

Solucions classificatori OICat 2020

Olimpíada Informàtica Catalana

Problema C1: Capses de supplements.

Autor: Edgar Moreno

Fixem-nos que per poder fer l'enviament de m mascaretes en q caixes amb capacitat a amb les restriccions indicades a l'enunciat cal que es compleixin dues condicions. La primera, que no ens sobrin mascaretes, és a dir, que la capacitat total de les caixes sigui major o igual que el nombre de mascaretes. Matemàticament: $m \leq q \cdot a$. La segona, que totes les caixes han d'estar plenes, i per tant a ha de dividir m .

Si podem posar les mascaretes a la caixa A, i els gels a la B, o viceversa, podrem fer l'enviament. I vigileu els overflows!

Problema C2: Velociraptor afamat.

Autor: Alex Alvarez

Expliquem dues maneres raonables de resoldre aquest problema. La primera, ordenem els nombres de l'entrada, i a continuació els recorrem de petit a gran comptant quants cops es van repetint els pesos. Per fer això ens cal guardar, d'una banda, la millor solució provisional, i de l'altra, el pes que estem mirant i el nombre de repeticions que portem.

L'altra manera, encara més senzilla, consisteix a crear un vector v de 151 elements, i cada cop que trobem un nou pes p , sumem 1 a $v[p]$. Finalment, mirem quin pes s'ha repetit més i escrivim la resposta.

Problema Q1: Tothom informat.

Autor: Max Balsells

Sigui $dp[i][j]$ la gent informada al dia i que ha informat durant j dies. Comencem amb $dp[0][0] = 1, dp[0][1] = 0, dp[0][2] = 0$, i a partir d'aquí, seguint el que diu l'enunciat, tenim que $dp[i][0] = 5(dp[i-1][0] + dp[i-1][1])$, $dp[i][1] = dp[i-1][0]$, i $dp[i][2] = dp[i-1][1] + dp[i-1][2]$, si $i > 0$. La nostra resposta serà el primer dia d pel qual $dp[d][0] + dp[d][1] + dp[d][2]$ passi del valor de població donat.

Problema C3: Rajoles en fila.

Autora: Maria Prat

Us explicarem **quatre** solucions, de pitjor a millor. Anomenarem $f(i)$ al cost de pintar les i primeres caselles de blanc, i les $n - i$ darreres de verd. Per tractar l'altre cas (primer verd i després blanc) en el pitjor dels casos en tindrem prou duplicant el codi.

Primera solució: Una solució senzilla però lenta que dona puntuació parcial: Per a cada i entre 0 i n , veiem què costa pintar les i primeres rajoles de blanc i la resta de verd, mirant totes les posicions. Com que la solució passa per fer $n + 1$ iteracions, i com que cadascuna mira n elements, tenim una solució amb complexitat $\mathcal{O}(n^2)$.

Segona solució: Podem convertir aquesta solució lenta en una solució ràpida amb una petita optimització. Podem observar que pintar els primers i elements de blanc i la resta de verd és exactament el mateix que pintar els $i + 1$ de blanc i la resta en verd, **exceptuant** una sola posició. D'aquí podeu extreure que:

$$f(i + 1) = \begin{cases} f(i) - 1 & \text{si la } i\text{-èsima rajola és blanca} \\ f(i) + 1 & \text{si la } i\text{-èsima rajola és verda} \end{cases}$$

Per tant, calculem $f(0)$ mirant les n rajoles i a continuació calculem els n valors $f(1), \dots, f(n)$ fent un sol càlcul per a cadascun d'ells. Tenim així una solució amb complexitat $\mathcal{O}(n)$.

Tercera solució: Podem veure que $f(0) = |B|$, on $|B|$ és el nombre de rajoles blanques. Observem que volem calcular $\min_{0 \leq i \leq n} f(i)$. Bàsicament anem a repetir la solució anterior, però ara evitant tant el primer pas de recórrer tota la fila de rajoles, com guardar informació en vectors. Fixeu-vos que tot i no saber $f(0)$, per a cada i podem obtenir l'increment acumulat, diguem-li $a(i)$ (és a dir, $f(i) = f(0) + a(i)$). En acabar de llegir-ho tot, si

hem comptat el nombre total $|B|$ de caselles blanques, tenim que la solució final serà $|B| + \min_{0 \leq i \leq n} a(i)$, ja que $|B| = f(0)$. Aquest mínim el podem actualitzar a cada iteració.

Quarta solució/Repte: Podem solucionar, a partir de la solució anterior, l'altra meitat del problema alhora. Sigui $g(i)$ el cost de pintar les primeres i caselles de verd i les $n - i$ finals de blanc. Demostreu que $f(i) + g(i) = n$ per tota i , i d'aquí traieu la solució sabent n , $|B|$, $\min_{0 \leq i \leq n} a(i)$ i $\max_{0 \leq i \leq n} a(i)$. Podeu veure la solució a `C3_sense_vectors.cc`

Problema G1: **Serp.**

Autor: Víctor Martín

Res massa complicat, però cal anar amb cura amb la implementació, especialment per decidir si la serp “baixa” per la dreta o per l'esquerra.

Problema C4: **Mininim.**

Autor: Víctor Martín

Amb casos petits es pot veure fàcilment que guanya el primer jugador si les dues piles són iguals, i el segon si són diferents. Demostrem-ho: Fixem-nos que l'últim moviment consisteix a igualar les dues piles. Fixem-nos que si les dues piles estan igualades (i no estan buides), el següent moviment per força les desigualarà. I fixem-nos que si no estan igualades, podem igualar-les traient algunes cartes de la pila gran.

Per tant, si les piles no comencen igualades, el primer jugador les pot igualar. Així, el segon jugador no tindrà cap més opció que desigualar-les, tornant així a la situació del principi. Com que el nombre de moviments és finit (com a molt se'n faran $a + b$), i l'últim moviment consisteix a igualar, aquest últim moviment el farà el primer jugador, així que aquest serà el guanyador. Finalment, si al principi les dues piles estiguessin igualades, el primer jugador hauria de desigualar-les per força, “regalant” així l'estratègia guanyadora al segon jugador.

Problema Q2: **Primers monòtons.**

Autor: Max Balsells

Per una banda, tenim els primers d'un sol dígit, i per l'altra, els que en tenen més. Per a aquest segon cas, si el dígit és diferent d'1, clarament no és primer (per exemple, $7777 = 1111 \times 7$). Si només té uns, veiem que no pot tenir-ne un nombre parell (perquè seria divisible per 11), cosa que descarta el cas amb 20 dígit. La resta els mirem un per un usant long longs. Fer els còmputus us pot portar perfectament dos minuts si useu Python.

Problema G2: Traductor Braille.

Autor: Víctor Martín

El millor que es pot fer en aquesta mena de problemes és parar-se a pensar com implementar-ho tot de la manera més senzilla possible. Fent això, el problema es torna raonablement senzill. L'única cosa comparativament "difícil" és passar de lletra a codi Braille. Una bona manera pot ser usant un map, evitant així una funció amb un munt d'ifs, i codificant cada lletra amb un string de sis 0's i 1's.

Problema C5: Alfils.

Autor: Xavier Povill

Veiem que si $n \geq 2$, la resposta usa $2n - 2$ alfils. Clarament no en pot usar $2n$ o més, perquè el tauler té $2n - 1$ diagonals, i pel **Principi del colomar** una de les diagonals tindria almenys dos alfils que s'estarien amenaçant.

Pel mateix principi, si en poséssim $2n - 1$, n'hauríem de posar un a cada diagonal. Però hi ha dues diagonals amb una sola casella. Resulta que estan en cantonades oposades, cosa que implicaria que aquests dos alfils s'estarien amenaçant entre ells.

Finalment, podem posar-ne $2n - 2$. Per exemple, ocupant completament, exceptuant la columna dreta, les files inferior i superior.

Problema Q3: Primers prefixats.

Autor: Jordi Rodríguez

Podem solucionar-ho amb un backtracking. Comencem amb els primers d'un sol dígit (és a dir, 2, 3, 5, 7), i a partir d'aquí, comprovem a cada pas si el nombre que tenim és primer, i si ho és intentem afegir un dígit al final. Per

exemple, com 31 és primer prefixat, provaríem amb 310, 311, ..., 319 (tot i que afegir un dígit parell o un 5 clarament ens portarà a un nombre compost, així que els podem descartar). Finalment, a cada primer que trobem, actualitzem la nostra solució, quedant-nos amb el més gran.

Problema C6: Menjant xocolata.

Autor: Félix Moreno

En primer lloc, observem que permutar els multiplicadors de les files per un costat, i/o el de les columnes per l'altre, no canvia res. Per tant, podem ordenar els valors dels multiplicadors de les files per un costat, i el de les columnes per l'altre.

Ara, en comptes de mirar per files i columnes, mirem per caselles. Cada casella pot ser eliminada o bé eliminant la fila corresponent, o bé eliminant la columna corresponent. Voldríem intentar per tant, de manera intuïtiva, eliminar cada casella amb la direcció que tingui el multiplicador més gran.

Aquí ve la part interessant: Si ajuntem files i columnes, i agafem la que tingui el multiplicador més gran, podem eliminar totes les seves caselles de manera òptima sense afectar la resta. Per tant, la solució òptima passarà per eliminar-la i a continuació “començar de nou” sense aquesta.

Ara bé, com que ho tenim tot ordenat de gran a petit, no cal fer cap recalculament, només cal repetir el procediment anterior, ara ignorant les files i columnes ja eliminades, fins que la rajola de xocolata quedi buida.

Problema G3: Aire Pur.

Autor: Víctor Martín

Podem solucionar aquest problema amb un algorisme de cerca sobre grafs. La solució més senzilla és un DFS. Cal que comencem per la casella amb la ‘D’ i, a cada iteració, per a cada casella adjacent a la casella on estem, cal observar si s’hi pot anar (és a dir, cal veure que la casella és dins del mapa, no és d’aigua, no ha estat ja visitada, i no és de ciutat si la casella actual és de bosc) i explorar-la si es pot. Finalment, cal dibuixar cada casella en funció del seu tipus i de si ha estat visitada o no.