ABC043 / ARC059 解説

evima & sigma425

A - キャンディーとN人の子供イージー / Children and Candies (ABC Edit)

ループを回して1+2+..+Nを計算すればよい。また、これはN*(N+1)/2に等しいのでこれを出力してもよい。

B - バイナリハックイージー / Unhappy Hacking (ABC Edit)

素直に、キーが押されるたびに文字列の変化をシミュレートすれば解くことができます。

プログラミング言語に用意された文字列関連のライブラリを用いると簡潔に実装できるでしょう。多くの主要なプログラミング言語では文字列を演算子 "+" で連結でき、文字列の末尾に文字 '0' や '1' を挿入する処理はこれで済みます。文字列の末尾の 1 文字を削除する処理では、言語によって大きく違いが出るかもしれません。文字列から文字を直接消すのが難しければ、文字列から最後の 1 文字を除いた部分文字列をとって、それで元の文字列を上書きするという方法も考えられます。(この方法は文字列が長いと時間がかかるおそれがありますが、今回は問題ありません)

もしこのようなライブラリの使い方が分からなければ、文字の配列を文字列に見立てるなどして文字列の処理を「自前で」実装することもできます。その場合、現在の文字列の長さを保持する変数を用意すると見通しがよくなります。

C - いっしょ / Be Together

全て同じ整数に変えるのだが、変える先の整数 X を全探索してそのうちコストが最小のものを出力する。この時-100 から 100 までの整数しか考える必要はない。(-100 未満や 100 より大きくしても、各$(a_i-X)^2$ は X=-100 や 100 の時と比べ大きくなるだけなので、無駄。) 計算量は O(N*座標幅)。

D - アンバランス / Unbalanced

以下、文字列 s の長さを n とします。s には、長さ 2 以上の部分文字列が n(n-1)/2 個存在します。

部分点のデータセットでは n が最大で 100 と小さいため、s の部分文字列をすべて列挙することができます。それぞれの部分文字列について、各アルファベットの出現回数を数えることでアンバランスか判定すれば、時間計算量 $O(n^3)$ で問題を解くことができます。

満点を得るためには、n が 10^5 に達するケースにも正解する必要があります。この n の値では、すべての部分文字列を列挙する時間はありません。どうすればよいでしょうか？

実は、長さ 2 の "XX"（X は何らかの同じアルファベット）というパターンと、長さ 3 の "XYX"（Y は何らかのアルファベット）というパターンのいずれも s に存在しなければ、アンバランスな部分文字列は存在しません。なぜなら、もしこれらのパターンが存在しなければ、同じアルファベットの 2 つの文字の間に必ず他の文字が 2 つ以上挟まっていることになり、一種類のアルファベットが部分文字列の文字の過半数を占めることはないからです。よって、時間計算量 O(n) で問題を解くことができます。

E - キャンディーとN人の子供 / Children and Candies

まず $f(x_1,x_2,...,x_N)$が具体的にどのような意味のある値になるかを考える。幼稚園の活発度の式を考えると、N人にC個のキャンディーを分配する というのは、変数 x_iたちの多項式でどの変数にいくら次数を割り当てるか(総次数はC) というものだと思える。するとfは、N変数C次の単項式を集めたもの になる。
(例: N=3,C=3として、見やすさのため $x=x_1,y=x_2,z=x_3$と置くと、
$f(x,y,z)=x^3+y^3+z^3+x^2y+x^2z+y^2x+y^2z+z^2x+z^2y+xyz$)
まずは部分点を考える。$x_1,x_2,...,x_N$が($A_i=B_i$として)与えられるので、$f(x_1,x_2,...,x_N)$を求めればよい。
NやCが大きいので全ての単項式を列挙することは出来ない。次のようなDPを考える。dp[i][j]=「x_1からx_iまでの変数を使って次数がjになる単項式の値の和」
先程の例を使うと、

i \ j	0	1	2	3
0	1	0	0	0
1	1	x	x^2	x^3
2	1	x+y	x^2+xy+y^2	$x^3+x^2y+xy^2+y^3$
3	1	x+y+z	$x^2+y^2+z^2+xy+xz+yz$	$x^3+y^3+z^3+x^2y+x^2z+y^2x+y^2z+z^2x+z^2y+xyz$

実際は式ではなくこの値を保持する。
dp[N][C]が答えとなる。
dpの計算は、まずdp[0][0]=1とし、遷移はdp[i+1][j]= sum {dp[i][j-k]* x_i^k} (k=0~j)となる。(つまり、変数x_iで次数をいくつ使うかで場合分け)
すると計算量は$O(NC^2)$となり、N=C=400なので間に合う。

$$\sum_{x_1=A_1}^{B_1}\sum_{x_2=A_2}^{B_2}\cdots\sum_{x_N=A_N}^{B_N} f(x_1,x_2,\ldots,x_N)$$

満点解法も同様のDPによる。
ΣΣ・・・Σf (上式を簡単のためこう書く) の値を考えるために、まず、x_iの次数がc_iである単項式Πx_i^{ci}に対しΣをとった値がどうなるかを考えると、$\Sigma\Sigma\cdots\Sigma\ (\Pi x_i^{ci}) = \Pi\ (\Sigma x_i^{ci})$になる(Πはi=1～N)。(例: $\Sigma\Sigma(x^2y) = (\Sigma x^2) * (\Sigma y)$) つまり、各変数ごとにわけて足しあわせてから積をとったものと同じになる。これは実際に積をとった式を展開すると元の単項式のΣの各項が出てくることからわかる。
すると結局、部分点のDPの式でx_iでk次消費した時にかけるべき係数を、x_i^kから、$A_i^k + (A_i+1)^k + \cdots + B_i^k$ に変えれば満点の場合でも答えが求まる。事前に、自然数の0~C乗の、$A_i.B_i$の上限(Xとおく)である400までの累積和を計算しておけばこ

の係数は O(1)で得られるので、この前計算が O(CX)で、DP が $O(NC^2)$で、結局 $O(CX + NC^2)$で答えが得られる。

F – バイナリハック / Unhappy Hacking

純粋な全探索を行うとしたら、探索中に保持する状態は「現在までにキーを押した回数」と「現在エディタに表示されている文字列」の組になるでしょう。しかし、「現在の文字列」を完全に保持する必要はなく、「文字列の長さ」と「そのうち何文字目まで s と一致するか」という 2 つの整数値の組を持てば十分です。この全探索アルゴリズムを動的計画法に変換することで時間計算量 $O(N^3)$ で問題が解け、部分点を得られます。

満点を得るには、状態の数をさらに減らす必要があります。s について考えると、s の中のそれぞれの文字が '0' であるか '1' であるかは問題の答えに関係しないことがわかります（N 回のキータイピングと最終的な文字列の関係を考えると、文字列の中のそれぞれの文字はどれか 1 回のキータイピングと結びついているからです）。すなわち、s の長さを M とおくと、'0', '1' のみからなる長さ M の文字列は 2^M 通り存在しますが、N 回キーを押してそれらの文字列のうちどれを得る方法の数も等しいです。

したがって、N 回キーを押して「何らかの」長さ M の文字列を得る方法の数を求めれば、それを 2^M で割ったものが s を得る方法の数、すなわち元の問題の答えとなります。部分点解法で保持する必要があった「現在の文字列の何文字目まで s と一致するか」という値が不要になり、時間計算量 $O(N^2)$ で問題を解くことができます。

なお、「mod 1000000007 の世界で数を 2 で割る」(以下、P = 1000000007 とします) には、$2a \equiv 1 \pmod P$ を満たす整数 a = 500000004（これを P を法とする 2 の逆元と呼びます）を掛ければよいです。一般に整数 n の逆元を求めるには、例えば n^{P-2} を繰り返し二乗法で求める方法があります（フェルマーの小定理より、$n \times n^{P-2} = n^{P-1} \equiv 1 \pmod P$ となります）。