JOIG2021/2022 春合宿 Day1 – JOIG ツアー

(JOIG Tour)

解説： 米内山 匠実

問題概要

- 東西に伸びる直線上に J，O，I，G の4種類の文字が N 個置いてある。各文字には座標(西からの距離)が定まっている。

- クエリが Q 個与えられる。j 番目のクエリは以下の通り。

・座標 S_j と T_j が与えられる。S_j, ‘J’, ’O’, ’I’, ‘G’, T_j の順に巡る経路の最短距離を求めよ。J~G の各文字はどこにあるものでも良い。

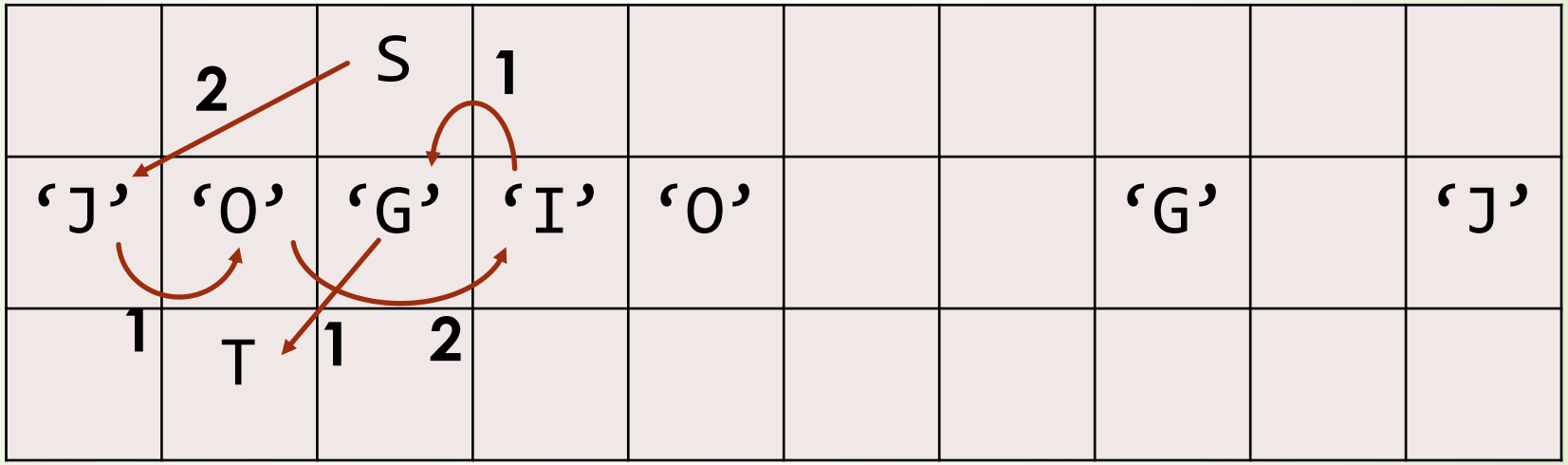
制約(満点を狙う場合)

- 置いてある文字の数 $N \leq 100000$
- クエリの個数 $Q \leq 100000$
- 各文字の座標、及び始点と終点 S_j, T_j は 10^{15} 以下(とにかく**大きい**)

- 各文字は1つ以上存在
- 各文字の座標は相異なる&昇順に与えられる
- (S_j, T_j) の組み合わせは各クエリで異なる

例(入出力例1)

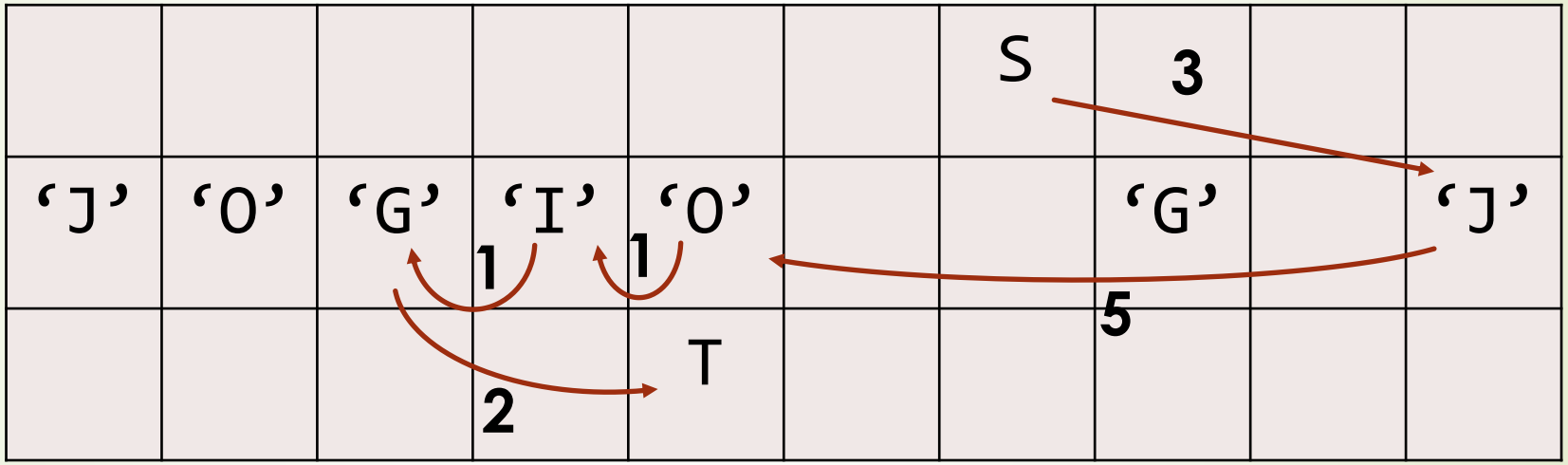
クエリ1



合計: 7

例(入出力例1)

クエリ2



合計: 12

考え方①

- 全探索

考えられる全ての経路のうち最短のものを求めたい。

→ 経路を**全探索**！

- J，O，I，G の選び方を全て調べる

それぞれの文字の個数を N_J, N_O, N_I, N_G とすると、それらの積の通り数だけ選び方がある。

- それぞれの選び方について、経路の長さを考える

始点と終点と各文字の位置が決まるから、各経路の長さは簡単！

その経路のうちの最短を求めればOK。

考え方①

- J，O，I，G の選び方を全て調べる

それぞれの文字の個数を N_J, N_O, N_I, N_G とすると、それらの積の通り数だけ選び方がある。

小課題1を考えると……

全ての文字の個数は合わせて $N \leq 80$

→ 選び方は高々 20^4 通りくらい。

クエリの数は $Q \leq 10$ なので、**小課題1は解ける。**

考え方②

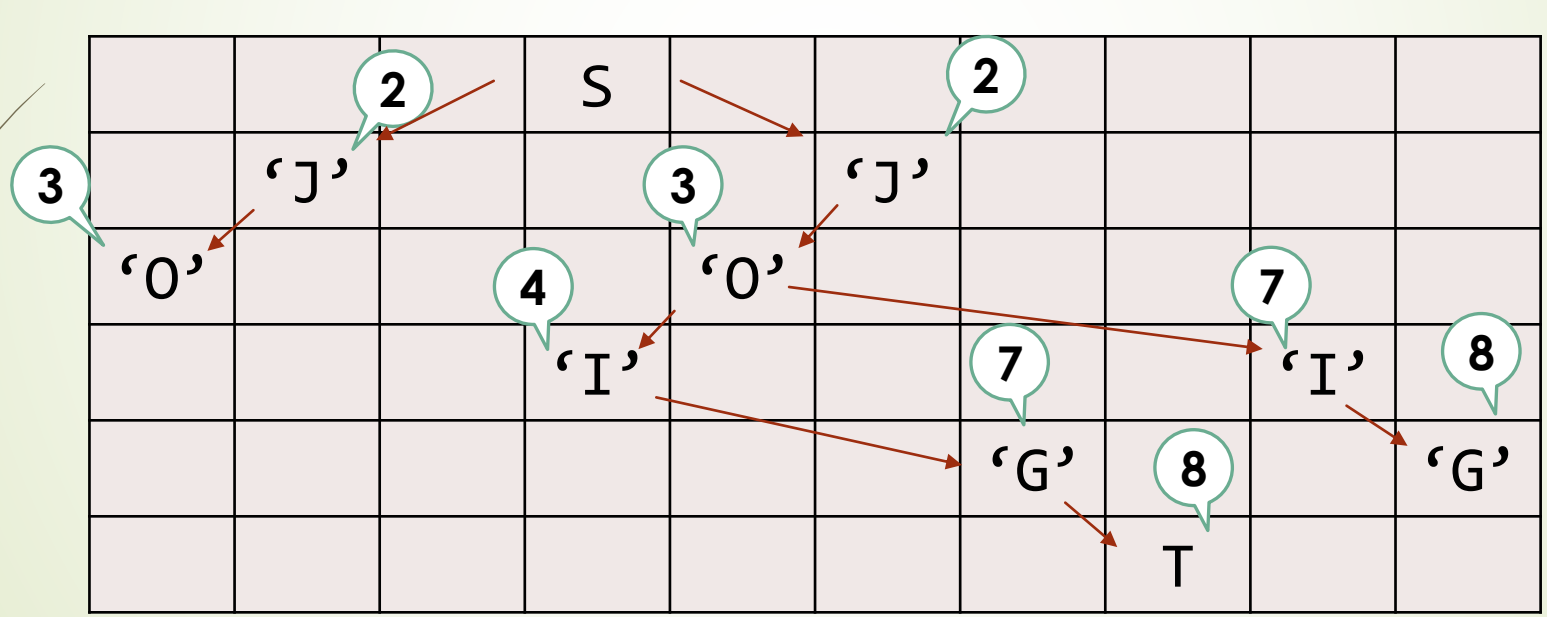
- 全探索では、小課題2以降は間に合わない……

→ 全てを探索する必要はあるのか？

- 【重要】求めたいのは**最短経路**のみ

- それぞれの文字にたどり着くための最短経路をメモしながら進めばいい。

考え方②



Jから
O,I,Gの
順に
見ていく

考え方②

- 文字の種類ごとに分けておく。

- J を見る時はそれぞれ S からの距離を記録
- O を見る時は、「ある J を経由した時の距離」を全ての J について探索し、最短を記録
- I を見る時は、「ある O を経由した時の距離」を全ての O について探索し、最短を記録
- G を見る時は、「ある I を経由した時の距離」を全ての I について探索し、最短を記録

- 最後に、全ての G について、そこまでの最短と T からの距離を合わせて最短を決定。

考え方②

- 計算量？

1クエリごとに、全ての J-O、O-I、I-G の組み合わせを調べることになる。(それぞれ $O(N^2)$)

それと、 S から各 'J'、各 'G' から T までの組は $O(N)$

→ 合わせて $O(N^2)$

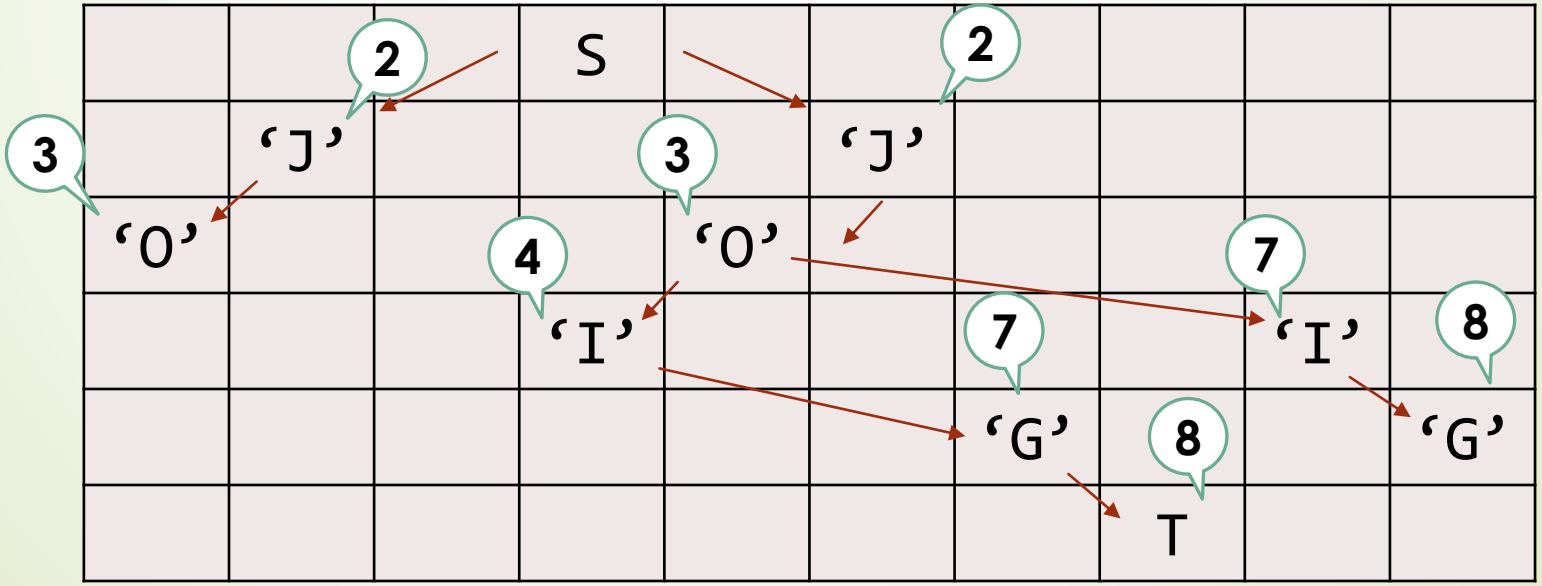
クエリ Q 個だと $O(N^2Q)$ ……**小課題3**まで解けた！

(J-O、O-I、I-G の組は実際は N^2 より結構少ない)

考え方③

- ここまで考えてきたのって、**グラフ**っぽい？

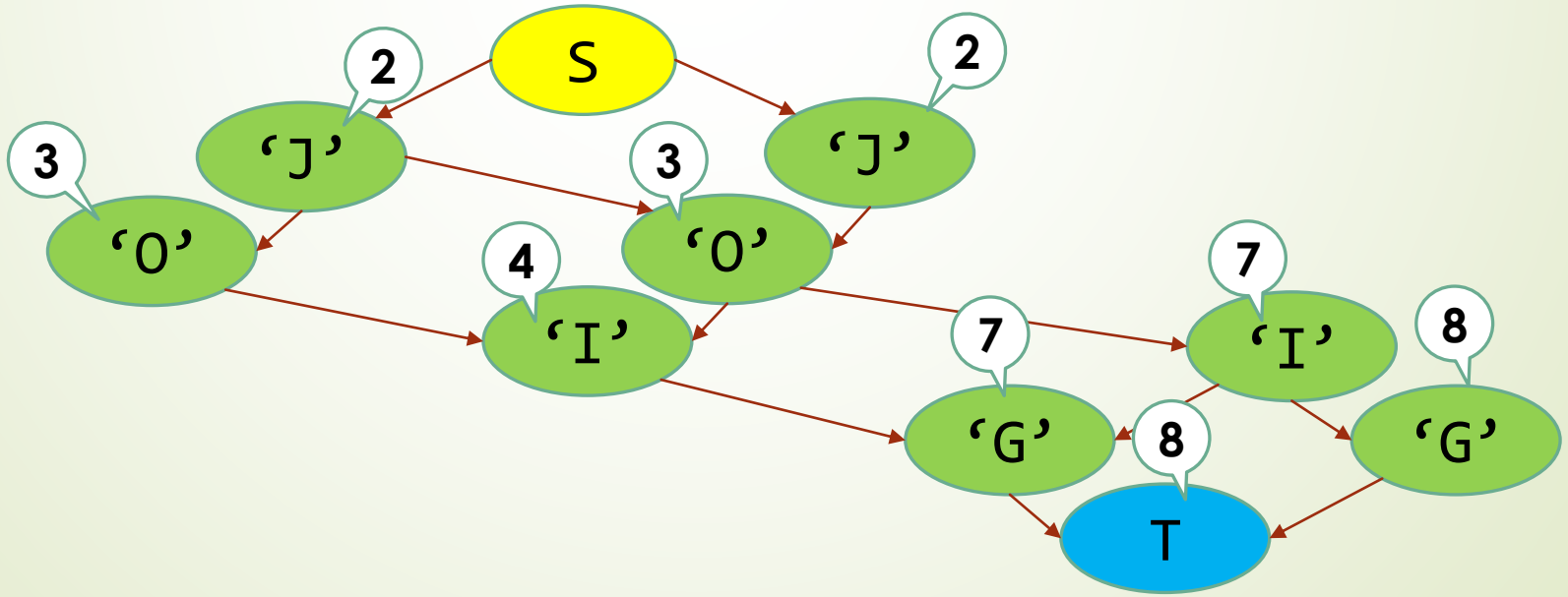
- J-O, O-I, I-G 間をグラフの辺と見なし、各文字+始点終点をグラフの頂点と見なせば……



考え方③

- ここまで考えてきたのって、**グラフ**っぽい？

- J-O, O-I, I-G 間をグラフの辺と見なし、各文字+始点終点をグラフの頂点と見なせば……



考え方③

- グラフの最短経路と言えば……**ダイクストラ**！

- ということで、頂点と辺を設定し、ダイクストラをする。
- クエリごとに始点と終点の頂点は決める必要があるが、それ以外は同じグラフである。

- 頂点は(始点と終点を除き) $O(N)$ 個、辺は $O(N^2)$
- よって、始点と終点を決めた1回の探索では $O(N^2)$

- 結局改善にはならない……

考え方③の補足

- 最短経路をグラフ化して考えるアプローチは、しばしば新たな解法の筋道になる。

- ダイクストラの計算量は $O(E + V \log E)$ ただし V は頂点数、E は辺数。
- 今回は辺が多いので E がネックになった。
- グラフ化してダイクストラをするのではなく、単純にDPをすれば $O(\log E)$ の分を省ける場合もある。

- 結論：柔軟に考えよう！

考え方④

- グラフで考えると $O(N^2)$ の辺を用意することになった。

でも……動き方ってそんなにたくさん考える必要あるの？

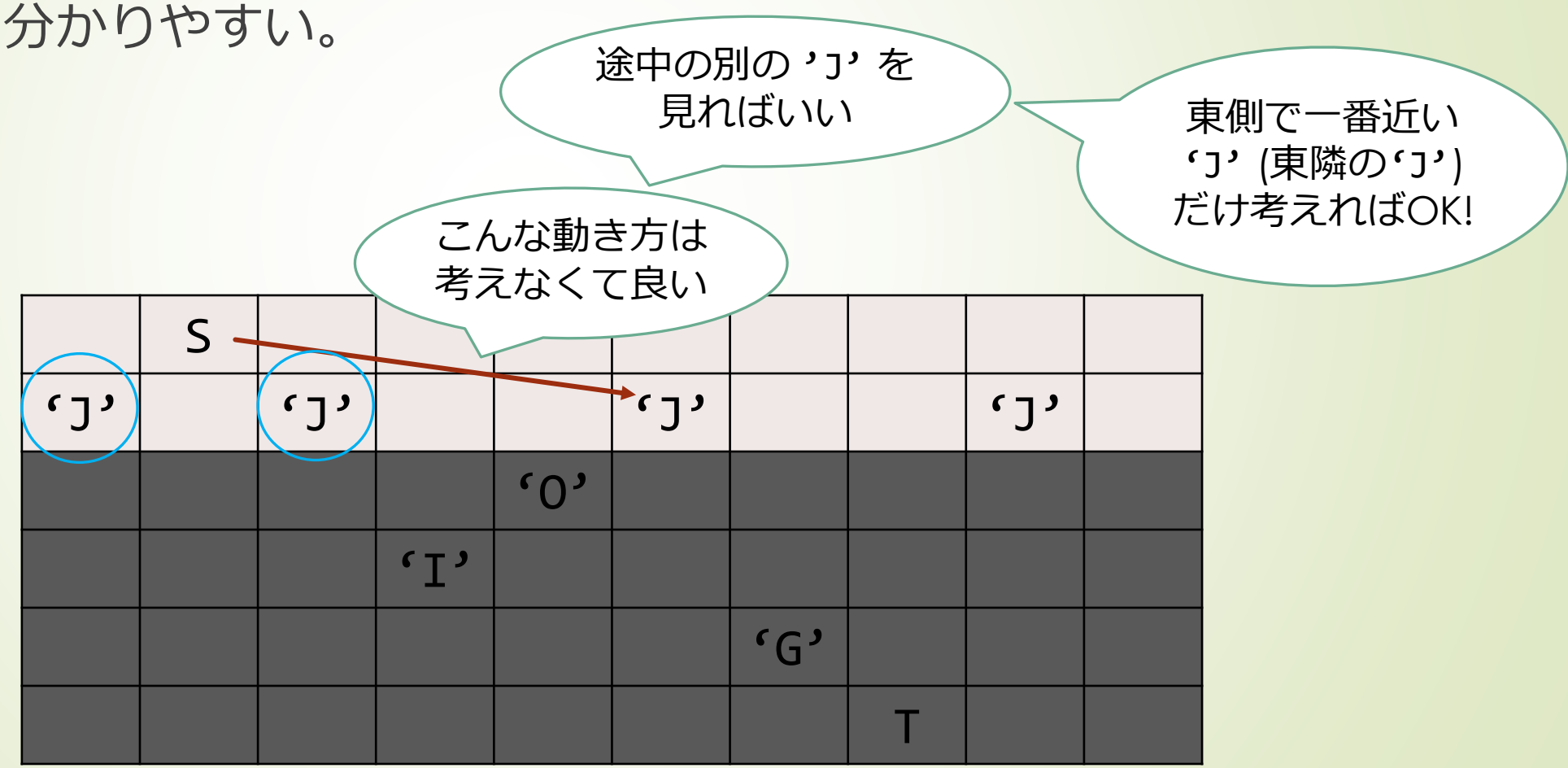
考え方④

- 例えば**小課題5**のような状況(J しか選ぶ余地がない)で考えると分かりやすい。

	S								
‘J’		‘J’			‘J’			‘J’	
				‘O’					
			‘I’						
						‘G’			
							T		

考え方④

- 例えば**小課題5**のような状況(J しか選ぶ余地がない)で考えると分かりやすい。



考え方④

- 例えば**小課題5**のような状況(ｺ しか選ぶ余地がない)で考えると分かりやすい。

- 「S から見て西隣の 'ｺ' か、東隣の 'ｺ' を選ぶ、2つの場合だけ考えればいい」ことに気付けば**小課題5**は解ける。

- 西隣 or 東隣の 'ｺ' は二分探索(lower_bound, upper_bound)で求められる。(後で補足)
- 計算量は1クエリあたり $O(\log N)$ であり、全体で $O(Q\ \log N)$ なので小課題5の制約に間に合う。

考え方④

- **小課題6**も同様。T から見て西隣の ’G’ か東隣の ’G’ かだけ考えればいい。

	S								
‘J’		‘J’			‘J’				
				‘O’					
			‘I’						
	‘G’					‘G’			
							T		

東隣がなければ無視

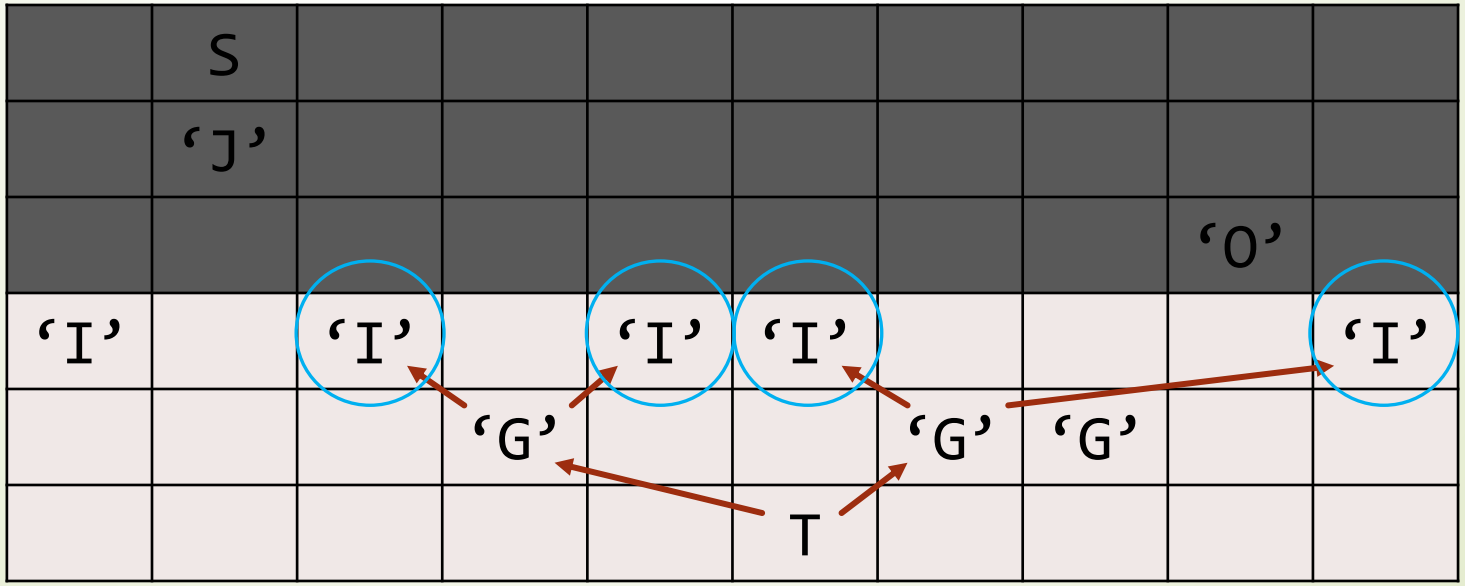
考え方④

- **小課題6**も同様。T から見て西隣の ’G’ か東隣の ’G’ かだけ考えればいい。
- S 側の選び方と T 側の選び方は互いに独立なので、それぞれ最小となるものを選べばいい。
- 計算量は1クエリあたり $O(\log N)$ であり、全体で $O(Q\ \log N)$ なので小課題6の制約に間に合う。

考え方④

- 西隣・東隣だけでいい、という考え方を発展させる。

- **小課題7**のように、’I’ も ’G’ も選べる場合は、T から見て西隣と東隣の2つの ’G’ について、さらに西隣の ’I’ と東隣の ’I’ を考えればいい！



考え方④

- 西隣・東隣だけでいい、という考え方を発展させる。
- **小課題7**のように、’I’ も ’G’ も選べる場合は、T から見て西隣と東隣の2つの ’G’ について、さらに西隣の ’I’ と東隣の ’I’ を考えればいい！
- 考えるべきは T からの4通りの「隣の隣」までの行き方と S からの2通りの「隣」までの行き方。
- これも計算量は1クエリあたり $O(\log N)$ であり、全体で $O(Q\ \log N)$ なので制約に間に合う。

考え方④

- ここまで来れば、同様に考えて……

- **満点解法**

S から両隣の ’J’ までの経路長を求め、
さらにその2つの ’J’ について、両隣の ’O’ までの経路長を求め、
さらにその4つの ’O’ について、両隣の ’I’ までの経路長を求め、
さらにその8つの ’I’ について、両隣の ’G’ までの経路長を求め、
その16通りの ’G’ について、T までの経路長を求め、

以上16通りで最も短かったものを出力すれば良い

考え方④

- 西隣・東隣を求めるパート

→ lower_bound, upper_bound で $O(\log N)$

- 1回のクエリでは、16通り(定数倍程度)の経路しか考えず、その経路は西隣・東隣を4回求めるだけでいいから、計算量は $O(\log N)$
- Q 回のクエリでは、$O(Q\ \log N)$ →**十分間に合う**

実装

- lower_bound, upper_bound について

Vector や set などのデータ構造には、ある値以上となる最小の要素を二分探索で求める標準機能が備わっている。それが lower_bound である。また、ある値を超えるものを知りたいなら upper_bound が使える。

説明すると長くなるし、説明はネットにたくさん転がっているので、そちらを参照してもらいたい。

キーワード：イテレータ (位置を指し示すもの)

実装

```
//例えば、Sの西隣・東隣を探索するコード
long long S;
vector<long long> J;
//配列Jには、文字Jの入っている位置が記録されているとしよう

auto it = lower_bound(J.begin(),J.end(),S);
long long L,R;
if(it == J.begin()) L = -1; //西隣が存在しない
else{
    it--;
    L = *it;
}

it = upper_bound(J.begin(),J.end(),S);
if(it == J.end()) R = LLONG_MAX; //東隣が存在しない
else R = *it;
```

実装

- ちなみに、自力で二分探索を実装するのもあり

自分で書けることもそれはそれで重要。

- 二分探索を諦めても点はもぎ取れる

想定解のうち二分探索だけできなくても、**小課題4**は取れる！

- 16通りを全探索する方法について

bit演算を用いることもできるが、西隣か東隣かという2択が4つあるので4重ループにしたり、for文をあきらめて愚直に16行書いても良い。

余談

・二分探索(lower_bound, upper_bound)を扱えるようになろう。

・テクニックが追い付かないなら、部分点を取ろう。

・「結局一番近いところだけ調べればいい」という考え方は頻出。

得点分布