Solucions del primer concurs classificatori

Olimpíada Informàtica Catalana 2018

Problema 1: Sumes i restes

Totes les preguntes són el mateix problema però canviant els números. Sigui x el primer nombre i y el segon. Cada cas ens diu $x + y$ i $x - y$, anomenem-los a i b respectivament. Això ens deixa el següent sistema:

$$x + y = a$$
$$x - y = b$$

Aïllant s'obté que $x = \frac{a+b}{2}$ i $y = \frac{a-b}{2}$.

Problema 2: Mitjons

És evident que 3 no són suficients: poden ser un negre, un blanc i un gris. Però amb 4 sempre es pot ja que només hi ha 3 colors diferents, així que com a mínim tindrem dos mitjons del mateix color. Per tant la resposta és 4.

Problema 3: Nombres particulars

Solució amb programació i intuició (o potser fe)

La solució esperada per aquest problema consisteix en fer un programa que busqui aquests nombres. Podeu trobar a `pb3.cpp` un exemple que busca fins a 10^{12}. Si ho feu, es pot veure que el programa no en troba cap. Tot i que òbviament això no demostra que no hi hagi cap, s'esperava que per intuïció es determinés que la resposta correcta és zero.

Solució matemàtica

Es pot demostrar que no hi ha cap. Suposem que existeix un nombre x així. Sigui $a \in [0, 9]$ el valor de tots els dígits de x.

Pel fet de que x és un quadrat perfecte, acaba en 0, 1, 4, 5, 6 ó 9, com es pot veure fàcilment. $a = 0$ no és vàlid i per tant només queden cinc candidats per a: 1, 4, 5, 6 i 9. Considerem els diferents casos:

- $a = 9$: Com que $9 = 3^2$, serà quadrat si i només si per $a = 1$ ho és.
- $a = 6$: Com que $6 = 2 \cdot 3$, cal que 2 sigui divisor al cas $a = 1$, però no ho és.
- $a = 5$: Per ser quadrat caldrà que 5 divideixi per $a = 1$, que no passa.
- $a = 4$: Com que $4 = 2^2$, serà quadrat si i només si per $a = 1$ ho és.

- $a = 1$: Com que és un quadrat senar, es pot posar de la forma $(2n+1)^2 = 4(n^2+n)+1$. Això vol dir que 4 divideix $x - 1$, que és de la forma $111\ldots110$ i arribem a una contradicció si $x \neq 1$.

Per tant hem demostrat que no hi ha cap nombre particular.

Problema 4: Successions matemàtiques

Solució amb programació

Podem fer un senzill programa que calculi els termes de la successió en funció dels nombres que ens donen i pari al terme que ens demanen. Podeu veure una possible implementació a `pb4.cpp`.

Solució matemàtica

Per la forma de la recurrència es pot veure fàcilment que

$$a_n = a_0 + c\sum_{i=1}^{n-1} i$$

Llavors, fent servir que la suma dels n primers nombres és $\frac{n(n+1)}{2}$ obtenim que

$$a_n = a_0 + c\frac{n(n-1)}{2}$$

Problema 5: Un triangle

Problema bastant directe, només cal fer servir la crida per pintar polígons de PIL. Podeu veure una solució a `pb5.py`.

Problema 6: Dues diagonals

El problema planteja dues dificultats: cal anar calculant les coordenades dels quadrats i si la n és senar cal pintar diferent el quadrat del mig. El més senzill és oblidar-se del mig i pintar les dues diagonals per després tornar a pintar el centre si cal. Podeu trobar una implementació a `pb6.py`.

Problema 7: Paràbola

Cal dividir el problema en dues parts: trobar l'alçada de la imatge i pintar la paràbola. Per trobar l'alçada cal trobar el punt màxim i mínim de la paràbola, que ho podem fer iterant punt a punt. Després per pintar tornem a iterar punt a punt sobre l'interval donat. Tot i que ja ho diu l'enunciat, cal fer servir la divisió entera. Mireu `pb7.py` per veure com fer-ho.

Problema 8: Sense tres en ratlla

Per resoldre aquest problema cal fer un backtracking per posar els colors. Anem iterant per les caselles en ordre i cada cop que es troba un espai buit es prova a posar fitxa en ordre lexicogràfic i es continua recursivament fins a trobar una solució. A `pb8.py` teniu el codi fet amb molts comentaris per poder-lo seguir bé.

Problema 9: Quatre dígits

Podem llegir el nombre com a enter i anar obtenint els dígits fent mòdul 10 i dividint per 10. Una altra opció es llegir-lo com a `string` i accedir a les últimes quatre posicions (compte a l'hora de convertir el caràcter a enter).

Problema 10: Primers primers

Com que com a molt ens demanen els 1000 primers primers no cal ser molt eficient calculant-los. Podem doncs anar iterant i comprovant nombre a nombre si és primer fins haver trobat tants com ens demanen. Això és el que fem a `pb10.cpp`

Per veure si un nombre és primer cal només veure si algun nombre que no sigui 1 o ell mateix el divideix. Un truc per anar molt més ràpid és veure que no cal provar divisors fins al propi nombre, és suficient veure que no tingui cap divisor menor o igual a la seva arrel quadrada: si $d|n$, llavors $\frac{n}{d}|n$, però no pot ser que d i $\frac{n}{d}$ siguin més grans que l'arrel quadrada de n.

Problema 11: Canvi mínim?

Es pot fer de diverses maneres. Una manera fàcil és per casos. Les següents condicions són suficients i necessàries perquè el canvi sigui mínim:

- Com a molt una moneda de 1 cèntim.
- Com a molt dues monedes de 2 cèntims.
- Com a molt una moneda de 5 cèntims.
- Com a molt una moneda de 10 cèntims.
- Com a molt dues monedes de 20 cèntims.
- Com a molt una moneda de 50 cèntims.
- Com a molt una moneda de 1 euro.
- No es poden fer servir una moneda de 1 cèntim i dues de 2 cèntims alhora.
- No es poden fer servir una moneda de 10 cèntims i dues de 20 cèntims alhora.

A `pb11.cpp` s'implementa aquesta idea.

Una altra solució utilitza que el nostre sistema monetari està fet de manera que si agafem les monedes de més gran a més petita i agafant tantes com es pugui sense passar-se produeix el canvi mínim. Podem calcular doncs el canvi mínim així i comparar-ho amb el nombre de monedes de l'entrada.

Problema 12: Festa salvatge

Cal comptar quantes files són anagrames de la primera. Com que les files no són gaire llargues, podem ordenar cadascuna i comparar-les amb la primera. Podeu veure una possible implementació a `pb12.cpp`.

Problema 13: Intercanvis

L'algorisme greedy és correcte en aquest cas: només cal anar intercanviant cada nombre mal posat per la posició on hauria d'anar. Però hem de fer això de forma eficient i la forma més fàcil de fer-ho és dividir la permutació en cicles i per cadascun sumar la seva longitud menys 1 al total. Per exemple, la permutació `2 4 5 1 3` es divideix en els cicles (`2 4 1`)(`5 3`) i per tant el resultat és 3. A `pb13.cpp` s'implementa aquesta idea.

Problema 14: Pòker

Aquest és el problema més difícil perquè és bastant pesat i llarg de picar. No té grans idees al darrere, potser únicament que el més fàcil és assignar a cada jugar un valor enter segons les seves cartes i després comparar només aquests enters. Tot i així, és delicat de programar i fàcil deixar-se algun cas. Podeu veure la solució oficial a `pb14.cpp`