Algorísmia i Programació 1, GCED, 17 de gener de 2019

Possibles solutions

1 Pupurri

1. El veredicte *Accepted* del Jutge indica que la solució proposada passa tots els jocs de proves públics i privats fent un ús raonable del temps i la memòria.
2. L'*assert* permet indicar en el codi que una condició hauria de ser certa. En temps d'execució, si les assercions estan activades, el programa s'aturarà amb un missatge d'error si en algun cas no ho és.
3. Donats dos vectors ordenats, l'algorisme de fusió retorna la unió ordenada dels seus elements.
4. No cal compilar els programes en Python.
   (Hi ha moltes altres respostes correctes.)
5. Els programes en C++ solen ser molt més ràpids.
   (Hi ha moltes altres respostes correctes.)
6. Els enters de C++ tenen un nombre fixe de bits (tipicament 32) i, per tant, pateixen sobreiximents. Els enters de Python poden ser arbitrariament llargs.
7. Amb `-Wall` el compilador proporciona més diagnòstics d'avisos que, sovint, són errors.
8. La indentació transmet visualment l'estructura dels programes, fet que facilita la seva lectura i comprensió als humans.
9. Idealment, una funció només calcula un resultat a partir d'uns paràmetres d'entrada. Una acció genera un efecte, com ara escriure o llegir dades, llançar míssils, o modificar paràmetres de sortida o entrada-sortida.
10. C++: `2 2 0`
    Python: `3 4 2`

2 Garbell d'Eratòstenes

1. *assert*($n \geq 1$)
2. Donat un enter $n \geq 1$, la funció *eratostenes* retorna un vector p de $n + 1$ booleans tal que $p[i]$ indica si i és primer o no.
3. El $p[i]$ == *true* es podria escriure com a $p[i]$ (perquè $p[i]$ ja és un booleà).
4. L'invariant conté aquests tres elements:
   - $p[0] = p[1] =$ *false*.
   - $i \geq 2$.
   - Per a tot j tal que $2 \leq j \leq n$, $p[j]$ val cert si i només si j no es pot dividir per cap nombre entre 2 i $i - 1$.
5. Al final de la funció, els valors $p[j]$ són correctes per aquestes raons:

- Per a $j \in \{0, 1\}$, la primera clàusula de l'invariant ho assegura i 0 i 1 no són primers per definició.
- Per a j tal que $2 \leq j \leq \sqrt{n}$, la tercera clàusula de l'invariant ho assegura perquè, quan el bucle acaba, $i > \sqrt{n}$.
- Per a j tal que $\sqrt{n} < j \leq n$, la tercera clàusula de l'invariant ho implica perquè, quan el bucle acaba, $i > \sqrt{n}$ i perquè j no pot tenir cap divisor entre $1 + \sqrt{j}$ i $j - 1$.

3 Generació combinatòria

1. Funció *solucions*:

   def *solucions* (n, ℓ, u, a, s, x, k, *sum_chosen*, *sum_rest*):
   if $k == n$:
   print(x)
   else:
   if *sum_chosen* + *sum_rest* − $a[k] \geq \ell$:
   $x[k] = 0$
   solucions (n, ℓ, u, a, s, x, $k + 1$, *sum_chosen*, *sum_rest* − $a[k]$)
   if *sum_chosen* + $a[k] \leq u$:
   $x[k] = 1$
   solucions (n, ℓ, u, a, s, x, $k + 1$, *sum_chosen* + $a[k]$, *sum_rest* − $a[k]$)

   Crida inicial:

   solucions (n, ℓ, u, a, s, x, 0, 0, s)

2. Donades les precondicions següents:

   - $n \geq 0$,
   - $0 \leq k \leq n$,
   - a i x ténen n posicions,
   - $0 \leq \ell \leq u \leq s = \sum_{i=0}^{n-1} a[i]$,
   - $x[i] \in \{0, 1\}$ per a tot $i < k$,
   - $sum\_chosen = \sum_{i=0}^{k-1} a[i]x[i]$,
   - $sum\_rest = \sum_{i=k}^{n-1} a[i]$,
   - $\ell \leq sum\_chosen \leq u$,

   solucions (n, ℓ, u, a, s, x, k, *sum_chosen*, *sum_rest*) escriu en ordre lexicogràfic creixent totes les solucions de la doble inequació $\ell \leq \sum_{i=0}^{n-1} a[i]x[i] \leq u$ havent fixat les primeres k posicions de x.

3. Només cal invertir l'ordre de les crides recursives: és a dir, intercanviar els dos **if**s (amb el seu codi corresponent) que hi ha dins de l'**else**.

4. Quan $\ell = 0$ i $u = s$, caldrà escriure totes les combinacions, no podent-ne espurgar cap.

5. $O(2^n)$ perquè hi ha n elements, cadascun amb dos valors possibles.

4 Maleïda cerca binària

1. Funció 1: No funciona si la taula només té una posició i un element que és més gran que el que es busca. Per exemple, quan $T = [1]$ i $x = 0$, es penja.

2. Funció 2: No funciona si la taula només té una posició i un element que és més petit que el que es busca. Per exemple, quan $T = [0]$ i $x = 1$, accedeix a una posició que no existeix. Tampoc funciona quan la taula és buida.

3. Funció 3: No funciona, per exemple, quan $T = [10, 20]$ i $x = 30$: accedeix a una posició que no existeix a la segona iteració.