Berezvai Dániel	1. beadandó/4. feladat	2012. április 13.
BEDTACI.ELTE		Programozás
3ice@3ice.hu		11. csoport

Feladat

Madarak életének kutatásával foglalkozó szakemberek n különböző településen m különböző madárfaj előfordulását tanulmányozzák. Egy adott időszakban megszámolták, hogy az egyes településen egy madárfajnak hány egyedével találkoztak. Melyik az a madárfaj, amelyik a leggyakrabban (azaz a legtöbb településen) fordult elő?

Bemenő adatok

A program billentyűzetről vagy fájlból is tud bemeneti adatokat fogadni. Kimenetét a konzolra ($stdout$) írja, melyet fájlba irányíthatunk vagy megtekinthetünk a képernyőn.

A bemeneti fájlt paraméterként kell megadni.
Ha nincs paraméter, a program automatikusan billentyűzetről kezd olvasni.

	Település 1	Település 2		Település n		Települések száma
Madárfaj 1	5	2		0		2
Madárfaj 2	0	0	...	2		1
...				...		...
Madárfaj m	9	7		3		3

Bemenő adatfájlok elvárt formája

$n\ m$
$\{n$ db madárfaj$\}$
$\{m$ db településnév$\}$
$\{n \times m$ méretű mátrix$\}$

A szöveges állomány formátumáról feltételezzük, hogy helyes.

Például (bemenet/pelda.txt)

2 3
Feketerigó
Veréb
Budapest
Esztergom
Pécs
400 300 200
5000 4000 4000

Specifikáció

→Feltételes maximumkeresésbe ágyazott számlálás.

Azért nem az egyszerűbb maximumkiválasztás tételt választottam, mert ha minden madár előfordulása 0 minden településen, akkor nem szeretnék megoldást visszaadni. Inkább kiírnám, hogy nincs elég madár.

Feltételes maximumkeresés

$f(i) \sim e = elofordul(t[i])$

$\beta(k) \sim e > 0$

$m..n \sim 1..n$

$A\colon (t\colon \mathbb{N}^{n \times m}\ l\colon \mathbb{L}, ind\colon \mathbb{N}, max\colon H, e\colon \mathbb{N})$

$Ef\colon (t = t')$

$$Uf\colon \begin{pmatrix} Ef\ \wedge (l = \exists k \in [1..n]\colon t[i] > 0) \wedge \\ (l \rightarrow ind \in [1..n] \wedge t[ind] > 0 \wedge max = t[ind] \wedge (\forall i \in [1..n]\colon t[i] > 0 \rightarrow max \geq t[i])) \end{pmatrix}$$

<table>
<tr><td colspan="4">$l := hamis$</td></tr>
<tr><td colspan="4">$i = 1..n$</td></tr>
<tr><td colspan="4">$e = elofordul(t[i])$</td></tr>
<tr><td>$e \leq 0$</td><td colspan="2">$e > 0 \wedge l$</td><td>$e > 0 \wedge \neg l$</td></tr>
<tr><td rowspan="2">$-$</td><td colspan="2">$e > max$</td><td rowspan="2">$l, max, ind = igaz, e, i$</td></tr>
<tr><td>$max, ind := e, i$</td><td>$-$</td></tr>
</table>

A nem megengedett $e = elofordul(t[i])$ értékadást kitranszformáljuk:

Számlálás

$m..n \sim 1..n$

$i \sim j$

$f(i) \sim t[j]$

$\beta(i) \sim t[j] > 0$

$A\colon (t\colon \mathbb{Z}^n, c\colon \mathbb{Z})$

$Ef\colon (t = t')$

$$Uf\colon \left(Ef\ \wedge\ c = \sum_{\substack{j=m \\ t[i]>0}}^{n} 1 \right)$$

<table>
<tr><td colspan="3">$elofordul(t[i])$</td></tr>
<tr><td colspan="3">$c, j = 0, 1$</td></tr>
<tr><td colspan="3">$j \leq n$</td></tr>
<tr><td colspan="3">$t[j] > 0$</td></tr>
<tr><td>$c := c + 1$</td><td>$-$</td></tr>
<tr><td colspan="3">$j : j + 1$</td></tr>
</table>

Implementáció

Adattípusok megvalósítása

Kódoláskor a t vektort $vector\ \langle vector\langle int\rangle\ \rangle\ t$-ként deklaráljuk, amelynek méretei $t.size()$ és $t[0].size()$ alakban érhetők el.

A $Madarak$ és a $Települések$ vektorokban tároljuk a neveket. (Lehetne tömbben is.)

Programozási trükkök / szükségletek

C++-ban a vektorokat 0-tól indexeljük, ezért a tervbeli ciklus nem az $[1..n]$, hanem a $[0..n)$ jobb oldalon nyílt intervallumot futja be.

A hármas (l, max, ind) értékadást a következőképpen oldottam meg. Először bevezettem egy $lmaxind$ nevű osztályt:

```
class lmaxind{
public:
        bool l;
        int max;
        int ind;
};
```

Majd a feltételes maximumkeresés végén létrehoztam ennek egy objektumát és feltöltöttem a visszatérési értékekkel:

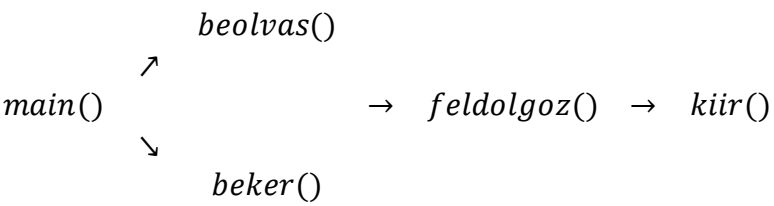
```
lmaxind l_max_ind;
l_max_ind.l = l;
l_max_ind.max = max;
l_max_ind.ind = ind;
```

Végül visszatértem vele:

```
return l_max_ind;
```

Függvények kapcsolódási szerkezete

Az előző beadandómhoz képest most sokkal egyszerűbb függvényszerkezettel dolgoztam:



Tesztelési terv

A feladat specifikációjára épülő (fekete doboz) tesztesetek:

1. Feltételes maximumkeresés, Számlálás, és Intervallumaik:
    a. Üres fájl (ures.txt)
    b. Egy-egy érték (egyertek.txt)
    c. Minden 0 (minden0.txt)
    d. Utolsó a max (utolsomax.txt)
    e. Első a max (elsomax.txt)
    f. Első 0 (elso0.txt)
    g. Utolsó 0 (utolso0.txt)
2. Különleges értékek esetei (Billentyűzetről tesztelendő):
    a. Üres sztring, mint érték (Figyelmen kívül hagyja, tovább vár a bemenetre, mintha misem történt volna.)
    b. Negatív számok (Hibaüzenetet ír ki.)
    c. Betűt tartalmazó értékek esete (Figyelmen kívül hagyja a betűt.)
    d. Speciális karakterek esete (Hibaüzenetet ír ki)

A megoldó programra épülő (fehér doboz) tesztesetek:

1. Hibás vagy nem létező állománynév megadása. (Hasznos hibaüzenetet ír ki.)
2. Paraméterként megadott fájlnév. (Az elvárt módon működik.)
3. Ismételt futtatás kipróbálása. (Az elvárt módon működik.)
4. Olyan állomány olvasása, ahol az utolsó sort nem zárja sorvége jel. (pelda.txt)
5. Stressz-teszt, 1.000.000 értékkel. (stressz_teszt.txt), (!stressz-teszt.bat)

A $feldolgoz()$ és az $elofordul()$ függvények tesztelése

1. Minden 0 (minden0.txt)
2. Első 0 (elso0.txt)
3. Utolsó 0 (utolso0.txt)
4. Első a max (elsomax.txt)
5. Utolsó a max (utolsomax.txt)
6. Ezen kívül még lehetne tesztelni például a darabszámok növelésszámára (0, 1, több)

Minden teszteset (kivéve stressz teszt) lefuttatása egyszerre: !teszt.bat

Még paraméter tesztet is raktam bele. (Nem létező fájl.123)

Stressz teszt

Előszeretettel generálok hatalmas tesztfájlokat és csodálkozom rá a számítógépek többmagos processzorainak számítási sebességére újra és újra. Ha majd CUDA programozással foglalkozzunk, még gyorsabb lesz minden: 20 billió helyett 1000 billió számítás/másodperc.

Az $1.000.000$ elemű, 3001 soros, 1000 oszlopos $3.6MB$ méretű stressz-tesztesetet a mellékelt BEDTACI.ZIP/bemenet/Stressz teszt.xlsx Excel fájl létrehozásával és txt-ként mentésével készítettem. (A tabulátorokat lecseréltem szóközre, ami 100% CPU kihasználtság mellett is eltartott 2 percig.) A használt formula: $= ABS\big(FLOOR(RAND( ) * 1000 - RAND( ) * 1000; 100)\big)$

Annak reményében bonyolítottam két véletlenszerű szám különbségére az egyszerűbb, $= ABS\big(FLOOR(RAND( ) * 1000; 100)\big)$ formulát, mert így nagyobb valószínűséggel lesz 0 az eredmény, ami csökkenti a fájlméretet és segíti a tömörítést. ($7.9MB$ helyett $3.6MB$ lett, tömörítve mindössze $576KB$.) Az így kapott egymillió szám átlaga 335.0205, míg az egyszerűbb formulával kapottaké 450.2177 volt. (Miért nem 500? A lefelé kerekítés $\big(= FLOOR( )\big)$ miatt.)

Majd átneveztem néhány száz madarat Magyarországon honos fajokra (Wikipédiáról másoltam a neveket, annyi módosítással, hogy a kis kezdőbetűket átírtam nagyra)

Legutolsó lépésként pedig népszerűsítettem a verebeket: 2 kivétellel minden településre raktam belőlük, hogy biztos nyerjenek. Az eredeti nyertes "Madár704" lett volna, 938 településsel.