Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozási Nyelvek és Fordítóprogramok Tanszék

Osztott rendszerek specifikációja és implementációja

IP-08bORSIG

Dokumentáció a 2. beadandóhoz

Nagy Richárd Tibor
GWSAZV

2017. november 22.

1. Kitűzött feladat

Feladatunk annak eldöntése, hogy egy adott halmaznak létezik e olyan részhalmaza, melyben található elemek összege pontosan megegyezik egy előre megadott számmal!

A szükséges adatokat a program 3 parancssori paraméteren keresztül kapja.

Az első, egy egész értékű adat, mely a feladatban definiált számot jelzi, ezt kell valamilyen módon elérni a halmazelemek összegével.

A második, egy fájl neve - ez tartalmazza a kiinduló halmaz elemeit (inputfájl). A felépítése az alábbi: A fájl első sorában egy nemnegatív egész szám (N) áll - a kiinduló halmazunk elemszáma tehát N. A következő sorban összesen N db egész számot olvashatunk (pozitív és negatív egyaránt), melyek a halmaz elemeit jelölik (sorrendiséget nem kötünk meg köztük).

Egy megfelelő bemeneti fájl (például data.txt) ekkor:

6

3 34 4 17 5 2

A harmadik paraméter, annak a fájlnak a neve, melybe a feladat megoldása során kapott választ kell írni.

A kimeneti fájlban található információ az alábbi legyen:

létezik N összegű részhalmaz:

'May the subset be with You!'

nem létezik ilyen részhalmaz:

'I find your lack of subset disturbing!'

Egyik esetben sem kell a sor végére enter (valamint az aposztrófok is csak a BEAD-os szemléltetés miatt vannak ott, azokat sem kell kiírni), a kis- és nagybetűkre figyeljetek a tesztelő miatt! A program egy lehetséges futtatása így az alábbi módon valósulhat meg:

pvm spawn -> master 12 data.txt result.txt

A kapott kimenet (result.txt) az előző bemenetre tehát a következő:

'May the subset be with you!'

(A 3, 4 és 5 elemek összege pontosan 12, így tudunk ilyen halmazt mutatni.)

A feladatot 'Divide & Conquer' módszer alapján PVM3 használatával kell megoldani, az alábbiak alapján:

Amennyiben az elvárt összeg 0, alapesetnél vagyunk, ezt ki tudjuk elégíteni az üres halmaz, mint részhalmaz segítségével (igaz).

Amennyiben az elvárt összeg nem nulla, de a halmazunk üres, erre nem tudunk megoldást adni, az eredmény hamis.

Egyéb esetben két lehetőségünk van, az elérni kívánt (összegű) halmazba a jelenleginek egy tetszőleges elemét vagy beletesszük, vagy nem.

Ha igen, akkor rekurzívan nézzük meg, hogy az a halmaz kielégíthető-e, melybe a most vizsgált elemet nem vesszük bele. Tehát pl. egy 10 összegű, K elemszámú halmaz akkor és csak akkor tartalmazza a 3-at, mint elemet, ha a 7 összegű, K-1 elemszámú halmazt ki tudjuk elégíteni (amely az itt vizsgált 3-as elemet nem tartalmazza) a feltételek alapján.

Ha nem, akkor a rekurzió azt dönti el, hogy egy kisebb elemszámú, de azonos súlyú halmazra teljesül e az elvárt feltétel. Ha az 5 összegű, K elemű halmazból ki akarjuk venni a 35-öt, mint elemet, de a vizsgált súlyt nem akarjuk változatatni, akkor vizsgáljuk meg az 5 összegű, K-1 elemű halmazt (a 35 nélkül).
Ha a rekurzió valamelyik ága igazat ad, akkor a válaszunk igaz, tehát tudunk ilyen halmazt mondani (az elemeket nem kell megadni), egyéb esetben nem.

(Pl: Szeretnék kifizetni 5000Ft-ot, és rendelkezésemre áll több címlet. Van nálam egy kétezres, dönthetek arról, hogy ezt szeretném -e használni a fizetés során vagy nem. Ha igen, akkor a kérdés az, hogy 3000-et tudok e fizetni a maradékból. Ha nem akarom használni, akkor a maradék címletből kell összetenni az 5000-ret.)

A program futásánál feltehetjük, hogy az inputfájl létezik, és a parancssori paraméterek száma és formája stimmel, valamint a fájlban található adatok is konzisztensek, ezeket nem kell külön ellenőrizni.

2. Felhasználói dokumentáció

2.1. Rendszer-követelmények, telepítés

A program futtatásához a PVM keretrendszerre van szükség, annak megléte nélkül nem használható. Külön telepítésre nincsen szükség.

2.2. A program használata

A programot taszkként indíthatjuk el PVM-ből. Három paramétert vár: az elérni kívánt halmazelem összeget, a bemeneti adatokat tartalmazó fájlt, valamint a kimeneti fájl nevét. Egy lehetséges futtatása az alábbi módon valósulhat meg:

```
pvm
spawn -> master 12 data.txt result.txt
```

A fent említett szöveges fájlokat a home könyvtárban kell elhelyezni. A bemeneti fájl formátumának helyessége nincs ellenőrizve, hibás formázás esetén az alkalmazás megfelelő működése nem garantált. A futás végeztével a paraméterként megadott kimeneti fájlban található az eredmény.

3. Fejlesztői dokumentáció

3.1. Megoldási mód

A feladat a subset sum probléma megoldása rekurzív módon PVM-mel. A párhuzamos programunk két különálló C++ programból áll (master és child). Futtatás során ezek kapcsolatba lépnek: a master fogja elindítani a child programot (ami újabb child-okat

indít) PVM-en keresztül, és üzenetet is váltanak egymással. A master végzi ezen felül a beolvasást és a kiírást, a child-ok pedig az egyes részhalmazokat vizsgálják.

3.2. Implementáció

A forráskódok mérete és komplexitása nem indokolja azok a már meglévőknél több fájlba való szétbontását, így azok teljes egészükben a `master.cpp` és a `child.cpp` nevű állományokban találhatóak. A bemenet sorait tartalmazó vektor a `std::vector<std::string>` típussal valósul meg. Az új taszkok indítását a `pvm_spawn()` függvény végzi. A taszkok közötti kommunikációt a `pvm_send()` és a `pvm_recv()` alprogramok, az üzenetek be- és kicsomagolását pedig a `pvm_pk*()` és `pvm_upk*()` mintájú függvények hajtják végre.

3.3. Fordítás menete

A fordításhoz elengedhetetlen egy `C++11` szabványt támogató fordítóprogram a rendszeren. Ehhez használhatjuk az *MSVC*, *g++* és *clang* bármelyikét. A fordításhoz célszerű létrehozni egy makefile állományt, ami automatikusan elvégzi a fordítást, ezek eredményének áthelyezését a megfelelő mappákba, a linkelést, valamint a takarítást. Fontos, hogy `-std=c++11` kapcsolóval fordítsunk, mert lehetséges hogy alapértelmezés szerint a fordítóprogram még a régi, C++98-as szabványt követi, melyben a felhasznált nyelvi elemek még nem voltak jelen.

3.4. Tesztelés

A tesztelést az ELTE Atlasz nevű számítógép klaszterén végeztem, mind a 12 blade-et kihasználva. A program aszimptotikus nagyságrendje miatt legfeljebb 12 elemű halmazok vizsgálatára alkalmas. A program tesztelését több különböző mintafájlon is elvégeztem, azok alapján a működését helyesnek ítéltem.
Az áltagos futási idők a következők:
4 elemű halmaz:

- 1 blade: 0.059024 sec
- 2 blade: 0.0522158 sec
- 4 blade: 0.03924 sec
- 8 blade: 0.0278627 sec
- 12 blade: 0.0183281 sec

12 elemű halmaz:

- 1 blade: >180 sec
- 2 blade: >60 sec
- 4 blade: 6.82119 sec
- 8 blade: 2.32274 sec
- 12 blade: 1.47436 sec