Solucions Final OICat 2021

Olimpíada Informàtica Catalana

Problema C1: Parells alternats

Autor: Víctor Martín

És fàcil veure que la suma dels n primers valors és $n+1$ si n és senar, i que és $-n$ si n és parell.

Repte: Demostreu-ho rigorosament per inducció.

Problema Q1: Nombres peculiars

Autor: Izan Beltrán

Per resoldre aquest problema es pot implementar les funcions de suma i producte de dígits, $s(n)$ i $p(n)$, respectivament. A continuació, anem mirant els nombres a partir d'1. Si es compleix $n = s(n)+p(n)$, acumulem la suma i escrivim el nombre per pantalla. A partir d'un cert punt, deixen d'aparèixer nombres peculiars nous.

Repte: Es pot demostrar quins nombres són peculiars sense ajuda d'un ordinador, tot i que no és fàcil. Aquí va una guia de com fer-ho: Tenint en compte quin és el dígit més significatiu, demostreu que per a tot natural n amb 3 o més dígits tenim $s(n) + p(n) < n$. A continuació, demostreu que podem descartar els d'un sol dígit, i que un nombre de dos dígits és peculiar si i només si acaba en 9.

Problema G1: Sortint de la piscina

Autor: Víctor Martín

Imaginem que el nen surt pel punt $(5n, r)$, per a un cert valor r. Com que coneixem la distància a aquest punt des del punt inicial, i la velocitat

a la piscina, podem calcular fàcilment quant triga en recòrrer aquest camí. Podem fer exactament el mateix per la part de gespa. D'aquí ens surt la fórmula del temps total en funció d'aquest valor r:

$$t(r) = \frac{\sqrt{(5n)^2 + r^2}}{v_p} + \frac{\sqrt{(x-5n)^2 + (y-r)^2}}{v_g}.$$

Ara, fem un bucle per comprovar quant val la funció $t(r)$ per a tots els valors vàlids r, i ens quedem amb el valor d'r que la minimitza. Finalment, fem el dibuix usant la r obtinguda.

Problema Q2: Nombres alegres (2)

Autor: Xavier Povill

Una manera de resoldre aquest problema és mirant tots els nombres a partir d'1. Quan trobem un nombre alegre (és a dir, un nombre per al qual $100n + 25$ és un quadrat perfecte, cosa que es pot comprovar usant la funció `sqrt()`), afegim $\frac{1}{n}$ a la suma. Fent això i anant escrivint per pantalla la suma que anem acumulant, podem veure que va convergint cap a 1. A continuació, demostrarem que la suma és, en efecte, 1.

Demostració: Sigui n un nombre alegre, és a dir, un nombre tal que $100n + 25 = m^2$ per a algun natural m. Reduït mòdul 2, tenim que $m^2 \equiv 1 \pmod 2$, cosa que implica que m és senar, i reduït mòdul 5, ens queda que $m^2 \equiv 0 \pmod 5$, cosa que implica que m és múltiple de 5. Per tant, podem escriure m com $m = 10k + 5$ per a algun enter $k \geq 1$.

Per tant, si n és un nombre alegre, llavors $100n + 25 = (10k+5)^2$ per a algun enter $k \geq 1$. És fàcil veure que el recíproc també és cert, és a dir, que si $k \geq 1$ és un enter, llavors $n = \frac{(10k+5)^2 - 25}{100} = k(k+1)$ és un enter alegre. D'aquí trobem que un nombre n és alegre si i només si és de la forma $n = k(k+1)$, d'on veiem que el k-èssim nombre alegre és $a_k = k(k+1)$.

Per tant, la suma que volem calcular, $\sum_{k=1}^{\infty} \frac{1}{a_k}$, la podem reescriure com $\sum_{k=1}^{\infty} \frac{1}{k(k+1)}$. Observem que $\frac{1}{k(k+1)} = \frac{1}{k} - \frac{1}{k+1}$. Per tant, la suma és

$$\sum_{k=1}^{\infty} \frac{1}{k(k+1)} = \sum_{k=1}^{\infty} \left(\frac{1}{k} - \frac{1}{k+1} \right) = \frac{1}{1} - \frac{1}{2} + \frac{1}{2} - \frac{1}{3} + \frac{1}{3} - \frac{1}{4} + \dots$$

Aquesta suma és una sèrie telescòpica! La suma dels N primers termes és $\frac{1}{1} - \frac{1}{N+1}$. Fent tendir N a infinit, ens queda que la suma total és $\boxed{1}$.

Problema C2: Ranges

Autor: Víctor Martín

Solució 1: Podem reduir tots els casos a un sol cas comú:

- $x \in$[1] range(n) si i només si $x \in \text{range}(0, n, 1)$.
- $x \in \text{range}(a, b)$ si i només si $x \in \text{range}(a, b, 1)$.
- Si $d < 0$, $x \in \text{range}(a, b, d)$ si i només si $-x \in \text{range}(-a, -b, -d)$.

Veiem que hem reduït tots els casos al cas $\text{range}(a, b, d)$ amb $d > 0$. Aquest cas encara el podem simplificar una mica més:

- Si $d > 0$, $x \in \text{range}(a, b, d)$ si i només si $x - a \in \text{range}(0, b - a, d)$.

Per tant, hem reduït tot al cas de mirar si un cert nombre y pertany a $\text{range}(0, k, d)$, amb $d > 0$. Aquest cas és molt senzill: y hi pertany si i només si $0 \leq y < k$ i y és divisible per d. D'aquesta manera, a més, evitem treballar amb nombres negatius.

Solució 2: Podíem resoldre aquest problema preguntant-li al propi `range` de Python si el nombre hi pertanyia o no :|

Problema Q3: La llista

Autor: Víctor Martín

Posem tots els nombres des d'1 fins a 10^6 en una priority queue de C++, i a cada pas traiem el nombre més petit x, traiem el nombre més petit que quedi a continuació y, i hi afegim $2x + y$.

Cal anar amb compte, ja que per defecte en una priority queue només podem mirar i eliminar l'element més gran. Per evitar això, podem posar els nombres amb el signe canviat, o podem usar una min priority queue, tal i com podeu veure al GitHub.

El problema també es podia solucionar de forma similar fent servir maps o multisets de C++.

[1] El símbol $\in$ vol dir "pertany a".

Problema C3: He perdut el joc

Autor: Xavier Povill

En aquest joc hi haurà posicions perdedores i guanyadores, i cal que mirem si la posició inicial és guanyadora o no. Una posició és perdedora si i només si és impossible moure'ns a una posició perdedora (això inclou les posicions terminals). Una posició és guanyadora si i només si ens podem moure a una de perdedora, ja que movent-nos-hi farem que l'altre jugador hagi de moure des d'aquesta posició perdedora.

Solució 1 (Solució parcial): Com que $a_i < i$, l'última posició sempre serà terminal i, per tant, perdedora. Inicialment, marquem totes les posicions com a perdedores. Ara, amb un bucle que vagi des de $i = n - 1$ fins a $i = 1$, a cada pas, si la posició i és perdedora, marquem a_i com a guanyadora. Finalment, escrivim la resposta en funció del que sigui la posició 0.

Solució 2 (DFS/Dinàmica recursiva): Primer, construïm un vector de vectors T, on $T[i]$ és la llista d'enters j tals que $a_j = i$ (és a dir, les posicions a les que ens podem moure des de la posició i). Començant al 0, i aprofitant que el joc no té cicles, podem fer una funció que ens digui si una posició i és guanyadora o no, mirant recursivament si alguna de les posicions j de $T[i]$ és perdedora. Si alguna ho és, retornem `true`. Si cap ho és (potser perquè i sigui una posició terminal) retornem `false`. Observem que aquesta solució també es pot veure com un DFS, on T representa la llista d'adjacència del graf els vèrtexs del qual són les posicions del joc, i les arestes del qual representen les transicions entre posicions.

Solució 3 (Ordenació topològica): Primer, creem i omplim un vector outdegree[i] amb el grau de sortida de cada posició i (el nombre d'elements j tals que $a_j = i$). Ara, usem una cua Q amb les posicions que sabem que són guanyadores o perdedores. Inicialment, afegim només les posicions terminals (és a dir, amb outdegree[i] $= 0$) a Q, ja que sabem que són perdedores, i marquem (de moment) totes les posicions com a perdedores.

Per a cada posició j que traiem de Q, si és perdedora, marquem $i = a_j$ com a guanyadora. Cada cop que fem això disminuim en 1 outdegree[i]. Quan ens quedi outdegree[i] $= 0$, ja sabem si i és guanyadora o no, i l'afegim a Q. Sabrem quin és el resultat final quan posem el 0 a la cua.

Problema G3: L'oracle capritxós

Autor: Xavier Povill

Si l'enter x que afegim a C fos senar, la suma del conjunt[2] C' resultant tindria suma senar, així que no el podríem separar en dos conjunts amb suma igual. Per tant, l'Oracle sempre respondria `No`, independentment de C.

Sigui $x = 2y$ el nombre parell que haurem d'afegir. Fixem-nos que si $C = \{500000 + y, 500000 - y\}$, l'Oracle respondrà sempre `Sí`, ja que podrà separar C' en $A = \{500000 + y\}$ i $B = \{500000 - y, 2y\}$, que tindran la mateixa suma. Però, si $500000 + y \neq 2021$ i $500000 - y \neq 2021$, l'Oracle estaria responent incorrectament.

Ens queden per tant dos casos a considerar. $500000 + y = 2021$ ens dona $x = 2y = 2 \cdot (2021 - 500000) = -995958$. Podem ignorar aquest cas ja que la x demanada ha de ser positiva.

El cas restant $x = 2y = 2 \cdot (500000 - 2021) = \boxed{995958}$ ha de ser la nostra solució. Veiem-ho: Volem demostrar que l'Oracle respon `Sí` si i només si hi ha un subconjunt amb suma 2021.

Demostrem primer que si hi ha un subconjunt A amb suma 2021, l'Oracle respon `Sí`. Sigui $B = C \setminus A$ la resta d'elements del conjunt. Llavors, els conjunts B i $A \cup \{995958\}$ tenen igual suma ($10^6 - 2021 = 997979$), així que l'Oracle respon `Sí`, com volíem veure.

Demostrem ara la segona implicació. Si l'Oracle respon `Sí`, és perquè C' es pot dividir en dos subconjunts A' i B' amb la mateixa suma. Si A', B' i C' sumen respectivament a', b' i c', tenim $a' = b' = c'/2 = (10^6 + 995958)/2 = 10^6 - 2021$. Suposem sense pèrdua de generalitat que A' conté el 995958 que s'ha afegit. El subconjunt A té suma $a' - 995958 = 10^6 - 2021 - 995958 = 2021$, com volíem demostrar.

Problema C4: Quasipalíndroms

Autor: Félix Moreno

Sigui n la longitud de la paraula s. Direm que un parell $(s[i], s[n - i - 1])$ és bo o dolent si els seus dos elements són iguals, o si són o diferents, respectivament. Sigui d el nombre de parells dolents. Una paraula s és un palíndrom si i només si $d = 0$. Caracteritzem ara els quasipalíndroms:

[2] Tècnicament, no estem parlant de conjunts, sinó de multiconjunts, perquè podem tenir elements repetits.

Si $d = 0$, s ja és un palíndrom, i intercanviant les dues posicions dels extrems ens queda la mateixa paraula. Per tant, els palíndroms també són quasipalíndroms.

Si $d \geq 3$, s no pot ser un quasipalíndrom, ja que intercanviant dos caràcters només podem canviar l'estat de com a molt dos parells, així que com a mínim ens quedaran $d - 2 > 0$ parells dolents.

Si $d = 2$, siguin (a, b) i (c, d) els parells dolents. Ens cal que $a = c$ i $b = d$ (intercanviem b i c), o $a = d$ i $b = c$ (intercanviem b i d).

Només ens queda veure el cas $d = 1$. Sigui (a, b) el parell dolent. L'única opció que tenim aquí és intercanviar o bé a o bé b pel caràcter central c d's, cosa que només podem fer si n és senar i $b = c$ o $a = c$.

Problema C5: Distància màxima

Autor: Izan Beltrán

Aquest problema es basa en una idea molt senzilla: La distància màxima entre tots els parells de posicions (a, b) i (c, d) amb $M[a][b] \neq M[c][d]$ només es pot assolir quan almenys una d'aquestes dues posicions és a una cantonada!

Demostrem-ho per reducció a l'absurd. Suposem que ni (a, b) ni (c, d) estan a una cantonada. Si reflexéssim horitzontalment o vertical la matriu, la resposta no canviaria, així que podem assumir sense pèrdua de generalitat que $a \leq c$ i $b \leq d$. Siguin $x = M[a][b]$ i $y = M[c][d]$. La posició $(0, 0)$ està més lluny de (c, d) que (a, b). Si $M[0][0] \neq y$, estariem contradint que la distància màxima s'assoleix entre (a, b) i (c, d). Per tant, $M[0][0] = y$. Per un argument similar, també obtenim $M[n - 1][m - 1] = x$. Ara bé, $M[0][0]$ i $M[n - 1][m - 1]$ tenen valors diferents i les seves posicions estan més allunyades entre si que (a, b) i (c, d). Això és una contradicció!

Per implementar una solució, guardem la matriu, i després comparem cada element amb les quatre cantonades, actualitzant a cada pas la resposta.

Problema G2: La Formiga de Langton

Autor: Víctor Martín

Simplement, guardem en una matriu de quin color està pintada cada casella. A cada pas, pintem la casella del color que correspongui i actualitzem la posició i direcció de la formiga.

La formiga de Langton pot crear uns dissenys espectaculars. En aquest video https://www.youtube.com/watch?v=1X-gtr4pEBU en podreu veure alguns. Per crear-los, en tindreu prou amb el codi solució del problema :)

Problema C6: Codificació irreversible (2)

Autor: Víctor Martín

El resoldrem amb una programació dinàmica. Sigui $dp[k]$ la resposta del problema si agaféssim els primers k dígits. Tenim $dp[0] = 1$ (la codificació buida només pot venir d'una paraula buida) i també $dp[1] = 1$ (el primer dígit serà un nombre entre 1 i 9). Inicialitzem la resta de valors $dp[k]$ a zero. Sigui s la codificació, de longitud n. Ara, des de $i = 2$ fins a $i = n$, si $s[i-1]$ és un digit vàlid (està entre 1 i 9), sumem $dp[i-1]$ a $dp[i]$ (a una codificació vàlida amb els primers $i-1$ digits, l'hi afegim un dígit vàlid més, que correspon a una lletra entre 'A' i 'I'). Si el número format pels dígits $s[i-2]$ i $s[i-1]$ correspon a alguna lletra (un nombre entre 10 i 26 es correspon a una lletra entre 'J' i 'Z'), sumem $dp[i-2]$ a $dp[i]$. Finalment, la resposta estarà a $dp[n]$. I recordeu d'anar aplicant el mòdul!

Problema G3: El petit teorema de Fermat

Autor: Félix Moreno

Generem els a^p possibles dissenys amb un backtracking. Per a cadascun, el descartem si és monocromàtic o si no és el més petit en ordre lexicogràfic, comparant-lo amb els altres $p-1$ dissenys cíclicament equivalents. Al GitHub hi veureu una manera còmoda de fer-ho, fent servir les opcions que ofereix Python. Finalment, dibuixem tots els dissenys que no haguem descartat.

Repte: El problema és una "guia" per demostrar el Petit Teorema de Fermat, però no n'és una demostració. Aquí van algunes idees: La relació d'equivalència cíclica és una relació d'equivalència. Investigueu una mica què significa això. Bàsicament, el que cal demostrar és hi ha $a^p - a$ coloracions no monocromàtiques, i que el conjunt d'aquestes es pot separar en $\frac{a^p-a}{p}$ classes d'equivalència respecte la relació d'equivalència cíclica.

Problema C7: Bipartició oculta

Autor: Xavier Povill

Segurament és el problema més difícil del concurs. Hem de pintar els vèrtexs d'un graf de vermell o de blau, de manera que almenys la meitat de les arestes siguin *bones* (connectin vèrtexs de colors diferents).

Solució 1: Des del vèrtex $i = 0$ fins el vèrtex $i = n - 1$, sabent quants dels veïns j de i amb $j < i$ (que ja estaran pintats) són vermells, i quants són blaus, pintem i del color menys freqüent. Així, a cada pas, almenys la meitat de les arestes considerades seran bones. En acabar aquest procés haurem tractat cada aresta exactament un cop i, per tant, el nombre d'arestes bones serà almenys la meitat. Finalment, escrivim totes les arestes bones.

Solució 2: Pintem cada vèrtex aleatòriament i de manera independent. Amb probabilitat $\frac{1}{2}$ el pintem de blau, i amb probabilitat $\frac{1}{2}$ el pintem de vermell. Amb probabilitat prou gran, hi haurà almenys $\frac{m}{2}$ arestes bones. Les comptem i, si no fos el cas, ho tornem a intentar fins que ho aconseguim. Demostrar que aquesta probabilitat és alta és difícil, però podem demostrar que aquest procés farà que, en promig, la meitat d'arestes siguin bones:

Sigui E el conjunt de les m arestes, i e qualsevol aresta d'E. Sigui X_e la variable aleatòria que val 0 si l'aresta és dolenta, i 1 si és bona. L'esperança (és a dir, el valor mitjà) d'X_e és $\mathbb{E}[X_e] = \frac{1}{2}$. Sigui Y la variable aleatòria que compta quantes arestes són bones, és a dir, $Y = \sum_{e \in E} X_e$. Per linearitat de l'esperança, $\mathbb{E}[Y] = \sum_{e \in E} \mathbb{E}[X_e] = \sum_{e \in E} \frac{1}{2} = \frac{m}{2}$. Això ens diu que en mitjana, la meitat de les arestes seran bones.

Comentari: Com que hi ha d'haver alguna coloració amb un nombre d'arestes bones no inferior a la mitjana, amb això també demostrem que sempre existeix almenys una solució al problema.

Problema G4: Blitspin

Autor: Félix Moreno

Podem resoldre aquest problema recursivament. A cada pas, copiem els quatre subquadrats (element per element), i els assignem a la seva nova posició (element per element). Si encara ens queda alguna iteració, apliquem l'algorisme recursivament a cadascun dels subquadrats.

Al GitHub en podeu veure una possible implementació. A més, us hem deixat un codi extra a `blitspin.py`, amb el que podreu aplicar l'algorisme

Blitspin a les vostres imatges. Podeu aprofitar per jugar amb el codi tant com volgueu i per investigar sobre eines com `numpy` o sobre les eines que ofereix la llibreria `PIL`.