Solucions del segon concurs classificatori

Olimpíada Informàtica Catalana 2019

Problema 1: Mínim nombre de monedes

Aquest problema es pot o bé pensar o bé fer un petit backtracking per trobar totes les possibles solucions, com podeu trobar a `pb1.cpp`. Per pensar-lo, cal primer veure que resoldre el problema és equivalent a resoldre el cas fins a 9 cèntims, ja que les monedes a partir de 10 cèntims són les mateixes multiplicades per 10: si podem sumar qualsevol desena i qualsevol unitat, tenim el problema resolt, ja que intuïtivament es veu que no té sentit sumar les desenes amb monedes de menys de 10 cèntims.

Amb menys de 4 monedes no és pot sumar qualsevol quantitat fins a 9, però amb 4 tenim dues solucions: 1-1-2-5 i 1-2-2-5. Escollint tal i com s'indica, la solució és 1-1-2-5-10-10-20-50.

Problema 2: Palíndroms bessons

Vegem com trobar el més gran de mida 6. Ha de ser de la forma $abccba$ i necessàriament cal fer arribar elròssec o *carry* a l'última a. Això implica que a la b de l'esquerra també li ha d'arribar *carry* i b ha de ser 9. Per ara tenim $a9cc9a$.

La c de l'esquerra més l'1 que se suma més un possible *carry* ha de ser com a mínim 10 per propagar *carry* a $b = 9$, per tant $c \geq 8$. Com que tractem de maximitzar, posem $c = 9$ i mirem si hi ha solució: ens queda $a9999a$. Finalment, veiem que per qualsevol $a \neq 9$ el nombre resultant és un palíndrom, de forma que la solució buscada és 899998.

Problema 3: Seqüència misteriosa

Cal adonar-se que com la seqüència només depèn dels dos nombres anteriors i aquests són menors que 100, la seqüència ciclarà i el cicle pot tenir com a molt longitud 10^4, tants com parells de nombres diferents menors de 100 hi ha. Podem doncs fer un petit programa que vagi calculant els termes fins a trobar el cicle, com es fa a `pb3.cpp`. Amb això trobarem que els termes x_4 i x_5 es repeteixen al cap d'un cicle de mida 60. Com que $10^{10} \mod 60 = 40$, només cal mirar x_{40}, que val 36.

Problema 4: Planetes

En primer lloc cal trobar les mides de la imatge, que seran la suma i el màxim dels diàmetres. Després només cal anar dibuixant centrats els diferents planetes, anant amb cura de no tenir errors als índexs. Com a anècdota, un dels exemples és realment el sistema solar. Reviseu la solució oficial a `pb4.py`.

Problema 5: Triangles rectangles

El problema en sí no presenta un enunciat complicat i tota la dificultat del problema acaba sent determinar si un triangle és rectangle o no, i aquí molta gent va tenir problemes de precisió. Comparar nombres reals amb `==` és normalment buscar-se problemes i cal evitar-ho en general.

En aquest problema en particular, els reals apareixen en fer arrels quadrades quan s'aplica Pitàgores, que és potser la manera més simple de saber si el triangle és rectangle. Però si mirem $a^2 = b^2 + c^2$ (sense fer cap arrel), tots els nombres són enters i no tindrem errors de precisió. Podeu veure una implementació a `pb5.py`.

Problema 6: Camí dins d'una graella

El problema només requereix fer la simulació que descriu l'enunciat per després pintar el camí del color pertinent. Tot i així, cal tenir cura dels índexs a l'hora d'aplicar l'escalat que es demana. Trobareu a `pb6.py` una possible solució.

Problema 7: Funció de Siracusa

Problema força senzill, només cal simular la funció que es demana. Podeu trobar solucions a `pb7.cpp` i a `pb7.py`.

Problema 8: Samarretes de l'OIcat

L'única dificultat del problema és calcular el nombre de samarretes. Per una talla determinada, si es van comprar c i van quedar r, el nombre que caldrà comprar és $\max(0, c - 2r)$. A `pb8.cpp` i `pb8.py` trobareu possibles solucions.

Problema 9: Seqüència Thue-Morse (2)

En primer lloc cal veure que a cada iteració la seqüència té com a longitud una potència de dos: 1, 2, 4, 8, etc. A més, si la seqüència té longitud 2^k, l'últim tros afegit té longitud $2^{\frac{k}{2}}$. La idea de la solució serà anar a la posició demanada i mirar quants salts fa fins arribar al zero original.

Busquem doncs la primera potència de dos que inclogui la posició demanada. Llavors, mentre la posició demanada no sigui la primera (0), mirem a quina de les dues meitats de la potència en qüestió cau. Si cau a la segona, comptem un canvi al dígit i anem a la posició que ha originat aquesta posició a la primera meitat. Si cau a la primera, dividim per dos la potència (descartem la segona meitat) i tornem a començar. Tot això es pot fer recursiu com a `pb9.cpp` o iteratiu com a `pb9.py`.

Alternativament, el problema es pot resoldre en temps constant mirant la representació binària del nombre: la resposta és el nombre d'uns en representació binària de $n - 1$ mòdul 2.

Problema 10: Àlbum de cromos

Per poder resoldre eficientment el problema cal anar afegint els nombres en ordre a una estructura ordenada tipus `std::set` i per cada nou nombre cal mirar els dos més propers (més petit i més gran) i sumar la diferència al que quedi més a prop. La solució oficial és a `pb10.cpp`.

Problema 11: L'alpinista maniàtic

Aquest problema es resol fent servir programació dinàmica. Sigui p_i la màxima suma que es pot aconseguir amb les i primeres muntanyes si l'últim cim escalat és parell. De forma equivalent, definim s_i pel cas senar. Així tenim que si l'alçada i-èsima x_i és parella, $p_i = \max(p_{i-1}, s_{i-1} + x_i)$ i $s_i = s_{i-1}$, i de forma equivalent definim el cas per x_i senar. Podeu trobar les solucions oficials a `pb11.cpp` i `pb11.py`.

Problema 12: UNO

No hi ha massa comentaris a fer d'aquest problema. És pesat de picar i és molt fàcil cometre errors implementant tots els detalls de les regles que s'expliquen a l'enunciat, però no té grans idees al darrere. Podeu trobar solucions a `pb12.cpp` i `pb12.py`.