ORSI 1. beadandó feladat - dokumentáció

Kovács Bálint - FANW4Z

November 27, 2016

1 Feladat

A bemenet `input.txt` első sorában egy N pozitív egész olvasható, ennyi diáknak kell biztosítani termet, míg a következő N sorban a hallgatók NEPTUN-kódjai, azaz N string követi egymást, ez mutatja a jelentkezési sorrendet.

Egy lehetséges bemeneti fájl:

```
7
OSAVH1
T0NDJB
4S1UPL
AXKAW4
22TQP7
NM8VPS
PJVNEU
```

A megoldás során TILOS a beépített rendezéseket használni (pl. $std :: sort()$), a feladat egy párhuzamosított MergeSort (összefésüléses rendezés) implementálása! A fő folyamat feladata az adatok beolvasása az `input.txt` fájlból, majd az output.txt kimeneti fájl létrehozása, benne a rendezett adathalmazzal. Rendezéshez kötődő számítást ne végezzen! (Összefésülést sem!)

2 Felhasználói dokumentáció

2.1 Környezet

A program futtatásához Erlang shell szükséges.

2.2 Használat

Fordítsuk $c(merge\_sort2)$ paranccsal, majd futtathatjuk $merge\_sort2 : start()$ függvényhívással. A fájl mellett kell elhelyezni az `input.txt` fájlt, melyet feldolgoz és az eredményt az `output.txt` nevű fájlba írja, a betűrend alapján. Egy lehetséges bemenetet tartalmaz a mellékelt `input.txt` tesztfájl. Saját bemeneti fájlok esetén a sorok és a játékok számának pontos megadása nem fontos, elég ha az elején kihagyunk egy sort.

Viszont a helyes működéshez szükséges a többi sor megfelelő formátuma, és hogy ne legyen több üres sor a dokumentumban.

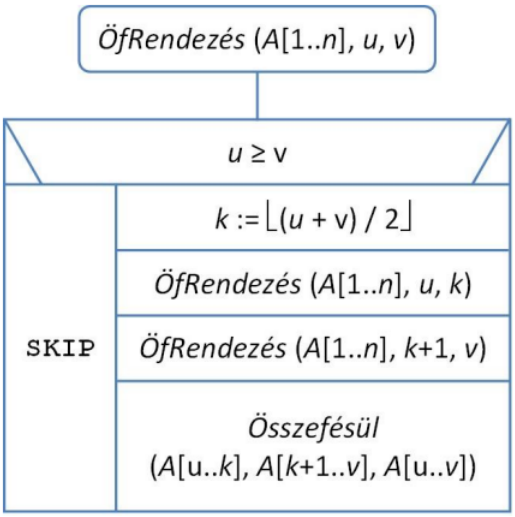
3 Fejlesztői dokumentáció

3.1 A megoldás módja

A főfolyamat beolvassa az input fájlt, majd az olvasott adatra elindítja a párhuzamosított összefésülő rendezést. Ez később buborékrendezésre vált. A rendezett listát kiírjuk a folyamat végén.

3.2 Implementáció

A bemeneti adatokkal meghívjuk a $merge\_sort$ függvényt, ami a rekurzív rendezésért felel. A beépített $spawn()$ függvény segítségével a rendezés egyik ágát párhuzamosítjuk. Az összefésülő rendezés algoritmusa, mely alapján az említett függvényt írtam (dr. Fekete István oldaláról):



A futás egy makróban megadott értéknél ($MergeLimit$) rövidebb listákra buborékrendezésre vált. Ennek az értéknek a meghatározásról a tesztelés fázisban írok. A teljes implementáció egyetlen forrásfájlba szervezve, a `merge_sort2.erl` fájlban található.

3.3 Fordítás

Fordítsunk $c(merge\_sort2)$ paranccsal, majd futtathatjuk a programot $merge\_sort2$: $start()$ függvényhívással.

3.4 Tesztelés

A programot 100000 sor hosszú fájllal teszteltem, ez volt az a méret, ahol látszódtak az eltérések a futási időben. Az eredmények, a buborék rendezésre váltást jelző szám függvényében, három futást átlagolva:

- 0 $\rightarrow$ 3,296s
- 5 $\rightarrow$ 2,867s
- 20 $\rightarrow$ 2,770s
- 40 $\rightarrow$ 2,737s
- 70 $\rightarrow$ 2,650s
- 100 $\rightarrow$ 2,794s

Így 70re állítottam az értéket. Az érték különböző rendszereken eltérő lehet.