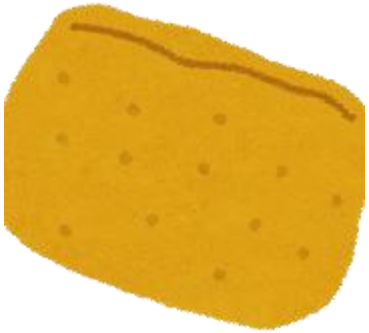
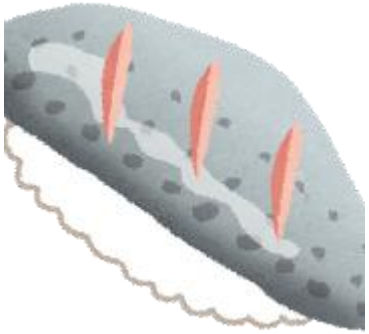
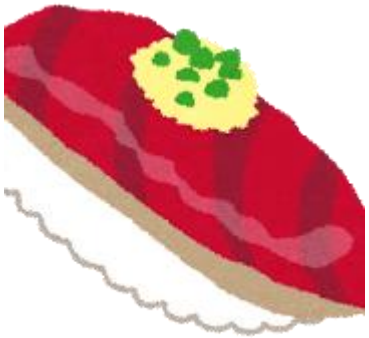
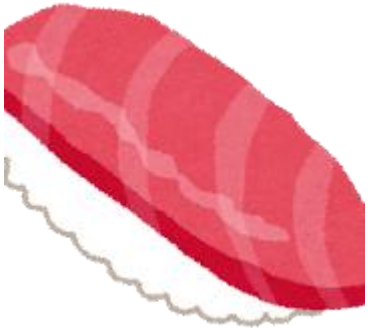
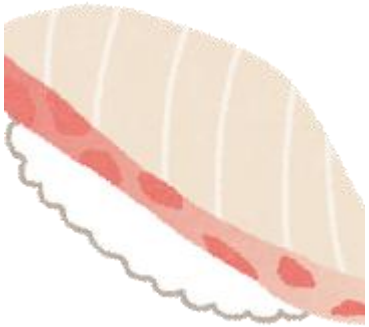





回転寿司 解説

DEGwer


問題概要

- 円環状に皿を持った人がN人並んでいる
- i人目の人の持っている皿の価格は最初x_i
- 次の処理を行った時の変数xの値を答えよというクエリQ個に答えよ
- $x = p_i$(ベルトコンベア上の皿の価格を表す)
- $for(k: s_i, \ldots, t_i) if(x < x_k) swap(x, x_k)$
  - (より安い皿が目の前にあれば交換)
- $N \leq 400000, Q \leq 25000$

問題概要

- 円環状に人がN人並んでいて、皿を1枚ずつ持っている
- i人目の持つ皿の価格は最初x_i
- 各人は、ベルトコンベアの自分の目の前に皿がある場合、自分の皿の価格と比較して小さいほうを手元に置き、大きいほうをベルトコンベアに乗せる
- 人sに皿を渡し、人tから皿を回収したとき、回収
- $N \leq 400000, Q \leq 25000$

とりあえず

- 直線として考えます
- 皿の経路が0地点を通るときは、そのクエリを始点からN番目までと1番目から終点までに分けて考えればいい

小課題1(5点)

- 書いてある通りに実装してください
- $O(NQ)$となり、5点が得られる

考察

- まず、ベルトコンベアの気持ちになって考えてみる

考察

- 各クエリに対し、ベルトコンベアに最後に乗っている皿の価格は、操作に関係した皿の価格の最大値になっているはず
  - そうでなければ、交換されているはずなので矛盾

- 正確には、区間$[s, t]$に、$x = p$としてクエリの操作を行った場合、最後には$\max(x_s, \ldots, x_t, p)$が$x$に入っているはず

小課題2(15点)

- $s_i = 1, t_i = N$
- クエリごとに$\max(x_1, \ldots, x_N, p)$を求めればよい
  - そこで求まった価格最大の皿は消滅するので、それ以降操作にかかわらないので、適当にないことにしてやる

小課題2(15点)

- これはpriority_queueでできることそのもの
  - priority_queueにpをpush
  - priority_queueのmaxを出力してpop
- を各クエリに対してやればいい
- $O(Q log\ N)$で、15点が得られる
  - 適当に分岐をすれば20点(575)

満点解法

- $N \leq 400000, Q \leq 25000$

考察

- 今度は人の気持ちになって考えてみる
- 同じ区間にたくさんクエリが来たとき、各人は価格いくらの皿を持っているか?
- 正確には、同じ区間に$x = p_1, p_2, \ldots, p_k$の順にクエリの操作を施した時、残る皿の列はどうなっているか?

考察

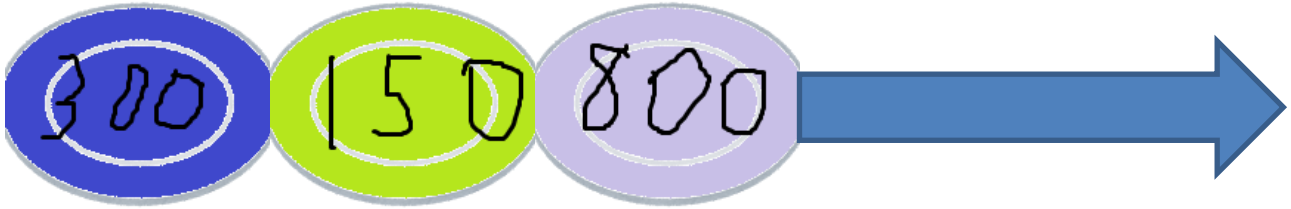
- ここで、交換の様子を見てみる
- 交換操作は、人の手元の皿とベルトコンベアの人の前にある位置の皿の価格を比べ、
- 大きいほうをベルトコンベアに
- 小さいほうを人の前に
- 置く操作と言える

- これは人の列とベルトコンベアについて対称！

考察

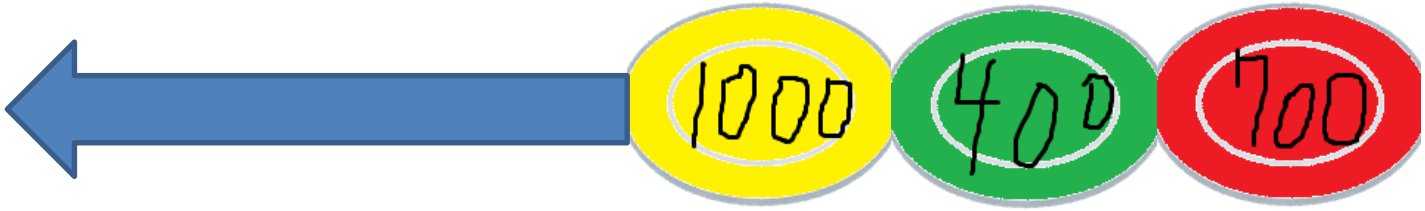
- つまり、同じ区間に対するクエリの操作は、
- 「人の列の前をベルトコンベア上の皿が通過する」とも
- 「ベルトコンベア上の皿の列の前を人が通過する」とも考えることができる

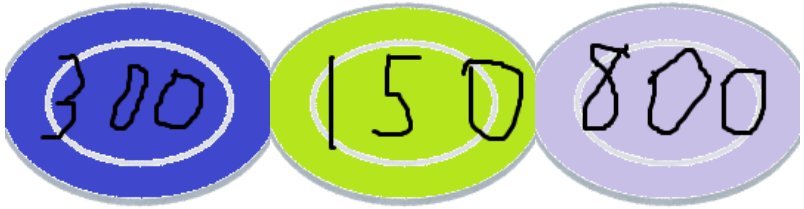
考察



皿がこの順にこの区間を通過するとする

考察





この操作は、人がこの順にベルトコンベアの前を通過すると考えても問題ない

考察

- つまり、同じ区間に対するクエリの操作において、小課題2と同じ操作によって、一連の操作後の各人が持っている皿の価格を高速に求めることができる
  - クエリで来る皿の列をpriority_queueに入れて同様の操作をすることで実現
  - ただし、符号だけ反転
- 具体的には$O((\text{区間長})\log(\text{クエリの数}))$

満点解法

- いい性質がわかったので、ここで平方分割を考えます
- 2通りのやり方がある
  - ①列の平方分割
  - ②クエリの平方分割

- 両方紹介します
- 本質的にはだいたい同じ

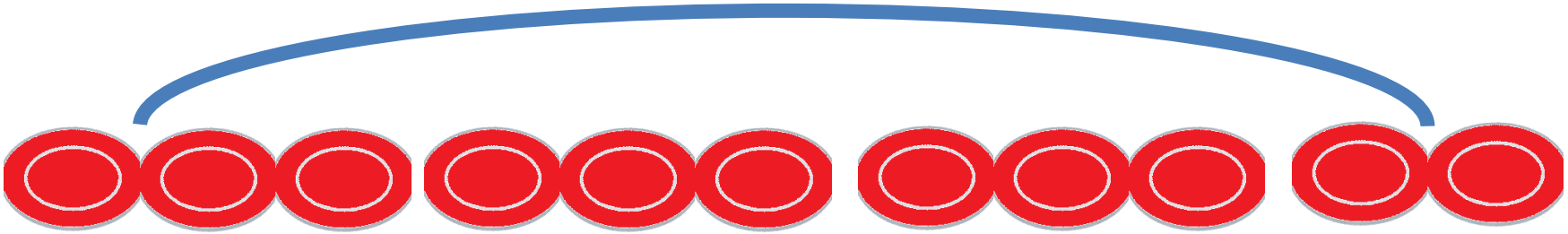
満点解法①

- 列の平方分割

- 人の列を連続B個ごとのバケットに分割する
- 各区間についてpriority_queueを持ち、その区間の人のもつ皿の価格を全部入れておく

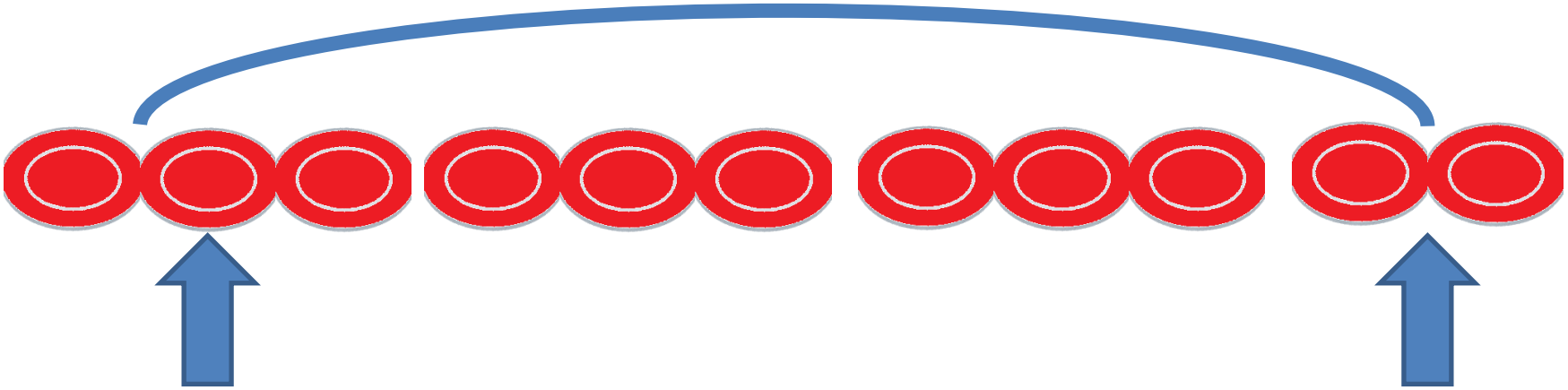
満点解法①

- この区間にクエリが来るとする



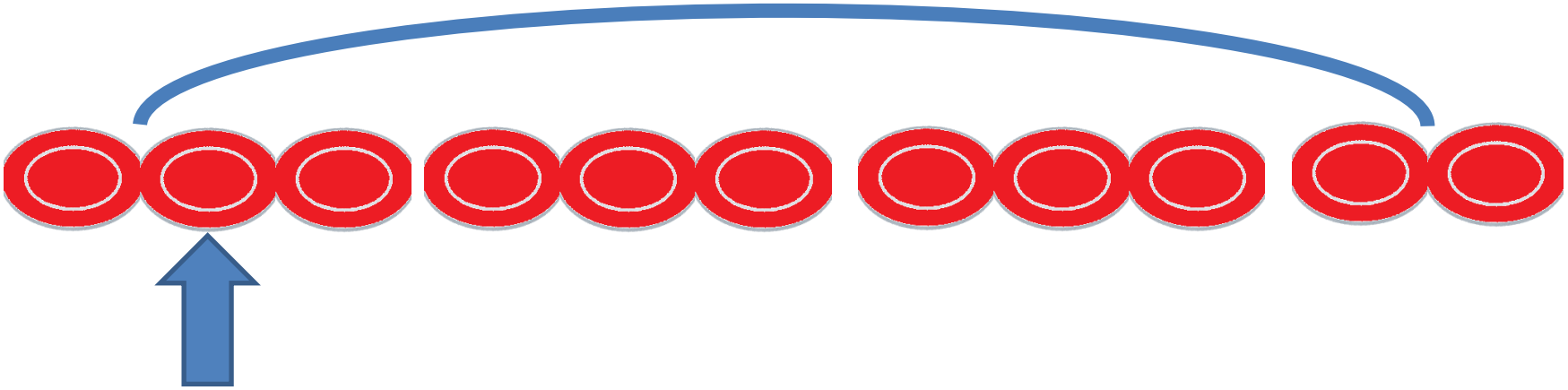
満点解法①

- 区間の端が存在するバケットでは、あらかじめ後述の操作をしておくことで、各皿の価格がわかるようにしておく



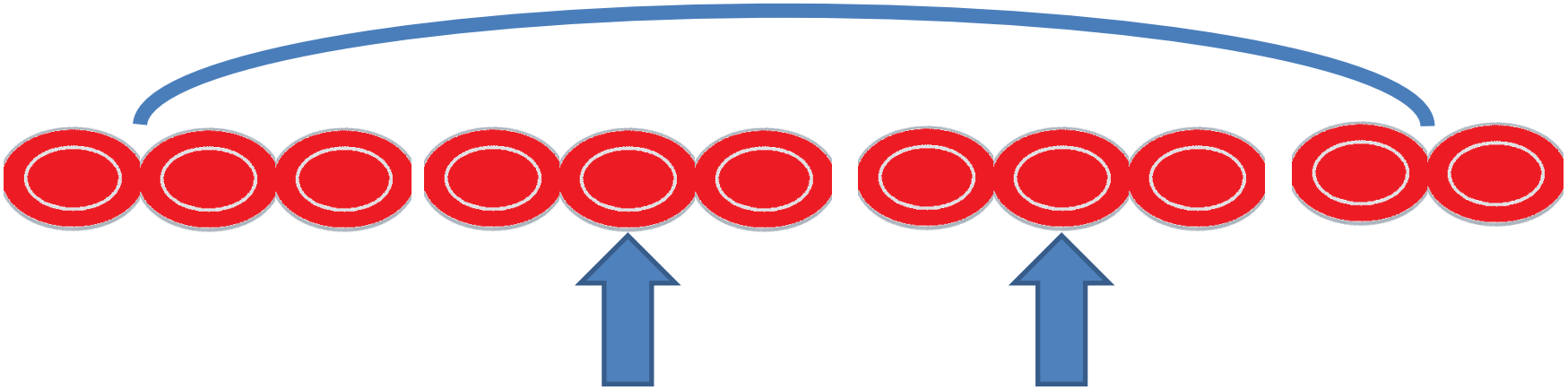
満点解法①

- このバケットについては、愚直に操作を行う



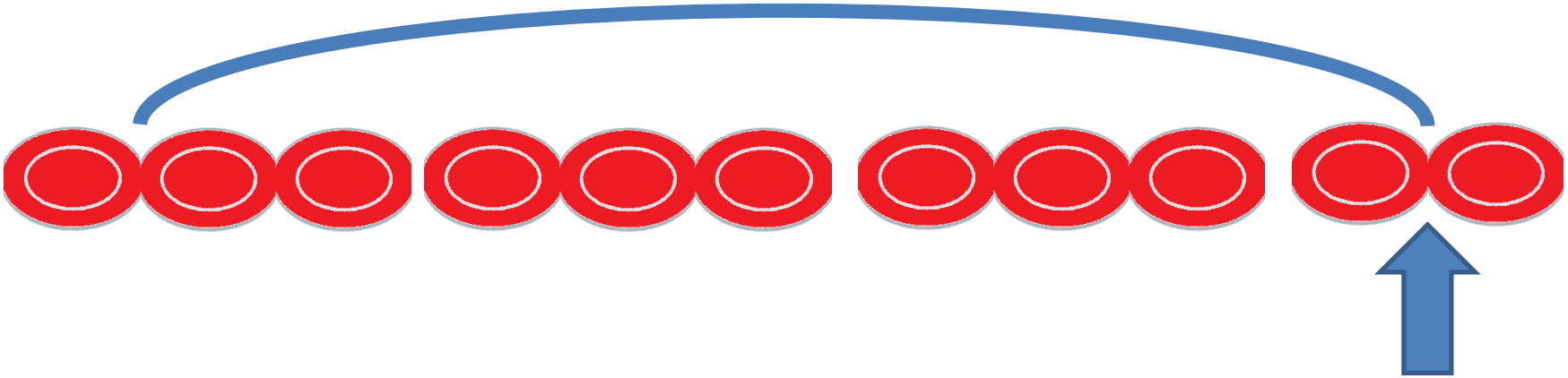
満点解法①

- これらのバケットについては、前のバケットから出てきた値をその区間の持つpriority_queueに入れ、最大値を取り出して次のバケットに渡す
  - ついでに諸事情によりその区間に入ってきた値をvectorに入れておく



満点解法①

- このバケットについても、愚直に操作を行う



満点解法①

- 区間の皿の価格の列を復元する操作がまだ残っている
- いま、元の価格列の値と、その区間に入ってきた皿の価格の値をすべて持っておいているので、これは先ほど考察した状況と一緒
- よってこの操作はならし$O(B log\ Q)$でできる

満点解法①

- 以上をまとめると、
- バケットの端のある区間に対し、$O(B\log Q)$
- そうでない区間に対し、$O(\frac{N}{B}\log B)$
- で操作ができる
- $\mathrm{B} = \sqrt{N}$とすれば、クエリあたりならし
  $O(\sqrt{N}(\log N + \log Q))$、全体で
  $O(Q\sqrt{N}(\log N + \log Q))$で満点が得られる

満点解法②

- クエリの平方分割
  - たぶんこっちのほうが実装が楽で、定数倍も速いです
  - ただしオフライン
- クエリを連続B個ごとのバケットに分割する
  - こうしておくと、このバケットに現れるクエリの区間の始点と終点の個数は高々$2B$個

満点解法②

- クエリの区間の端点の組で座標圧縮をし、できた各区間にpriority_queueを持たせる
- バケット内の各クエリに対し、priority_queueに適切な操作を行う
- バケット内の各クエリには、$O(\mathrm{Blog}\ \frac{N}{B})$で答えられる

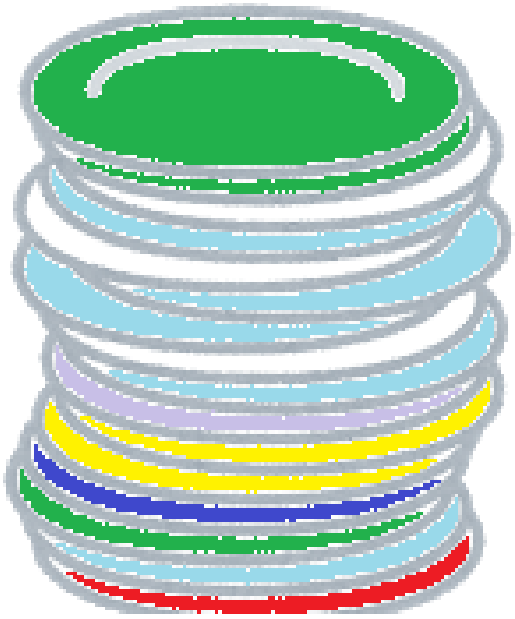
満点解法②

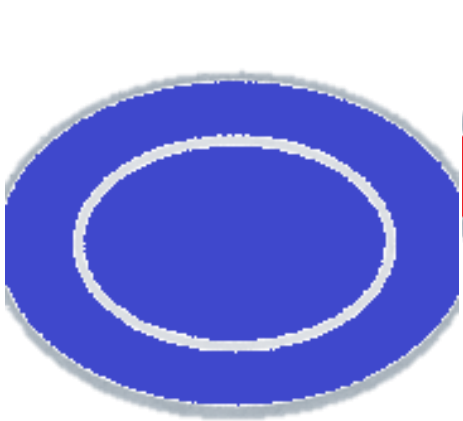
- 各バケットを処理したら、元の配列を復元する
- この操作は、先ほどと同様にやると$O(Nlog\ B)$でできる

満点解法②

- まとめると、各バケットについて
- $\mathrm{O}(B^2 \log \frac{N}{B} + Nlog\ \mathrm{B})$でできる
- バケットは全部で$\mathrm{O}(\frac{Q}{B})$個あるので、
- 全体で$\mathrm{O}(QB \log \frac{N}{B} + \frac{QN}{B} \log \mathrm{B})$でできる
- $\mathrm{B} = \sqrt{N}$とすれば、全体で$\mathrm{O}(Q\sqrt{N} \log N)$となり、満点が得られる

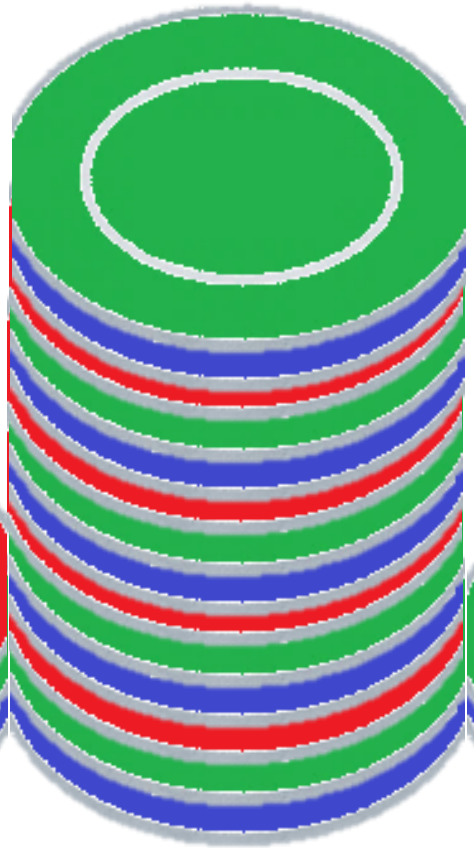
得点分布



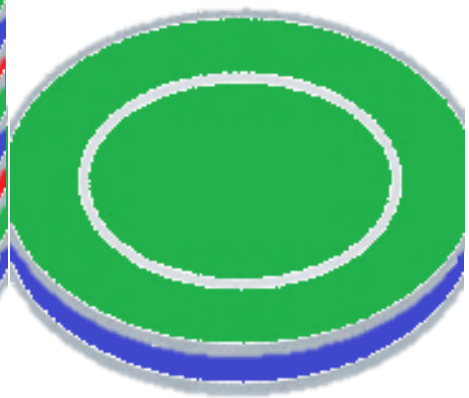


0点

5点



20点



100点