"Programozás"
beadandó feladat:
4. feladat

Készítette: Bárdosi Bence
Neptun-azonosító: VY9NJN
E-mail: bardosi.bence@gmail.com

2017-03-06

Tartalom

1 Dokumentáció

1.1 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. Volt-e olyan település, ahol mindegyik madárfaj előfordult?

1.2 Specifikáció

$\mathbf{A} = (adat : \mathbb{N}^{nxm}, l : \mathbb{L})$

$\mathbf{Ef} = (adat = adat')$

$$\mathbf{Uf} = \left(Ef \wedge (l, \_) = \sum_{i=1}^{n} mind(adat[i]) \right)$$

ahol mind(adat[i]) :

$\mathbf{A} = (sor : adat[i](\mathbb{N}^m), l : \mathbb{L})$

$\mathbf{Ef} = (sor = sor')$

$$\mathbf{Uf} = \left(Ef \wedge l = \forall \sum_{i=1}^{m} sor[i] > 0 \right)$$

1.3 Algoritmus

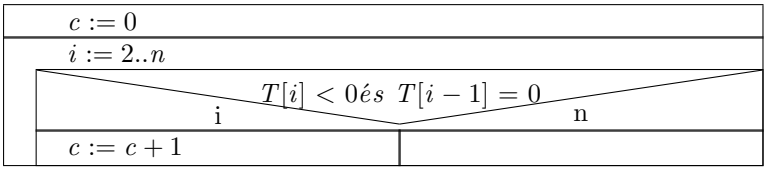
A feladatot a lineáris keresés, az alfeladatot az optimista lineáris keresés programozási tételeire vezetjük vissza.

Lineáris keresés

Tétel		Feladat
m	$\leftarrow$	1
n	$\leftarrow$	n
ind	$\leftarrow$	_
$\beta(i)$	$\leftarrow$	$\sum_{i=1}^{n} mind(adat[i])$

Optimista lineáris keresés

Tétel		Feladat
m	$\leftarrow$	1
n	$\leftarrow$	m
$\beta(i)$	$\leftarrow$	$\forall \sum_{i=1}^{m} sor[i] > 0$



1.4 Implementáció

1.4.1 Adattípusok megvalósítása

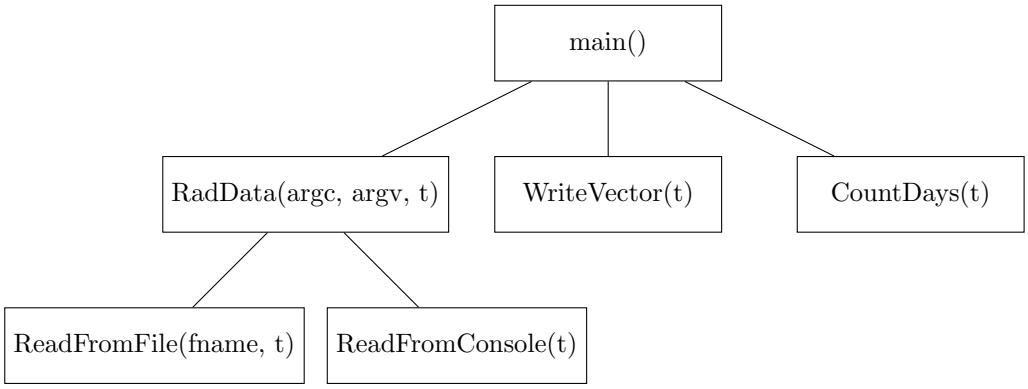
A kódoláskor a t tömböt `vector<float>`-ként deklaráljuk, amelynek mérete `t.size()` alakban érhető el. Mivel a vektor 0-tól indexelődik, azért a tervbeli ciklus nem a `2..n`, hanem az 1..n-1 intervallumot, pontosabban a `1..t.size()-1` intervallumot futja be, aminek következtében a struktogramm kódja az alábbi lesz:

```
1         int c=0;
2         for(int i=1; i<(int)t.size(); ++i)
3             if(t[i]<0 && t[i-1]==0)
4                 ++c;}
```

1.4.2 Bemenő adatok formája

A bemenő adatokat egy szöveges állományból kell a tömbbe bemásolni. Az állományban a megadott értékeket szóközökkel, tabulátor jelekkel vagy sorvége jelekkel elválasztva kell beírni. Az állomány minden sorát sorvége jel zárja le.

1.4.3 Függvények kapcsolódási szerkezete



1.5 Tesztelés

1.5.1 A feladat specifikációjára épülő (fekete doboz) tesztesetek:

Megszámlálás tétel tesztesetei:	
intervallum hossza szerint:	
1. *nulla* hosszú:	Egyetlen nap sincs be1.txt: [] - válasz: 0
2. *egy* hosszú:	Egyetlen nap be2.txt: [5.2] - válasz: 0
3. *kettő* hosszú:	Kettő, a feltételnek eleget nem tevő nap be3.txt: [5.2, 0] - válasz: 0 Kettő, a feltételnek eleget tevő nap be3.txt: [0, -2] - válasz: 1
4. *több* hosszú:	Több nap be5.txt: [0, -2, 0, 5, 6, 0, -0.5] - válasz: 2
intervallum eleje szerint:	
	Sorozat elején található csak a feltételnek megfelelő nappár be6.txt: [0, -2, 0, 5, 6, 0] - válasz: 1
intervallum vége szerint:	
	Sorozat végén található csak a feltételnek megfelelő nappár be7.txt: [0, 2, 0, 5, 6, 0, -2] - válasz: 1
tételre jellemző esetek szerint:	
1. Egyetlen megfelelő nappár van:	be7.txt: [0, 2, 0, 5, 6, 0, -2] - válasz: 1
2. Nincs megfelelő nappár:	be8.txt: [0, 1, 2, 3, 4] - válasz: 0
3. Egy ”0” napot több ”negatív” nap követ:	be9.txt: [1, 2, 0, -1, -2, -1] - válasz: 0

1.5.2 A megoldó programra épülő (fehér doboz) tesztesetek:

- Hibás vagy nem létező állománynév megadása.
- Állomány nevének megadása parancssorból.
- Olyan állomány olvasása, ahol egy sorban több érték is található egyetlen illetve több szóközzel és/vagy tabulátor jellel elválasztva (be10.txt).
- Olyan állomány olvasása, ahol minden érték külön sorban van (be9.txt).
- Olyan állomány olvasása, ahol az utolsó sort nem zárja sorvége jel, és éppen ennek a sornak a tartalma határozza meg az eredményt (adat: [1, 2, 3, 0, -1] – válasz: 1) (be11.txt).
- Főprogram ciklusának ellenőrzése: olyan bemenő adatokkal, amelyekre a ciklus egyszer sem fut le (Pl: be1.txt), pontosan egyszer fut le (Pl: be3.txt), vagy többször lefut(Pl:be5.txt).