Solucions del segon concurs d'entrenament

Olimpíada Informàtica Catalana 2018

Problema 1: Carta més alta

Hi ha un total de 12 parelles que podem agafar (primer podem triar-ne 4 i després 3 més, per tant $4 \cdot 3 = 12$), però no ens importa l'ordre, per tant tenim 6 opcions diferents (cada opció de dues cartes es pot agafar de dues maneres diferents, per tant dividint entre 2 obtenim el nombre d'opcions sense considerar l'ordre). D'aquestes 6 opcions només ens valen $\{2, 5\}$ i $\{3, 5\}$, per tant la probabilitat és $\frac{2}{6} = \frac{1}{3}$.

Problema 2: Repartint caramels

Solució amb programació

La manera més ràpida i senzilla de resoldre el problema és possiblement fer un programa que iteri sobre els possibles n i comprovi les condicions de l'enunciat. Podeu mirar el codi `pb2.py` per veure una possible implementació.

Solució matemàtica

La tercera condició implica que n és de la forma $12k+1$ per cert k. Aquesta mateixa condició implica la primera, ja que el residu de dividir $12k + 1$ entre 6 és sempre 1. Per tant, podem anar iterant sobre k fins trobar un número que compleixi la segona condició.

$$\begin{array}{lll} k = 1 & \rightarrow & n = 13 \equiv 3 \mod 10 \\ k = 2 & \rightarrow & n = 25 \equiv 5 \mod 10 \\ k = 3 & \rightarrow & n = 37 \equiv 7 \mod 10 \\ k = 4 & \rightarrow & n = 49 \equiv 9 \mod 10 \end{array}$$

Problema 3: Nombre auto-descriptiu

Solució manual

La idea d'aquesta solució és anar fent proves a mà i restringint opcions per trobar els dígits. Denotarem els dígits del nombre com $a_0a_1a_2a_3a_4a_5a_6a_7a_8a_9$. Vegem un possible camí a seguir:

- La suma dels dígits ha de ser 10, ja que indiquen quantes vegades apareix cadascun d'ells:
$$a_0 + a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + a_9 = 10$$
- a_i vol dir que i apareix i vegades, per tant pel mateix motiu que abans tenim:
$$a_1 + 2a_2 + 3a_3 + 4a_4 + 5a_5 + 6a_6 + 7a_7 + 8a_8 + 9a_9 = 10$$

- $a_5, a_6, a_7, a_8, a_9 \leq 1$ i com a molt un pot ser diferent de zero per poder complir la segona condició. $a_5 = 2$ implica $a_2 \geq 1$ i totes dues coses no poden passar alhora. Els altres ells sols ja contradiuen la segona condició si són més grans de 1. Tampoc poden ser tots zero, ja que això implica $a_0 \geq 5$, que ho contradiu.

- Si algun dels a_5, a_6, a_7, a_8, a_9 val 1, això vol dir que $a_1 \geq 2$ (no pot ser 1 perquè llavors ja hi hauria dos 1s). A més, com que a_5, a_6, a_7, a_8, a_9 tindran 4 zeros i un 1 tenim que $a_0 \geq 4$. Tot això junt fa impossible satisfer la segona condició si no succeix que a_0 valgui $j \geq 5$ i $a_j = 1$.

Això ho redueix a provar 5 casos com a molt:

- $a_5 = 1$: Aleshores $a_0 = 5, a_1 \geq 2$. Com que cal posar un zero més, suposem que $a_4 = 0$ (el que ens suma més de cara a la segona restricció). Però tot i així $a_2, a_3 > 0$ i amb tot junt ja sumem més de 10 a la segona condició.

- $a_6 = 1$: Aleshores $a_0 = 6$. Com abans, mirant de sumar el mínim possible fem $a_3 = a_4 = 0$. Llavors necessàriament $a_1 = 2$ i $a_2 = 1$, que determina la nostra solució. Ja no cal provar més casos.

  Per tant, el número que buscàvem és 6210001000.

Solució amb programació

Una possible solució consisteix en iterar sobre els 10^{10} candidats i per cadascun d'ells mirar si és la solució correcta. Aquest codi serà lent i trigarà una estona en donar el resultat.

Una millor alternativa consisteix en fer un backtracking que només consideri els nombres tals que la suma dels seus dígits és 10. Aquesta optimització fa que el programa sigui immediat. Mireu `pb3.cpp` per veure una possible implementació.

Problema 4: Diana

Aquest és un problema bàsic de gràfics amb Python on cal fer el que diu l'enunciat. Podeu mirar una possible solució a `pb4.py`.

Problema 5: Funcions linials

La solució esperada consisteix en iterar sobre els punts amb un parell de bucles anidats i pintar punt a punt amb la fórmula donada. A `pb5.py` teniu un exemple de solució.

Problema 6: Reines pacífiques

La part difícil del problema és trobar una solució, pintar-la després és bastant més senzill. Per trobar una solució cal programar un backtracking que vagi provant de posar les reines columna a columna: a la primera columna provem de posar la reina a cada fila i intentem recursivament omplir la resta del tauler. Si a cada columna provem de completar sempre començant per dalt podrem garantir que la primera solució trobada serà la que demana el problema.

Cal tenir en compte recordar durant el backtracking quines files i diagonals són vàlides sense anar-ho comprovant cada cop o al final, ja que sinó el codi no serà prou ràpid. La manera

més simple es fer servir tres vectors de booleans durant el backtracking per marcar les files i diagonals (un per les diagonals d'esquerra a dreta i l'altre per les de dreta a esquerra). Fixeu-vos que les diagonals a la posició (i, j) les podeu identificar com $i + j$ i $i - j$. Podeu trobar la solució oficial a `pb6.py`.

Problema 7: Suma de quadrats

La solució esperada consisteix en anar sumant els quadrats en un bucle i imprimir el resultat després.

Tot i que no és necessària, existeix una fórmula per aquesta suma. Considerem la següent igualtat:

$$(k-1)^3 = k^3 - 3k^2 + 3k - 1$$

Movem ara els termes de costat:

$$k^3 - (k-1)^3 = 3k^2 - 3k + 1$$

Sumem ara de $k = 1$ fins a n. Observeu que el costat esquerre esdevé una suma telescòpica, és a dir, els termes es van anul·lant i queda n^3:

$$n^3 = 3\sum_{k=1}^{n} k^2 - 3\sum_{k=1}^{n} k + \sum_{k=1}^{n} 1$$

Fent servir la fórmula per la suma dels n primers nombres, obtenim això:

$$n^3 = 3\sum_{k=1}^{n} k^2 - 3\frac{n(n+1)}{2} + n$$

Simplificant:

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

Problema 8: Nombre de dígits

La solució típica consisteix en anar dividint per 10 mentre el nombre no sigui zero per anar comptant els dígits. Això no obstant falla al cas del nombre 0, que es pot tractar de forma separada. També es pot comptar sempre un dígit de més i canviar la condició de final com fem a `pb8.cpp`.

Problema 9: Màxim de cada seqüència

Problema de bucles on potser la dificultat es troba al fet de que cal llegir l'entrada fins que s'acabi, sense saber quants casos hi haurà. Per exemple, a C++ ho podeu fer utilitzant que `cin` indica si ha pogut llegir o no. Feu una ullada a `pb9.cpp` per veure la solució oficial.

Problema 10: Barres

Aquest és un problema que té una solució molt senzilla si es fa servir recursivitat. Un patró de mida n consisteix en un patró de mida $n-1$, n asteriscos i un altre patró de mida $n-1$. Aquesta definició recursiva de la sortida es pot escriure fàcilment en codi fent una funció recursiva. A `pb10.cpp` podeu veure com implementar-ho.

Problema 11: Tresors en un mapa

Aquest és un problema clàssic de recorreguts en grafs aplicat a una graella. Bàsicament volem saber si hi ha un camí des de la casella inicial a una que tingui un tresor. Les dues maneres més estàndar de fer això són fent servir cerca en amplada (Breadth-first search) o cerca en profunditat (Depth-first search).

Explicarem la solució fent servir BFS, tot i que DFS també seria correcte i podeu veure la implementació oficial amb DFS a `pb11.cpp`. BFS utilitza una cua com a estructura auxiliar, on una cua és una estructura tipus FIFO (First In First Out, el primer que entra és el primer en sortir). Inicialment fiquem la casella inicial a la cua i la marquem com a vista. Llavors, mentre hi hagi elements a la cua, treiem el primer, explorem les caselles adjacents i si no estan vistes, les marquem i les fiquem a la cua.

Un detall important de BFS (que DFS no satisfà) és que visitem sempre les caselles en ordre de distància respecte l'origen per com funciona la cua. Per aquest problema en particular no és important, però en altres casos sí que ho és.