ABC 158 解説

writer: satashun evima, tempura0224

2020 年 3 月 7 日

For International Readers: English editorial starts on page 7.

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、dequeue 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 $U_i \bmod P$ を計算し、現在 $x \bmod P$ 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 i 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、$f_i := g_i \,\mathrm{OR}\, f_j(j \neq i, X_j < X_i < X_j + D_j)$ 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。

A: Station and Bus

If S has both A and B, then the answer is Yes, and if it has only either of them, then the answer is No.

The following is a sample code in Python.

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

B: Count Balls

It is easy to think when one operation is considered as a set.
Among N balls, the number of operations that is completely included in is $\lfloor \frac{N}{A+B} \rfloor$.
The number of operations that is included halfway is at most one, but the remainder of N divided by $A+B$ is only important, and let R be the remainder of $A+B$ divided by N, then there are $min(R, A)$ balls.

The following is a sample code in Python.

```
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

Instead of directly finding the price such that the consumption tax is A and B yen when tax rate is 8% and 10%, by the condition of $1 \leq A \leq B \leq 100$, the maximum price that has to be considered is 1009 yen, so it is sufficient to calculate in the increasing order.

Task: Find the answer in a total of $O(1)$ time.

D: String Formation

By only performing simulation in accordance with the problem statement, the reverse query cannot be processed fast, and it will cost a total of $O((N+Q)^2)$ time, which is slow.

When $T_i = 1$, instead of actually manipulating the string, you can hold the state of which side is the beginning of the string so that, by considering the state also in the query of $T_i = 2$, the simulation can be done while maintaining the positional relation. Note that you may need to reverse the whole string at last.

Here, in $C++$ or some other programming languages, operation of appending to the ordinal string type has bad time complexity, so you can handle with it by managing the string with dequeue, or storing the string with two parts, left and right, and at last concatenating properly.

E: Divisible Substring

First, we can find the answer in a total of $O(N^2)$ time, by fixing the left end and actually calculating the remainder by P while extending the right end.

Let's consider a formularization that enables to calculate the remainder by P when, for example, the both ends of the string is fixed.

Let $U_i := S[i : N]$ (the i-th and later character of S), evaluated as an integer. For convenience, let $U_{N+1} = 0$. (At this point we are not considering mod P yet.)

Then, substring of S from i-th to j-th, evaluated as an integer, can be denoted as $\frac{U_i - U_{j+1}}{10^{j+1-i}}$. $(1 \leq i \leq j \leq N)$

Here, it is important whether P and 10 are coprime or not. If $P = 2, 5$, then it only depends on whether the right end is divisible by P, so it can be solved in a total of $O(N)$ time.

Assume that $P \neq 2, 5$. Then, $\frac{U_i - U_{j+1}}{10^{j+1-i}}$ is divisible by P if and only if $U_i - U_{j+1}$ is divisible by P. Therefore, by calculating U_i mod P from right to left, and managing the number of j such that x mod P with arrays or maps, the problem could be solved in a total of $O(N + P)$ time. (This time P is small, but when P is large, use map)

F: Removing Robots

First, the order of Takahashi's operations are not important.

Let $f_i :=$ a boolean value whether robot i was activated in the series of operations, and $g_i :=$ a boolean value whether robot i was directly activated by Takahashi. Then, it can be seen that $f_i := g_i \,\mathrm{OR}\, f_j (j \neq i, X_j < X_i < X_j + D_j)$.

Let R_i be the maximum j such that, when robot i is activated when all the robots are remaining, robot j is accordingly activated. (If it does not contribute to other robots, then let $R_i = i$.) Then, it holds that

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

so, by using data structures like segment tree, R_i can be calculated in the decreasing order of X. Even when some robots are removed, the robots in the same range will be removed when it's activated.

Let $dp_i :=$ the number of possible sets of robots, when considering only robot i and later. Then, if robot i is not activated... dp_{i+1}, and if robot i is activated... dp_{R_i+1}, and these can be calculated in the decreasing order of X.

The time complexity above is $O(NlogN)$ in total.