Algorísmia i Programació 1, GCED, 8 de gener de 2021

Possibles solucions

1 Les preguntetes d'en Jordi

1. Un algorisme és un conjunt finit d'instruccions o passos que serveixen per a resoldre un problema.
2. Un intèrpret executa directament les instruccions escrites en un llenguatge de programació mentre que un compilador transforma les instruccions a codi màquina [o codi de més baix nivell].
3. Una precondició és una condició o predicat que ha de ser cert just abans d'executar una secció de codi per tal que aquest funcioni tal com s'ha especificat.
4. Ordenació per fusió, ordenació per inserció, ordenació per selecció, ordenació per bombolla.
5. El canvi de base d'un logaritme consisteix en multiplicar per un factor constant. Com que la notació asimptòtica ignora els factors constants, no cal precisar la base.
6. $O(n^2)$.
7. Com que l'algorisme d'ordenació per selecció és $O(n^2)$, hauria de trigar quatre cops més per treballar sobre el doble de dades. Per tant, un minut aproximadament.
8. No té sentit: L'algorisme de cerca binària hauria de fer unes 20 comparacions sobre d'un milió elements. Potser l'ordinador és extraordinàriament lent, l'algorisme està mal implementat, les dades no estan ordenades...

   Error freqüent: Dir que l'ordinador de l'Àngel és molt ràpid.
9. C++: `unicorn rainbow`
   Python: `rainbow unicorn`

2 Trobeu els errors

1. Cal utilitzar `math.pi` enlloc de l'aproximació 3'1416.
2. - La variable *darrer* hauria de ser de tipus `double` enlloc de tipus `int`. [De fet, segurament millor no tenir aquesta variable.]
   - Les condicions de l'`and` s'haurien de posar del revés, per evitar l'accés a la posició -1 del vector.
3. - La k de $a[i][k]$ hauria de ser una j.
   - El bucle de les j hauria d'anar fins a $m-1$.
   - Quan la matriu a és buida, a[0].*size*() provoca un error.
4. Aquesta funció no du a terme l'efecte desitjat: els paràmetres reals no canvien. Caldria retornar una nova tripleta i recollir-ne el resultat.
5. - El vector té un 1000000 posicions, entre 0 i 999999, però s'accedeix a la 1000000.
   - L'ús del valor màgic 1000000 arreu demostra que caldria haver definit alguna constant.

- La funció *erastotenes* no fa tota la feina que li pertoca perquè relega la inicialització del vector a qui la crida.
- Els $v[i]$ == *true* es poden escriure $v[i]$ perquè $v[i]$ ja és un booleà.

3 Trobeu els errors

1. Donat un natural n, *mystery1*(n) retorna el factorial de n.
2. Donat un natural n, *mystery2*(n) retorna l'n-èsim nombre triangular, és a dir, $\sum_{i=1}^{n} i = n(n+1)/2$.
3. Donat un natural n, *mystery3*(n) retorna $n+1$.
4. Donat un natural n, *mystery4*(n) retorna una aproximació de π utilitzant el mètode Montecarlo tot usant n punts aleatoris.

Error freqüent: No esmentar que n és un natural com a precondició.

4 El pot de bales

[Es sobreentén que i, b, r i n són naturals.]

(a) Invariant:

- p indica si b és parell,
- $b + r = i$, i
- $i \geq 1$.

(b) Primerament, demostrem que l'invariant és cert al començar al bucle: p s'ha inicialitzat a la paritat del nombre de bales blaves al pot, i s'ha inicialitzat a n (que és $b + r$), i $i \geq 1$ per la precondició $n \geq 1$.

Segonament, demostrem que l'invariant es manté després d'una iteració del cos del bucle:

Quan s'entra dins del bucle es té que $i > 1$ (perquè la condició del bucle ha de ser certa), que $b + r = i$ i que p indica si b és parell. Hi ha tres casos:

- Quan les dues bales extretes són diferents, r es decrementa d'una unitat però b no canvia (ja que se'n treu una però se n'afageix una altra). Com que i també es decrementa, es manté que $b + r = i$. I com que b i p no han canviat, p continua indicant si b és parell.
- Quan les dues bales extretes són blaves, r s'incrementa d'una unitat i b es decrementa de dues. Com que la i es decrementa d'una unitat, es manté que $b + r = i$. Com que b s'ha decrementat de dues unitats, p continua indicant si b és parell.
- Quan les dues bales extretes són rojes, r es decrementa d'una unitat (se'n treuen dues i se n'afageix una) i b no canvia. Com que la i es decrementa d'una unitat, es manté que $b + r = i$. I p continua indicant si b és parell.

Als tres casos, el valor final d'i és més gran o igual que 1, perquè el valor original era estrictament més gran que 1 i s'ha decrementat d'una unitat.

(c) Quan el bucle acaba sabem que es compleix l'invariant i la negació de la condició del bucle. Per tant, quan el bucle acaba, $b + r = i = 1$ i p indica si b és parell.

Si el nombre inicial de bales blaves al pot és parell, llavors p serà cert i per tant b serà parell. Llavors $b = 0$ i $r = 1$, tal com cal demostrar.

Si el nombre inicial de bales blaves al pot és senar, llavors p serà fals i per tant b serà senar, per tant $b = 1$ i $r = 0$, tal com cal demostrar.

5 Quatre en ratlla

using *Fila* = *vector*<**int**>;

using *Tauler* = *vector*<*Fila*>;

struct *Partida* {
    int *cols*;
    vector<*string*> *noms*;
    int *torn*;
    Tauler tauler;
};

L'estat del tauler es representa amb la matriu *Tauler*, que tindrà c columnes i tantes files com calgui (començant de zero). Les fitxes de cada jugador es codifiquen a través del seu índex, i es guarda el valor especial -1 per a les posicions buides.

Una partida és una estructura que guarda el nombre de columnes al tauler, els noms dels jugadors (per identificar el guanyador al final de la partida), el torn actual (per saber a qui li toca jugar), i la situació actual del tauler. Valors com el jugador actual o el nombre de jugadors es poden deduir de les representades.

Al final de la partida d'exemple es trindria:

```
cols:  3
noms:  {"Anna", "Berta", "Carla"}
torn:  11
tauler:  {
    { 2,  2, 0,  0,  0,  0},
    {-1, -1, 1,  1, -1, -1},
    {-1, -1, 1, -1, -1, -1},
}
```

Partida crear_partida ();
Llegeix el nombre i noms de jugadors, llegeix el nombre de columnes i retorna una partida amb aquests valors i un tauler buit, inicialitzat al primer torn. El format de l'entrada és el descrit a l'enunciat.

bool *acabada* (**const** *Partida*& *partida*);
Indica si la *partida* ja està acabada (si algun jugador ha fet quatre en ratlla).

string guanyador (**const** *Partida*& *partida*);

Donada una *partida* que ja està acabada, retorna el nom del seu guanyador.

void *jugar* (*Partida*& *partida*, **int** *pos*);
Actualitza una *partida* que no està acabada per tal de reflectir que el jugador del torn actual tira la seva fitxa a la columa *pos*. Si *pos* no és legal, la *partida* no canvia excepte pel torn, que sí queda incrementat.

void *escriure* (**const** *Partida*& *partida*);
Escriu una *partida*, el format de la sortida és el descrit a l'enunciat.