (i,j) が $(1,1)$ と連結であるとき、

- $i > 1$ ならば、 $(i-1,j)$ も $(1,1)$ と連結である。
- $j > 1$ ならば、 $(i,j-1)$ も $(1,1)$ と連結である。

となることが帰納的に示せます。つまり、

- 頂点 $(i,1)$ $(i > 1)$ が連結になるとき、$(i-1,j)$ と (i,j) を結ぶ辺が選ばれます。
- 頂点 $(1,j)$ $(i > 1)$ が連結になるとき、$(i,j-1)$ と (i,j) を結ぶ辺が選ばれます。
- 頂点 (i,j) $(i > 1, j > 1)$ が連結になるとき、$(i-1,j)$ と (i,j) を結ぶ辺か、$(i,j-1)$ と (i,j) を結ぶ辺のどちらか小さい方が選ばれます。

これらの選ばれた辺だけで全域木ができるので、求める辺の重みは、

$$\sum_{i=1}^{H-1}(A_i+B_1)+\sum_{j=1}^{W-1}(C_1+D_j)+\sum_{i=2}^{H-1}\sum_{j=2}^{W-1}\min(A_{i-1}+B_j,\ C_i+D_{j-1})$$

となります。

1.2 ブルーフカ法を用いた考察

ブルーフカ法は、現在の各連結成分について、それに接続する辺の中で重みが最小のものを、最小全域木の辺として同時に選んでいく方法です。

このブルーフカ法を、頂点 $(i,j)(i>1,j>1)$ すべてについて適用することを考えます。このとき、数列 A,B,C,D の単調性から、頂点 (i,j) について選ばれる辺は、$(i-1,j)$ と (i,j) を結ぶ辺か、$(i,j-1)$ と (i,j) を結ぶ辺のどちらか小さい方になります。

辺を選ぶと、頂点 $(i,j)(i>1,j>1)$ は、 $(x,1)$ $(x>1)$ または $(1,y)$ $(y>1)$ と連結になります。グラフは、頂点 $(1,1)$ 単体と、頂点 $(x,1)$ $(x>1)$ を含む連結成分、頂点 $(1,y)$ $(y>1)$ を含む連結成分になります。

これらの連結成分をすべて連結にすることを考えます。

$(1,y)$ $(y>1)$ を含む連結成分に注目すると、それに含まれる頂点の各行での列番号の最小値は広義単調増加です。連結成分の作り方から、その連結成分に接続している、 $(i-1,j)$ と (i,j) を結ぶ辺があったとすると、$y\leq j$ であり、この辺の重みは、 $(i,j-1)$ と (i,j) を結ぶ辺以下です。A,B の単調性から、$(1,y)$ を含む連結成分から出ている辺の中では、 $(1,y-1)$ と $(1,y)$ を結ぶ辺の重みが最小であることがわかります。

$(x,1)$ $(x>1)$ についても同様なことが言えます。

したがって、$(i-1,1)$ と $(i,1)$ を結ぶ辺、$(1,j-1)$ と $(1,j)$ を結ぶ辺をすべて選ぶのが最適です。

選ばれる辺の重みの和は、

$$\sum_{i=1}^{H-1}(A_i+B_1)+\sum_{j=1}^{W-1}(C_1+D_j)+\sum_{i=2}^{H-1}\sum_{j=2}^{W-1}\min(A_{i-1}+B_j,\ C_i+D_{j-1})$$

となります。

2 答えの求め方

$\sum_{i=1}^{H-1}(A_i+B_1)+\sum_{j=1}^{W-1}(C_1+D_j)$ は $O(H+W)$ で求めることができます。

$\sum_{i=2}^{H-1}\sum_{j=2}^{W-1}\min(A_{i-1}+B_j,\ C_i+D_{j-1})$ は、愚直に求めると $O((H+W)^2)$ かかってしまうので、高速化が必要です。

- $A_{i-1}-C_i\leq -B_j+D_{j-1}$ のとき、$\min(A_{i-1}+B_j,\ C_i+D_{j-1})=A_{i-1}+B_j$
- $A_{i-1}-C_i\geq -B_j+D_{j-1}$ のとき、$\min(A_{i-1}+B_j,\ C_i+D_{j-1})=C_i+D_{j-1}$

であることから、各 i について $A_{i-1}-C_i$ の小さい順に、 $A_{i-1}+B_j$ の累積和と、C_i+D_{j-1} の累積和をとります。

こうすると、各 j について、$A_{i-1}-C_i\leq -B_j+D_{j-1}$ となるような i についての $A_{i-1}+B_j$ の和と、$A_{i-1}-C_i\geq -B_j+D_{j-1}$ となるような i についての C_i+D_{j-1} の和を $O(\log H)$ で求めることができます。

全体の計算量は $O((H+W)\log H)$ となります。

KUPC2020 autumn G

= G. Generate the Sequences

(writer: gazelle)

2020 年 10 月 10 日

解説

1 番目の種類の操作については，A の状態によらず 2 通りが考えられます。

2 番目の種類の操作については，$\sum_{1 \leq i < |A|}(|a_i - a_{i+1}| - 1)$ 通りが考えられます。この値は，1 番目の種類の操作で現在の末尾と異なる数を追加するたびに $M-2$ 増加し，2 番目の種類の操作を行うたびに 1 減少します。

この性質を考えると，「全操作回数」，「1 番目の種類の操作で末尾と異なる数を追加した回数」，「2 番目の種類の操作を行った回数」 が一致すれば，状態をまとめ上げることができます。しかしこの方法で愚直に動的計画法を行ったときの計算量は $O(N^3)$ です。

ここで，1 番目の種類の操作で現在の末尾と同じ数を追加するのは，最後にまとめて行うと考えても問題ないことに注目します。というのもこのタイプの操作は，操作回数を増やす以外に状態や場合の数を変更しないからです。したがって，それ以外の 2 つのタイプの操作だけをまず行うことにすれば，一方を行った回数は，もう一方を行った回数を全操作回数から引けば分かるので，動的計画法で持つべき情報を 1 つ減らすことができます。よって $O(N^2)$ で数え上げることができます。

H: Beans on the Grid `= H. How to Move the Beans`

各マスについて、そのマスに豆があるときの grundy 数を求めたいです。
行ごとに考えます。

[見ている行に皿のないマスが存在する、すなわち # が存在する場合]
$G[i][j]$ をマス (i, j) に豆があるときの grundy 数とし、$L[i][j]$ を「豆がマス (i, j) にあり、かつ豆が左隣のマスを通ったことがあるときの grundy 数」とし、$R[i][j]$ を「豆がマス (i, j) にあり、かつ豆が右隣のマスを通ったことがあるときの grundy 数」と定義します。
このとき、$G[i][j]$ は $G[i+1][j], L[i][j+1], R[i][j-1]$ から計算でき、$L[i][j]$ は $L[i][j+1], G[i+1][j]$ から計算でき、$R[i][j]$ は $R[i][j-1], G[i+1][j]$ から計算できます。状態数も $3 * H * W$ 個なので十分高速に計算できます。

[見ている行に皿のないマスが存在しない、すなわち # が存在しない場合]
この場合は上記の場合と同様に状態を持とうとすると、1 週して同じマスに戻ってきてしまうかもしれません。そこで、$l[i][j][k]$ を「豆がマス i, j にあり、かつ左側を k マス通ったときの grundy 数」と定義し、$r[i][j][k]$ を「豆がマス i, j にあり、かつ右側を k マス通ったときの grundy 数」と定義します。k は 1 以上 W 以下である必要があります。
このとき、 $G[i][j]$ は $G[i+1][j], l[i][j+1][1], r[i][j-1][1]$ から計算でき、$l[i][j][k]$ は $l[i][j+1][k+1], G[i+1][j]$ から計算でき、 $r[i][j][k]$ は $r[i][j-1][k+1]$、$G[i+1][j]$ から計算できます。

ただし、このままではまだ状態数が $O(H*W*W)$ 個と非常に多く、制限時間に間に合いません。そこで、$l[i][j][k]$ および $r[i][j][k]$ の計算が高速化できないか考えます。どちらも同様の方法で考えられるので、特に $l[i][j][k]$ について注目します。上述した通り、$l[i][j][k]$ は $G[i+1][j], l[i][j+1][k+1]$ の mex によって求められます。これは言い換えると、$l[i][j+1][k+1]$ の値を $G[i+1][j]$ の値によって変換したことになります。

例えば、$G[i+1][j] = 0$ であったとき、$l[i][j+1][k+1]$ の値が $0, 1, 2, 3$ のそれぞれの場合について、$l[i][j][k]$ の値はそれぞれ $1, 2, 1, 1$ となります。
さて、このように $l[i][j][k]$ の計算の仕方を数字への操作に落とし込んだところで、操作同士の積を考えます。例えば、上の例で $G[i+1][j] = 0$ かつ $G[i+1][j+1] = 0$ のときは、数字に上記の操作を 2 回することになり、結果の列はそれぞれ $2, 1, 2, 2$ となります。
こうなると、例えば操作を $2 \times x$ 回する場合の数字の変換は、古い x 回の操作による数字の変換と新しい x 回の操作による数字の変換を合成することで求めることができます。

このようにダブリングの要領を用いることで $O(logW)$ で各 i, j についての $l[i][j][1]$ の値を求めることができます。(実際にはセグ木に操作を載せるのが一番楽かもしれません。)

KUPC2020 I: Coloring Paths

= I. Interesting Coloring

原案, 作成: yamunaku

1 方針

グラフが連結で、どの辺についてもその辺の迂回パスが存在することと、グラフ全体が 2 重連結であることは同値です。

頂点を 1 つ選んで根として、DFS 木を作ります。このとき、木に含まれない辺（後退辺と呼びます）が結ぶ 2 頂点は、先祖と子孫の関係にあります。

DFS 木の各辺を、次の条件 (☆) が満たされるように塗りたいです。

> (☆) 木のどの葉についても、葉と根を結ぶパスは 7 種類以下の色で塗られている。

この条件を満たす塗り方ができれば、次のようにして、 8 種類以下の色で塗られた迂回パスを各辺 i について構築することができます。

- 辺 i が後退辺であるとき、辺 i が結ぶ 2 頂点を端点とするような DFS 木上のパスが、求めるものである。
- 辺 i が後退辺でないとき、辺 i が結ぶ 2 頂点を u, v とする。ただし u は v の親であるとする。u の先祖であるような頂点 s と、v の子孫であるような頂点 t で、s と t が後退辺で結ばれているようなものが存在する（グラフ全体が 2 重連結であるから)。u と s を端点とする DFS 木上のパスと、s と t を結ぶ後退辺、t と v を端点とする DFS 木上のパスを順につなげたものが、求めるものである。

2 色の塗り方

次のようにして DFS 木の各辺を塗ります。

- 根から DFS をする。
- DFS で現在訪れている頂点を v とする。v と v の子を結ぶ辺に色を塗りたい。
- まず、根と v を端点とするパスに使われている色で、 v と v の親を結ぶ辺に使われていないものを列挙する。これらを色 $C_1, C_2, ..., C_k$ とする。
- v の子を、部分木の大きさの降順にソートする。
- 部分木の大きい最大 k 個の子への辺を、色 $C_1, C_2, ..., C_k$ で塗る。
- それ以外の子への辺は、今まで使ったどの色とも異なる新しい色で塗る。

この塗り方は (☆) を満たすことが、次のように示されます。

根から葉に降りていくことを考える。今いる頂点を v として、根から v へのパスに使われている色の集合を T とする。$|T| > 0$ であり、かつ次に向かう子 w への辺の色が T に含まれていないとき、

$$(v \text{ を根とする部分木の大きさ}) \geq |T| \times (w \text{ を根とする部分木の大きさ}) + 1$$

が成立する。もし、根と葉を結ぶパスが 8 種類以上で塗られているならば、

$$N \geq 7! + 6! + \ldots + 1! = 5913$$

だが、この問題での N は、 $N \leq 5555$ を満たす。したがって、根と葉を結ぶパスは 7 種類以下の色で塗られている。

以上より、条件を満たす色の塗り方と、各辺の迂回パスを 1 つ求めることができました。

Median Query

= J. Joy with Permutations

writter: moririn2528

October 10, 2020

概略

まず、最初の 4 つ a_0, a_1, a_2, a_3 に対して、タイプ 1 の質問を 4 回使います (3 回でも可能ですが、4 回で説明します)。質問の答えより、上位を値が大きい方から 2 つ、下位を値が小さい方から 2 つとすると、上位の添え字集合、上位の min、下位の添え字集合、下位の max が得られます。この 4 つの情報を基本情報と呼ぶことにします。

この情報とタイプ 1 の質問を 2 回使って元の 4 つと新たな a_i のうち、一つの添え字と値のペア、そしてその他 4 つに対する基本情報を得ることができます。これが場合分けによってできます。

また、基本情報と値集合がわかれば、上位 2 つ、下位 2 つにタイプ 2 の質問をそれぞれすることで添え字と値のペアがわかります。これによって隠された順列を判定することができます。

解説

$a_i, a_j, a_k, a_l (a_i < a_j < a_k < a_l)$ に対して, 下位の添え字集合を $\{i, j\}$、下位の max を a_j、上位の添え字集合を $\{k, l\}$、上位の min を a_k とします。そして、この 4 つの情報をまとめて (i, j, k, l) の基本情報と呼ぶことにします。

また、タイプ 1 (i, j, k) で質問し、Alice が答えた数値を $f(i, j, k)$ とします。

(i, j, k, l) の基本情報は a_i, a_j, a_k, a_l に対して質問 1 を 4 回することで得られます。具体的には、$a_i < a_j < a_k < a_l$ のとき、$f(i, j, k) = f(i, j, l) = a_j, f(j, k, l) = f(i, k, l) = a_k$ より、同じ答えが返ってくる添え字集合の共通部分を取ることで、上位、下位の添え字集合がわかります。

次に、(i, j, k, l) の基本情報はわかっているときに、a_x が未知である x を取ってきて、i, j, k, l, x のうち 1 つの添え字に対応する値と、それ以外 4 つに対する基本情報をタイプ 1 の質問 2 回で得ます。

(i, j, k, l) に対して、下位の添え字集合を $\{i, j\}$、下位の max を L、上位の添え字集合を $\{k, l\}$、上位の min を H とし、それ以外の情報は知らないとします。タイプ 1 (i, k, x) で質問します。

- $f(i, k, x) < L$ のとき、$a_k > L$ から、$a_i, a_x < L$ となります。a_i か a_j どちらかは L であるので、$a_j = L$ となり、$\max(a_i, a_x) = f(i, k, x)$ となり、(i, x, k, l) の基本情報全てそろいます。

- $f(i,k,x) = L$ のとき、順列なので $a_i = L$ です。また、$a_i < a_k$ より $a_x < a_i$ となります。よって、(j,x,k,l) の基本情報は下位の max 以外はわかっています。下位の max を知るためにタイプ 1 (j,x,k) で質問します。
- $L < f(i,k,x) < H$ のとき、$a_x = f(i,k,x)$ です。
- $f(i,k,x) = H$ のとき、$f(i,k,x) = L$ と同様にできて、$a_k = H$、(i,j,x,l) の基本情報は上位の min をタイプ 1 (i,x,l) で質問して知れば ok です。
- $H < f(i,k,x)$ のとき、$f(i,k,x) < L$ と同様、$a_l = H$、(i,j,k,x) の基本情報全てそろいます。

これを $n-4$ 回繰り返すことによって、(i,j,k,l) の基本情報と、(i,j,k,l) 以外の添え字 x に対する値 a_x がわかります。すると、値の集合 $\{a_i, a_j, a_k, a_l\}$ も知ることができます。(i,j,k,l) の上位の添え字集合に対してタイプ 2 の質問をし、下位の添え字集合に対してタイプ 2 の質問をすることで、a_i, a_j, a_k, a_l をソートした時の添え字の順番がわかります。これより a_i, a_j, a_k, a_l もわかり、隠された順列の値を全て知ることができます。

想定部分点解法 1

タイプ 1 を $3n$ 回以下、タイプ 2 を 2 回使って解く方法を示します。これは基本的に解説と同じです。質問の仕方が異なるだけです。

(i,j,k,l) の基本情報はわかっているときに、a_x が未知である x を取ってきて、i,j,k,l,x のうち 1 つの添え字に対応する値と、それ以外 4 つに対する基本情報をタイプ 1 の質問 3 回で得ます。

タイプ 1 の質問を (i,j,x) と (k,l,x) でします。$p = f(i,j,x), q = f(k,l,x)$ とします。

- $p = L, q = H$ のとき、$L < a_x < H$ です。タイプ 1 の質問を (i,k,x) とかですると、a_x が返ってきます。
- $p < L$ のとき、$a_x < L$ です。タイプ 1 の質問を (i,k,x) でして、$r = f(i,k,x)$ とします。$r = L$ のとき $a_i = L$ であり、$r \neq L$ のとき $a_j = L$ となります。

  $a_i = L$ のとき、$\max(a_j, a_x) = p$ であるので、(j,x,k,l) の基本情報がそろっています。$a_j = L$ の時も同様です。
- $q > H$ の時も上記と同様にすることができます。

余談　この問題が生まれたときのタイプ 1 のクエリ回数上限は $6n$ 回でした。時間を経るうちに上限が $4n, 3n, 2n$ 回と減っていき、今回の問題となりました。上に示した解法は上限が $3n$ 回だった時の想定解です。

$2n$ 回で出来ることがわかっていれば $3n$ 回から $2n$ 回にするのは難しくないと思っていたため、当初部分点を設定しない方針でした。しかし、解く側からするとそこまで簡単ではないうえ、いろんな人に考えてもらえるよう、部分点をつけることになりました。タイプ 1 の上限が $3n$ 回あれば意外と様々なやり方があると思います。

想定部分点解法 2

タイプ 1 を $2n$ 回以下、タイプ 2 を 3 回使って解く方法を示します。

これは解法とすこし異なります。セット (i, j) を持ちます。a_i, a_j はまだ数字が入っていないとします。タイプ 2 の質問を (i, j) でして、$a_i < a_j$ を仮定します ($a_i > a_j$ の時は i, j を swap する)。タイプ 1 の質問を 2 回することで、まだ数字が入っていない a_x と a_i, a_j 3 つのうち 1 つに数字をいれます。

まずタイプ 1 の質問を (i, j, x) でします。

- $a_k = f(i, j, x)$ となる k が存在しないとき、$a_x = f(i, j, x)$ とします。
- $a_k = f(i, j, x)$ となる k が存在するとき

  $a_i = f(i, j, x)$ か $a_j = f(i, j, x)$ となります。$a_i = f(i, j, x)$ となるとき、$a_i < a_j$ より、$a_k < f(i, j, x)$ かつ $a_x < f(i, j, x)$ となり、$a_j = f(i, j, x)$ となるとき、$f(i, j, x) < a_k$ かつ $f(i, j, x) < a_x$ となります。よって、タイプ 1 の質問を (i, k, x) でして、$f(i, k, x) < f(i, j, x)$ であれば $a_i = f(i, j, x)$、そうでなければ $a_j = f(i, j, x)$ となります。

  $a_i = f(i, j, x)$ のとき、セットを (x, j) とし、$a_k = f(x, j, k)$ とします。$f(x, j, k) = f(i, k, x)$ であるので、質問をする必要はありません。$a_j = f(i, j, x)$ のときも同様にセットを更新します。

この操作が終了したあと、セットが (i, j), $a_x = 2$, $a_y = n-1$ とします。このとき、i, j, x, y 以外の添え字 k での数字 a_k は正しくなっています。$a_i = 1, a_x = 2$ か $a_i = 2, a_x = 1$ かどうか、また $a_j = n-1, a_y = n$ か $a_j = n, a_y = n-1$ かどうか、はわからないので、質問 2 を 2 回使って確定させます。

余談　これは、Inside Story の タイプ 3 だけのジャッジの解法に思いを馳せると生まれる解法だと思います。想定部分点解法 1 くらいの難易度があると思ったので、これも想定部分点解法としました。

満点解法別解

想定部分点解法 2 の改良です。

最初のセット (i, j) を決めるときに、タイプ 2 を 1 回使うのではなく、タイプ 1 を 4 回使います。想定解法の最初のタイプ 1 を 4 回使うやり方をすると、上位の添え字集合と下位の添え字集合が得られるので、それぞれの集合から 1 つずつ取ってきてセットにします。これで満点を取ることが可能です。

余談　tatyam さんの提出コードを見るまで想定部分点解法 2 から満点解法を得れるとは思いませんでした。もしこれが作問時に思いついていたら、Inside Story の問題はなくなっていたかもしれませんね。

KUPC2020 K: Deleting Edges

= K. Kingdoms and Quarantine

原案, 作成: yamunaku

C_i の偶奇が等しい 2 頂点を結ぶ辺を赤色、異なる 2 頂点を結ぶ辺を青色で塗ります。

ひとつの頂点 v に注目します。操作において取り除く辺のうち、端点の一方が v であるようなものについて、その色を取り除く順番で並べると、赤・青・赤・青・……となるはずです。したがって、取り除く辺の集合 U を決めたとき、U は次の条件を満たさなければなりません。

どの頂点 v についても、

$$\begin{aligned}&(U \text{ に含まれ、端点の一方が } v \text{ である青い辺の数})\\ \leq\ &(U \text{ に含まれ、端点の一方が } v \text{ である赤い辺の数})\\ \leq\ &(U \text{ に含まれ、端点の一方が } v \text{ である青い辺の数}) + 1\end{aligned}$$

を満たす。

実はこの条件は、U が取り除く辺の集合であるための必要条件だけではなく、十分条件にもなっています。(証明は省略)

この条件を満たす U のうち、サイズ $|U|$ が最も大きいものを求めましょう。入力されるグラフの辺の番号の集合を E とし、各辺 $i \in E$ について、最終的にその辺を取り除いているかどうかを表す変数 $x_i = 0, 1$ を考えます。$x_i = 1$ のとき、辺 i は取り除かれていて、$x_i = 0$ のとき、辺 i は残っていることにします。さらに、頂点の集合を V 、 赤い辺の番号の集合を R 、青い辺の番号の集合を B とすると、次のような 0-1 整数計画問題になります。

$$\sum_{i \in E} x_i \to \text{最大化}$$

$$\sum_{\substack{i \in B \\ v \in \{a_i, b_i\}}} x_i \leq \sum_{\substack{i \in R \\ v \in \{a_i, b_i\}}} x_i \leq 1 + \sum_{\substack{i \in B \\ v \in \{a_i, b_i\}}} x_i \quad (v \in V)$$

$$x_i = 0, 1 \quad (i \in E)$$

スラック変数 $s_v = 0, 1$ $(v \in V)$ を導入すると、この問題は次のように書き換えることができます。

$$\sum_{i \in E} (-1) \times x_i + \sum_{v \in V} 0 \times s_v \to \text{最小化}$$

$$\sum_{\substack{i \in R \\ v \in \{a_i, b_i\}}} x_i = s_v + \sum_{\substack{i \in B \\ v \in \{a_i, b_i\}}} x_i \quad (v \in V)$$

$$x_i = 0, 1 \ (i \in E), \quad s_v = 0, 1 \ (v \in V)$$

さらに、$1 \leq a_i \leq N_1$、$N_1 + 1 \leq b_i \leq N_1 + N_2$ を踏まえると、

$$\sum_{i \in E} (-1) \times x_i + \sum_{v \in V} 0 \times s_v \to \text{最小化}$$

$$s_v + \sum_{\substack{i \in B \\ v = a_i}} x_i = \sum_{\substack{i \in R \\ v = a_i}} x_i \quad (1 \leq v \leq N_1)$$

$$\sum_{\substack{i \in R \\ v = b_i}} x_i = s_v + \sum_{\substack{i \in B \\ v = b_i}} x_i \quad (N_1 + 1 \leq v \leq N_1 + N_2)$$

$$x_i = 0, 1 \ (i \in E), \quad s_v = 0, 1 \ (v \in V)$$

となります。ここで、2 番目の式と 3 番目の式について和をとると、

$$\sum_{1 \leq v \leq N_1} s_v = \sum_{N_1 + 1 \leq v \leq N_1 + N_2} s_v$$

が成立します。したがって、この問題は次のように言い換えることができます。

赤い辺は a_i から b_i への向きをつけ、青い辺は b_i から a_i への向きをつける。これらの辺の容量を 1, コストを -1 とする。
新たな頂点 s, t を用意する。頂点 s から頂点 v $(1 \leq v \leq N_1)$ へ、容量 1, コスト 0 の辺を張る。頂点 v $(N_1 + 1 \leq v \leq N_1 + N_2)$ から頂点 t へ、容量 1, コスト 0 の辺を張る。
s から t への、流量任意の最小費用流を求めよ。

適切に負コストの辺を除去すると、この問題は Primal-Dual を用いて $O((N + M)^2 \log(N))$ で解くことができます。

Inside Story of Median Query

= L. Lazy Judge

writter: moririn2528

October 9, 2020

概略

質問が来るたびに、最後に答える順列を少しずつ確定させていきます。ただしこのとき、1 から順に確定させていくようにします。そうすることで、質問 1 だと多くても 2 つの数字、質問 2, 3 だと 1 つの数字さえ割り振れば答えることができます。なぜなら、1 から数字を割り振るため、後の質問で割り振られることになる数は今割り振った数字より大きく、今の質問の答えに影響しないからです。

ここで、質問が来たとき、いくつかの場合において、すぐに数字を確定させてしまうのではなく、「どれかにこの数字が入っているよ」、という情報を持たせることにします。この情報は 2 つのパターンがあって、a_i, a_j, a_k のどれかに p と q が割り振られているよという情報 (以下サイズ 3 の揺らぎと呼ぶ)、a_i, a_j のいずれかに p が割り振られているよという情報 (以下サイズ 2 の揺らぎと呼ぶ) があります。質問が来たときに、揺らぎが被らないようにサイズを減らしたり、揺らぎを消したり、新たに揺らぎを作ります。

サイズ 3 の揺らぎのポテンシャルを 2、サイズ 2 の揺らぎのポテンシャルを 1 とし、順列のポテンシャルを数字が固定されていない要素の数の 2 倍とすると、質問 1,2 ではコスト (ポテンシャル減少量) 2、質問 3 ではコスト 1 となります。このポテンシャルは、配列の自由度のようなものです。これは、Bob の残りスタミナ値 S 以上となるので、条件を満たす順列を作ることが可能となります。順列 $b_1, b_2, \ldots, b_n$ の作り方は、揺らぎ内の数字をシフトして、揺らぎで指定されていた数字以外すべてをシフトすることで作ることができます。

解説

質問が来たときに、1 から順に数字を確定させていくようにします。さらに、数字を確定させてしまうのではなく、「どれかにこの数字が入っているよ」、という情報を持たせることにします。

ここで以降の解説の簡略化のため、様々な定義をします。

サイズ 3 の揺らぎ a_i, a_j, a_k どれかに p と q が割り振られているという情報、「$(\{i, j, k\}, (p, q))$」$(p < q)$ とも表す。

サイズ 2 の揺らぎ a_i, a_j いずれかに p が割り振られているという情報、「$(\{i, j\}, (p))$」とも表す。

揺らぎ サイズ 3 の揺らぎ、サイズ 2 の揺らぎの総称。

揺らぎがかぶる サイズ 3 の揺らぎ同士がかぶる、とは、$(\{i,*,*\},(*,*))$ と $(\{i,*,*\},(*,*))$ がともに存在することを指す。$*$ はそれぞれ任意の数字を表す。サイズ 2,3 の揺らぎがかぶる、サイズ 2 の揺らぎ同士がかぶる、も同様に定義され、それらの総称を指す。

i **が揺らぎに含まれる** $(\{i,*,*\},(*,*))$ か $(\{i,*\},(*))$ が存在すること。

i **を含む揺らぎ** i が揺らぎに含まれないときは存在しない。i が揺らぎに含まれるとき、$(\{i,*,*\},(*,*))$ か $(\{i,*\},(*))$ で存在している揺らぎを指す。揺らぎがかぶらなければ、存在しても 1 つしかない。

$(\{i,j,k\},(p,q))$ **から** (i,p) **を削除** このサイズ 3 の揺らぎをサイズ 2 の揺らぎ $((\{j,k\},(q))$ にし、$a_i = p$ とすることを表す。

$(\{i,j\},(p))$ **から** (i,p) **を削除** このサイズ 2 の揺らぎを削除し、$a_i = p$ とすることを表す。

揺らぎから (i,p) **を削除** i が含まれる揺らぎから (i,p) を削除することを表す。このコストは 1 となる。

揺らぎの最小値 揺らぎが $(\{i,j,k\},(p,q))$ のとき $\min(p,q)$、$(\{i,j\},(p))$ のとき p とする。

質問が来たときに、揺らぎが被らないようにサイズを減らしたり、揺らぎを消したり、新たに揺らぎを作ります。具体的には以下のようにします。さらに、サイズ 3 の揺らぎのポテンシャルを 2、サイズ 2 の揺らぎのポテンシャルを 1 とし、順列のポテンシャルを数字が固定されていない要素の数の 2 倍として、操作のコストも考えます。

順列は最初 INF で初期化されているとします。

質問 1 が (i,j,k) で来たとき

- $(\{i,j,k\},(p,q))$ が存在すれば、q を出力。コスト 0
- $(\{i,j\},(p))$ が存在するとき
  - k が揺らぎに含まれないとき
    まず、a_k に数字が割り振られていなければ割り振る。そのあと $\max(a_k,p)$ を出力。コスト 2
  - k が揺らぎに含まれるとき
    k が含まれる揺らぎの最小値を q とする。その揺らぎから (k,q) を削除。$\max(p,q)$ を出力。コスト 1

  $(\{j,k\},(p))$, $(\{k,i\},(p))$ の時も同様にする。
- $(\{i,j,x\},(p,q))$ が存在するとき
  - k が揺らぎに含まれるとき、r を k が含まれる揺らぎの最小値とする。
  - k が揺らぎに含まれないとき、$r = a_k$ とする。

3 つのペア $(p,i),(q,j),(r,k)$ を作り、これを昇順ソートした 3 つのペアを $(P,I),(Q,J),(R,K)$ とする。I を含む揺らぎから (I,P) を削除、J を含む揺らぎから (J,Q) を削除。Q を出力する。

揺らぎから削除を 2 回しているので、コストは 2。

$(\{j,k,x\},(p,q))$, $(\{k,i,x\},(p,q))$ の時も同様にする。

- それ以外のとき、i,j,k のうち 2 つ以上含む揺らぎは存在しない。

  i が含まれる揺らぎがあればその最小値を p, なければ $p=a_i$ とし、j,k も同様、q,r を作成。

  3 つのペア $(p,i),(q,j),(r,k)$ を作り、これを昇順ソートした 3 つのペアを $(P,I),(Q,J),(R,K)$ とする。

  - $I=\mathrm{INF}$ のとき、i,j,k いずれかを含む揺らぎが存在しない上、a_i,a_j,a_k に数字が割り振られていない。この時は数字 $x,x+1$ を割り振って、新たにサイズ 3 の揺らぎ $(\{i,j,k\},(x,x+1))$ を作成。$x+1$ を出力。コスト 2
  - $J=\mathrm{INF}$ のとき、J,K いずれかを含む揺らぎが存在しない上、a_J,a_K に数字が割り振られていない。この時は数字 x を割り振って、新たにサイズ 2 の揺らぎ $(\{J,K\},(x))$ を作成。さらに、I が含まれる揺らぎがあれば、それから (I,P) を削除。1 から順に数字を割り振っているので $P<x$ であることから、x を出力。コスト最大 2
  - それ以外のとき I が含まれる揺らぎがあれば、それから (I,P) を削除し、同様に J が含まれる揺らぎがあれば、それから (J,Q) を削除して、Q を出力。コスト最大 2

同様に質問 2,3 も場合分けしていきます。質問 2,3 は質問 1 の場合分けを一部削除して少し改良するだけでできるので、詳しくは書きません。自分で頑張ってください (書くのが大変になりました)。

質問 2 (i,j) について、質問 1 では 2 つのペアを揺らぎから削除をして固定していたところを 1 つのみでよく、i を含む揺らぎ、j を含む揺らぎともにない場合は a_i か a_j に数字を割り振れば答えることができます。

質問 3 (i,j) について、質問 2 同様 1 つのペアを揺らぎから削除をして固定すればよく、i を含む揺らぎ、j を含む揺らぎともにない場合は新しい数字 x を割り振ってサイズ 2 の揺らぎ $(\{i,j\},(x))$ を作成します。

順列 $b_1,b_2,\ldots,b_n$ の作り方について、揺らぎ内の数字をシフトして、揺らぎで指定されていた数字以外すべてをシフトすることで作ることができます。

つまり、以下の操作をします。

1. まだ割り振っていない数字の集合を $C=\{c_i\}_{i=1,2,\ldots,m}$ とし、$c_{m+1}=c_1$ とする。
2. 以下の操作ができなくなるまでする。

   t 回目の操作とする。

- 処理していない揺らぎがあるとき
  $(\{i,j,k\},(p,q))$ の時は $a_i = p$, $a_j = q$, $\boldsymbol{a_k = c_t}$, $\boldsymbol{b_i = c_{t+1}}$, $b_j = p$, $b_k = q$ とする。
  $(\{i,j\},(p))$ の時は $a_i = p$, $a_j = c_t$, $b_i = c_{t+1}$, $b_j = p$ とします。
- 処理していない揺らぎが存在せず、$a_i = \mathrm{INF}$ である i があるとき
  $a_i = c_t$, $b_i = c_{t+1}$ とする

注意: i が揺らぎに含まれず、かつ、$a_i = \mathrm{INF}$ となる i が一つしかないことがあります。i が揺らぎに含まれ、 $a_i = \mathrm{INF}$ となる i も含めてシフトの処理をしないといけません。(太文字の部分の処理)

これによって作られた 2 つの順列は、揺らぎの部分を含めた、数字を固定していない部分に関して要素が異なります。つまり、揺らぎの部分を含めた、数字を固定していない部分の添え字集合を S_I とすると、$\forall i \in S_i$, $a_i \neq b_i$ となります。この操作によって条件を満たした数列を作れているのかどうか確かめます。

まず、質問による Bob のスタミナ減少量はコスト以上なので、Bob のスタミナ減少量の合計は順列と揺らぎのポテンシャルの減少量以上になります。最初の Bob のスタミナと最初の順列と揺らぎのポテンシャルは同じであるので、質問が終わったときの順列と揺らぎのポテンシャル合計 Φ は S 以上になります。

揺らぎの部分を含めた、数字を固定していない部分の要素数を z、サイズ 3 の揺らぎの個数を x、サイズ 2 の揺らぎの個数を y とすると、

$$z = 3x + 2y + (m - x - y) = m + 2x + y$$

$m + 2x + y$ は整数であるので、

$$m + 2x + y \geq \left\lceil \frac{1}{2}(2m + 2x + y) \right\rceil = \left\lceil \frac{\Phi}{2} \right\rceil \geq \left\lceil \frac{S}{2} \right\rceil$$

となり、以上の操作によって作られた 2 つの順列は条件を満たします。

M: Many Parentheses `= M. Multiple Parentheses`

$DP[i][j]$ を i 個の箱に (が合計 j 個あるような括弧列の詰め方の総数と定義します。$DP[i][j]$ を $O(1)$ で求めることができれば、条件の「長さが K と一致する括弧列をいれてはいけない」の部分は包除原理を用いて対処できます。

したがって、$DP[i][j]$ を $O(1)$ で求められるように考えます。

まず、 (が i 個ある括弧列の数を C_i とします。これはカタラン数と呼ばれており、 $C_i = \sum_{1 \leq j < i}(C_j * C_{i-j})$ を満たします。
i 個目の箱に何個 (が入っているかを考えると、次の式が成立します。

$$DP[i][j] = \sum_{0 \leq k \leq j} (DP[i-1][j-k] * C_k) \tag{1}$$

これをさらに $DP[i-1][j-k]$ の方で展開すると、

$$\begin{aligned}
DP[i][j] &= \sum_{0 \leq k \leq j} (DP[i-1][j-k] * C_k) \\
&= \sum_{0 \leq k \leq j} \left(\sum_{0 \leq l \leq j-k} (DP[i-2][j-k-l] * C_l * C_k) \right) \\
&= \sum_{0 \leq k+l \leq j} (DP[i-2][j-(k+l)] * C_l * C_k) \\
&= \sum_{0 \leq x \leq j} (DP[i-2][j-x] * C_{x+1})
\end{aligned}$$

となります。(最後の等式はカタラン数の漸化式から導かれます。) ところで (1) より

$$\begin{aligned}
DP[i-1][j+1] &= \sum_{0 \leq x \leq j+1} (DP[i-2][j+1-x] * C_x) \\
&= DP[i-2][j+1] + \sum_{1 \leq x \leq j+1} (DP[i-2][j+1-x] * C_x) \\
&= DP[i-2][j+1] + \sum_{0 \leq x \leq j} (DP[i-2][j-x] * C_{x+1})
\end{aligned}$$

であるので、

$$DP[i-1][j+1] = DP[i-2][j+1] + DP[i][j]$$

が成立します。書き直すと、

$$DP[i][j] = DP[i-1][j] + DP[i+1][j-1] \tag{2}$$

が成立します。

この漸化式を用いれば、$O(N^2)$ で $DP[i][j]$ の値を前計算できます。この際、初期値として

$$DP[i][0] = 1 (0 \leq i)$$
$$DP[0][j] = 0 (0 < j)$$

を用いることになります。

さて、さらにここから考察を深めます。$i >= j$ について $DP2[i][j] := DP[i-j][j]$ と定義すると、(2) から

$$DP2[x][y] = DP2[x-1][y] + DP2[x][y-1] \tag{3}$$

が導けます。この遷移式を使う時、初期値は $x = y$ の部分と $y = 0$ の部分で与えられており、$x = y = 0$ のとき 1、$x = y$ かつ $y > 0$ のとき 0 、$y = 0$ のとき 1 となります。

これは「途中で合計が負にならないように $+1$ を $x-1$ 個、-1 を y 個並べる方法の数」と言い換えられるので、(カタラン数を求めるときに盤面を折り返すテクを使うのと同様の手法を用いることで)

$$DP2[x][y] = (x-1+y, y) - (x-1+y, y-1)$$

が導かれます。

結論として、

$$DP[i][j] = (i-1+j+j, j) - (i-1+j+j, j-1) \ (i > 0)$$
$$DP[i][j] = 1 \ (i = j = 0)$$
$$DP[i][j] = 0 \ (i = 0 \text{ かつ } j > 0)$$

という事実が導かれ、これは $O(1)$ で計算できます。

(追記) この問題は多項式について考えることで $O(MlogM)$ で解くことが可能です。私はこの解法に気付かず $N, M <= 10^6$ と設定し、本番では多くのチームにこの解法で通されてしまいました。今度からこういった問題では $N, M <= 10^7$ にすべきだと反省しました。