Maciej Jaśkowski
Treść zadania, Opracowanie

Szymon Acedański
Program

OI, Etap III, dzień drugi, 30.03.2006

Misie

Bajtocka firma 0101010 produkuje zabawki dla dzieci. 0101010 jest bardzo znaną firmą, a ich zabawki mają opinię bardzo solidnych. Pracownicy firmy z przerażeniem stwierdzili, że ostatnie cztery modele misiów: A1, A2, B1 i B2 mają ukrytą wadę: jeśli weźmiemy trzy misie, które wszystkie mają tę samą literę w oznaczeniu modelu, lub wszystkie mają tę samą cyfrę w oznaczeniu modelu i ustawimy je obok siebie w rzędzie, to misie ulegną nieodwracalnemu uszkodzeniu.

Ustawienie misiów w rzędzie nazwiemy bezpiecznym, jeśli w jego wyniku żaden miś nie ulegnie uszkodzeniu, tzn. żadne trzy kolejne misie nie będą wszystkie miały tej samej litery w oznaczeniu modelu, ani tej samej cyfry.

Bajtazar ma kolekcję misiów, w której znajdują się tylko feralne modele. Bajtazar bawi się misiami ustawiając je w rzędzie. Zastanawia się, ile jest możliwych bezpiecznych ustawień misiów. Napisz program, który pomoże mu to ustalić.

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia liczbę misiów każdego modelu,*
- *obliczy liczbę bezpiecznych ustawień misiów w rzędzie, modulo 1 000 000,*
- *wypisze wynik na standardowe wyjście.*

Wejście

W pierwszym i jedynym wierszu wejścia znajdują się cztery nieujemne liczby całkowite: n_{A1}, n_{A2}, n_{B1}, n_{B2}, *oddzielone pojedynczymi odstępami* ($0 \leqslant n_{A1}, n_{A2}, n_{B1}, n_{B2} \leqslant 38$). *Oznaczają one liczbę misiów, odpowiednio modelu A1, A2, B1 i B2. Możesz założyć, że sumaryczna liczba misiów jest dodatnia.*

Wyjście

W pierwszym i jedynym wierszu wyjścia Twój program powinien wypisać liczbę dobrych ustawień misiów w rzędzie modulo 1 000 000.

Przykład

Dla danych wejściowych:
```
0 1 2 1
```
poprawnym wynikiem jest:
```
6
```

Istnieje 6 poprawnych ustawień misiów: B1 A2 B1 B2, B1 A2 B2 B1, B2 A2 B1 B1, B2 B1 A2 B1, B1 B2 A2 B1 oraz B1 B1 A2 B2.

Rozwiązanie

Wprowadzenie

Pytanie przedstawione w zadaniu jest modyfikacją problemu, który spotyka się w ... badaniach psychologicznych. Na przykład człowiek mający reagować w doświadczeniu na bodziec L naciskając przycisk lewą ręką, zaś na bodziec P — prawą, ma tendencję do wykorzystywania pozornych regularności do upraszczania sobie zadania — gdy trzy razy z rzędu powtórzy się bodziec L, to za czwartym razem najpewniej również zareaguje lewą ręką, nawet jeśli pojawi się bodziec P.

Psychologowie zwykle chcą unikać takich reakcji i stąd wzięło się zadanie. Oryginalny problem polegał na wygenerowaniu losowego ciągu spełniającego założenia takie, jak w zadaniu. Czytelnik bez trudu zmodyfikuje rozwiązanie wzorcowe tak, aby generowało żądany ciąg.

Oznaczenia i spostrzeżenia

Niech $T = \{A1, A2, B1, B1\}$.

Definicja 1 Uporządkowaną trójkę misiów $(p,q,r) \in T^3$ nazwiemy *trójką bezpieczną*, jeśli ustawienie tych misiów w rzędzie jest bezpieczne.

1. Niech U będzie zbiorem trójek bezpiecznych.
2. Niech $M(a_1,a_2,b_1,b_2)$ oznacza liczbę bezpiecznych ustawień a_1 misiów modelu $A1$, a_2 misiów modelu $A2$ itd.
3. Niech $N(a_1,a_2,b_1,b_2,p,q)$ oznacza liczbę bezpiecznych ustawień misiów takich, że przedostatni i ostatni miś w rzędzie to odpowiednio modele p oraz q ($p,q \in T$).

Odnotujmy jeszcze następujący fakt, prawdziwy dla układów, w których $a_1 + a_2 + b_1 + b_2 \geqslant 2$.

Fakt 1 $M(a_1,a_2,b_1,b_2) = \sum_{(p,q)\in T\times T} N(a_1,a_2,b_1,b_2,p,q)$

Rozwiązanie wzorcowe

Skorzystamy z faktu 1. Zamiast liczyć od razu wartości funkcji M, wyznaczymy najpierw, korzystając z metody programowania dynamicznego, wartości funkcji N.
Uczyńmy kluczowe spostrzeżenie.

Spostrzeżenie 2 *Dla* $a_1 + a_2 + b_1 + b_2 > 2$ *zachodzą równości:*

$$
\begin{aligned}
N(a_1,a_2,b_1,b_2,q,A1) &= \sum_{\{p:(p,q,A1)\in U\}} N(a_1-1,a_2,b_1,b_2,p,q),\\
N(a_1,a_2,b_1,b_2,q,A2) &= \sum_{\{p:(p,q,A2)\in U\}} N(a_1,a_2-1,b_1,b_2,p,q),
\end{aligned}
$$

$$N(a_1,a_2,b_1,b_2,q,B1) = \sum_{\{p:(p,q,B1)\in U\}} N(a_1,a_2,b_1-1,b_2,p,q),$$
$$N(a_1,a_2,b_1,b_2,q,B2) = \sum_{\{p:(p,q,B2)\in U\}} N(a_1,a_2,b_1,b_2-1,p,q),$$

Pisząc w skrócie:

$N(a_1,a_2,b_1,b_2,q,r) =$
$= \sum_{\{p:(p,q,r)\in U\}} N(a_1-[r=A1],a_2-[r=A2],b_1-[r=B1],b_2-[r=B2],p,q),$
gdzie $[r = x]$ *dla* $x \in T$*, to odpowiedź na pytanie, czy* $r = x$*, czyli jedynka, gdy* $r = x$*, a zero w przeciwnym przypadku. W ogólności zapis [wyrażenie logiczne], nazywany notacją Iversona, jest zdefiniowany jako* 1*, jeżeli wyrażenie jest prawdziwe, a* 0 *w przeciwnym przypadku.*

Zauważmy, że aby policzyć $N(a_1,a_2,b_1,b_2,q,r)$, dla dowolnego r wystarczy znać wartości $N(a'_1,a'_2,b'_1,b'_2,s,t)$, gdzie s i t są dowolnymi modelami misiów, zaś $a'_1+a'_2+b'_1+b'_2 = a_1+a_2+b_1+b_2-1$. Wartości N, które musimy wyznaczyć bezpośrednio, to przypadki, gdy mamy tylko dwa misie: $a_1+a_2+b_1+b_2 = 2$ — można je policzyć „na palcach” (na przykład $N(1,1,0,0,A1,A2) = 1$). Dla wartości $a_1+a_2+b_1+b_2 < 2$ funkcję $N(a_1,a_2,b_1,b_2,*,*)$ definiujemy jako równą zeru.

Implementacja

Powyższe spostrzeżenia pozwalają nam zapisać następującą funkcję.

```
function licz_ustawienia(a1, a2, b1, b2)
  { a1, a2, b1, b2 to odpowiednio liczby misiów modelu A1, A2, B1, B2 }
  if a1 + a2 + b1 + b2 ⩽ 1 then
    return 1;
  else begin
    { zainicjuj wartości N dla a1 + a2 + b1 + b2 = 2 }
    { pozostałe pola N wyzeruj }
    for i := 0 to a1 do
      for j := 0 to a2 do
        for k := 0 to b1 do
          for l := 0 to b2 do
            for q ∈ T do
              for r ∈ T do
                for p ∈ T do
                  if i + j + k + l > 2 and (p,q,r) ∈ U then
                    { Dodawanie wykonywane modulo 1000000 }
                    N(i,j,k,l,q,r) := N(i,j,k,l,q,r)+
                    +N(i−[r=A1], j−[r=A2], k−[r=B1], l−[r=B2], p, q);
  end;
  {  M — wynik  }
  M := 0;
  for p ∈ T do
```

```
23:       for q ∈ T do
24:         { Dodawanie wykonywane modulo 1000000 }
25:         M := M + N(a1, a2, b1, b2, p, q);
26:   return M;
```

Rzut oka na wiersze 8-11 pozwala przekonać się, że algorytm działa w czasie $O(a_1 a_2 b_1 b_2)$. Jest jednak jeszcze jeden problem: program zużywa za dużo pamięci! W tak zaimplementowanej procedurze tablica N zajmuje niemal $160MB$ pamięci, a mamy tylko $32MB$.

Można temu jednak zaradzić. Zauważmy, że po policzeniu $N(a_1+1, a_2, b_1, b_2, p, q)$ już nigdy nie skorzystamy z wartości $N(a_1, a_2, b_1, b_2, p, q)$. W szczególności więc możemy tak zaimplementować procedurę `licz_ustawienia`, żeby tablica N miała rozmiar $2 \times 39 \times 39 \times 39 \times 4 \times 4$. Szczegóły implementacyjne pozostawiamy Czytelnikowi jako nietrudne ćwiczenie.

Testy

Rozwiązania zawodników sprawdzane były na zestawie 13 testów, podzielonych na 10 grup. Większość testów została wygenerowana w sposób losowy.

Nazwa	a_1	a_2	b_1	b_2	Opis
mis1a.in	2	5	3	7	
mis1b.in	0	0	1	0	przypadek brzegowy
mis2a.in	7	1	8	9	
mis2b.in	5	2	1	1	wynikiem jest 0
mis3.in	6	6	4	3	
mis4.in	9	8	7	6	
mis5.in	15	17	8	21	
mis6.in	28	17	4	33	
mis7.in	38	0	1	37	
mis8.in	20	35	20	37	
mis9a.in	38	32	28	30	
mis9b.in	35	20	10	11	wynikiem jest 0
mis10.in	37	37	30	38	

XII Bałtycka Olimpiada Informatyczna,

Heinola, Finlandia 2006