KUPC2013 問題 I

σ

原案：楠本

解答：楠本，小浜

解説：楠本

解説編集：田村

Swap query

(0-indexed とする) ある順列 σ0 に対し，σ0 の i 番目と (i+1) 番目の要素をひっくり返したものを σ1 とする．このとき 2 つのクエリ結果 Guess(σ0) と Guess(σ1) の違いを考える．

```
          i (i+1)

σ0 = [... a   b   c d e f ...]

σ1 = [... b   a   c d e f ...]
```

のようになっていたとする． π 上での値 x の位置を π^-1 で表すことにする． もし π^-1(a) <= i, π^-1(b) <= i であるとき，

```
                    i

π  = [... a ... b ... c ...]
```

のようになっているとすると， Guess(σ0) = { .... (i-π^-1(a)) ... (i+1-π^-1(b)) ... } Guess(σ1) = { .... (i+1-π^-1(a)) ... (i-π^-1(b)) ... } となる．すなわち，クエリ結果は「どれか一要素が増加してどれか一要素が減少する」． これにより，もし π^-1(a) <= i, π^-1(b) <= i であると仮定した場合は π^-1(a) と π^-1(b) の値を特定することができる．

また，π^-1(a) >= i+1, π^-1(b) >= i+1 のとき，

```
        (i+1)

π = [... c ... a ... b ... ]
```

のようになっているが，このときもまた Guess(σ0) = { .... (π^-1(a)-i) ... (π^-1(b)-i-1) ... } Guess(σ1) = { .... (π^-1(a)-i-1) ... (π^-1(b)-i) ... } となり，もし π^-1(a) >= i+1, π^-1(b) >= i+1 であると仮定した場合は π^-1(a) と π^-1(b) の値を特定することができる.

上記のいずれでもない場合. たとえば π^-1(a) <= i, π^-1(b) >= i+1 のようなとき, Guess(σ0) = { .... (i-π^-1(a)) ... (π^-1(b)-i-1) ... } Guess(σ1) = { .... (i+1-π^-1(a)) ... (π^-1(b)-i) ... } となり，2 つの変化値がともに 1 だけ増加する. この場合両者を区別できないので (π^-1(a), π^-1(b)) としてはちょうど 2 通り存在する. π^-1(a) >= i+1, π^-1(b) <= i のときも同様である. このときは変化値がともに 1 だけ減少する.

以上をまとめると次のようになる.

Guess(σ0)と Guess(σ1)の結果(変化があったところのみ)	(π^−1(a), π^−1(b)) のとりうる値
X→X+1, Y→Y−1	(i−X, i+1−Y) or (i+X, i+1+Y)
X→X+1, Y→Y+1	(i−X, i+1+Y) or (i−Y, i+1+X)
X→X−1, Y→Y−1	(i+X, i+1−Y) or (i+Y, i+1−X)

なので普通にスワップしただけだと完全に特定することはできない(ので困る). しかしたとえば取りうる値のうち片方が負数とか N 以上になっていた場合，その可能性は排除できる. つまり，もし X >= i+2 or Y >= i+2 なら一意な特定が可能で，これは i が小さいときには高い確率で発生しうる. これを利用する.

Algorithm

最初にランダムな置換 σ0 を生成する. σ1 を σ0 の 0, 1 番目の要素をスワップさせたもの, σ2 を σ1 の N-1, N-2 番目の要素をスワップさせたもの, σ3 を σ2 の 2, 3 番目の要素をスワップさせたもの, σ4 を σ3 の N-3, N-4 番目の要素をスワップさせたもの, σ5 を σ4 の 4, 5 番目の要素をスワップさせたもの, … とし, σ0, σ1,... をクエリしていく. 上記考察から, σ(2i) or σ(2i+1) のときに一意的な決定に失敗する十分条件は π^-1(σ0[2i])<= 2i+1 かつ π^-1(σ0[2i+1]) <= 2i+1 である. σ0 はランダムな順列であることから，これが発生する確率は高々 4i^2/n^2 である.

t をある整数(あとで決める)として, σt までこれをやる，という試行を At とする. 一意な決定に失敗したら，そこからさらに 2 回の追加のクエリをして完全に特定するようにする. At を

n/(2t) 回繰り返せば全ての列を完全に特定することができる. (2 回目以降の At では, 既に位置が特定された数字は真ん中に寄せて端には現れないようにランダムな順列を発生させる.)

平均クエリ回数を計算する.

```
E[Q] = n/(2t) * (1 + t + Σ_{i=1→t/2} 2*2*4i^2/n^2)

     <= n/(2t) * (1 + t + 2t^3/(3n^2))
```

で, t=n^(2/3) とするとこれは

```
E[Q]  <= n/2 + (5/3) * n^(1/3)
```

となる.