ABC 158 解説

writer: satashun evima, tempura0224

2020 年 3 月 7 日

For International Readers: English editorial will be published in a few days.

A: Station and Bus

S 内に A, B の両方が存在する場合は Yes、どちらかしかない場合は No です。
以下は Python での実装例です。

```
S = input()
if S[0] == S[1] and S[1] == S[2]:
  print('No')
else:
  print('Yes')
```

B: Count Balls

1 回の操作をセットで考えると考えやすいです。
N 個のうち、完全に含まれる操作の回数は $\lfloor \frac{N}{A+B} \rfloor$ 回です。
中途半端に含まれる操作は高々 1 回分ですが、N を $A+B$ で割った余りと A の大小関係が重要で、N を $A+B$ で割った余りを R とすると、$min(R, A)$ だけ含まれます。

以下は Python での実装例です。

```
N, A, B = map(int, input().split())
ans = N // (A + B) * A
rem = N % (A + B)
ans += min(rem, A)
print(ans)
```

C: Tax Increase

消費税が 8% のとき A 円、消費税が 10% のとき B 円となる金額を直接求めなくても、$1 \leq A \leq B \leq 100$ の条件から、最大でも考える必要のある金額は 1009 円であるため、小さい方から計算していけばよいです。

課題 : $O(1)$ で答えを求めてください。

D: String Formation

問題文の通りに直接シミュレーションするだけでは反転クエリが高速に処理できず、$O((N+Q)^2)$ となり遅いです。

$T_i = 1$ のとき実際に文字列を操作するのではなく、現在どちら側が先頭かという情報を持っておけば、$T_i = 2$ のクエリでもその情報を考慮することで、位置関係を保ったままシミュレーションできます。最後に全体を反転する必要がある場合に気をつけてください。

ここで、$C++$ などで普通の文字列型の先頭に追加する操作は計算量が悪いため、deque で文字列を管理するか、左右で分けて文字列を保持し最後に適切に連結するといった方法で対処できます。

E: Divisible Substring

まず、左端を固定して実際に P で割った余りを計算しながら右側を伸ばしていくことで、$O(N^2)$ で答えを求めることができます。

例えば文字列の左右を固定した場合に P で割った余りを計算できないか、定式化を考えてみましょう。

$U_i := S[i:N]$ (S の i 文字目以降) を評価した整数 と置きます。便宜上、$U_{N+1} = 0$ とします。(この時点ではまだ mod P で考えていません)

このとき、S の i 文字目から j 文字目を評価した整数は、$\frac{U_i - U_{j+1}}{10^{j+1-i}}$ と表せます。$(1 \leq i \leq j \leq N)$

ここで、P と 10 が互いに素かどうかが重要です。$P = 2, 5$ の場合は右端が P で割り切れるかどうかのみで決まるので、$O(N)$ で解けます。

$P \neq 2, 5$ とします。このとき、$\frac{U_i - U_{j+1}}{10^{j+1-i}}$ が P で割り切れることは、$U_i - U_{j+1}$ が P で割り切れることと同値です。よって、右側から U_i mod P を計算し、現在 x mod P であるような j が何個あるかを配列や map で管理しておくことで、$O(N + P)$ でこの問題が解けました。(今回は P が小さいですが、P が大きい場合は map を使いましょう)

F: Removing Robots

まず、高橋君が操作を行う順番は重要ではありません。

$f_i :=$ ロボット i が一連の操作の中で起動されたかどうかの bool 値、$g_i :=$ ロボットが高橋君によって直接起動されたかどうかの bool 値と置きます。このとき、$f_i := g_i OR f_j (j \neq i, X_j < X_i < X_j + D_j)$ と表せることがわかります。

全てのロボットが残っている場合に、ロボット i を起動した場合に連動して起動することになる最大の j を R_i で表します。(他のロボットに関与しない場合は、$R_i = i$ とします) このとき、

$R_i = max(i, R_j (i < j, X_i + D_i > X_j))$

が成り立つので、segment tree などを用いて R_i を X の大きい方から求めていけます。一部のロボットが取り除かれている場合も、起動した場合に同じ範囲のロボットが取り除かれた状態になることがわかります。

$dp_i :=$ ロボット i 以降のみを考慮した場合の組み合わせの個数と置きます。このとき、ロボット i を起動しない場合... dp_{i+1}、ロボット i を起動する場合 ... dp_{R_i+1} となり、X の大きい方から求まります。

以上の計算量は $O(NlogN)$ です。