Naloga: TRE
Lov na zaklad

CEOI 2011, Dan 1. Source datoteka `tre.*` Omejitev spomina: 256 MB. *9.07.2011*

Saj ste že slišali za gusarje in njihove zaklade? Bajtko je našel staro steklenico z navodili, ki opisujejo lokacijo skritega zaklada, vendar jih je precej težko razvozlati. Zaenkrat Bajtko ve le to, da mora najti dve posebni točki v bližnjem parku, kajti zaklad leži točno na sredini poti med tema točkama.

V parku, v katerem se skriva zaklad, obstaja veliko stezic, ki neposredno povezujejo po dve sosednji točki. Vse stezice so dolge po en korak. Ozemlje zunaj stezic je gosto zaraščeno in zato nedostopno za ljudi. Stezice so postavljene na poseben način: med vsakima dvema točkama v parku poteka natanko ena pot. Posamezna pot sestoji iz ene ali več stezic, vendar med sprehodom po poti nobene točke ne obiščemo več kot enkrat.

Bajtko nima načrta parka in je zato prosil svoje prijatelje, da mu ga pomagajo raziskati. Lov na zaklad bodo začeli v neki točki v parku. Park bodo raziskovali v fazah. V vsaki fazi eden od prijateljev izbere točko, ki je bila že odkrita, in se iz nje sprehodi po poti določene dolžine, pri čemer obiskuje samo točke, ki do zdaj še niso bile odkrite.

Med raziskovanjem bo Bajtko pazljivo analiziral strukturo parka. Vsake toliko časa se mu zna zazdeti, da je uganil, kateri dve točki v parku sta tisti posebni, ki ju opisuje gusarski načrt. Takrat bo želel hitro ugotoviti, katera točka v parku leži točno na sredini edine poti med tema dvema točkama. Pomagaj mu.

Komunikacija

Napisati moraš knjižnico, ki bo komunicirala z ocenjevalnim programom. Tvoja knjižnica mora vsebovati naslednje tri funkcije, ki jih bo klical ocenjevalni program (lahko pa vsebuje seveda tudi poljubne druge funkcije):

- `init` — klicana bo natanko enkrat, na začetku izvajanja programa. V njej lahko inicializiraš podatkovne strukture in podobno.
  - C/C++: `void init();`
  - Pascal: `procedure init();`

  Privzemi, da se je v trenutku, ko je ta funkcija poklicana, raziskovanje parka ravno začelo in obstaja v parku natanko ena odkrita točka. Označimo jo s številom 1.
- `path` — klicana bo, ko prijatelji v parku odkrijejo novo pot.
  - C/C++: `void path(int a, int s);`
  - Pascal: `procedure path(a, s: longint);`

  Pot se začne v točki a (ki je bila že odkrita) in naredi s korakov ($s > 0$). Po vsakem koraku pridemo v doslej neobiskano točko in jo označimo z novim številom: najmanjšim naravnim številom, ki še ni bilo uporabljeno za označevanje točk. Ta funkcija bo poklicana vsaj enkrat.
- `dig` — klicana bo, ko Bytka zanima, kje iskati zaklad.
  - C/C++: `int dig(int a, int b);`
  - Pascal: `function dig(a, b: longint) : longint;`

  Funkcija naj vrne število, prirejeno točki, ki leži na sredini poti med točkama, označenima s številoma a in b. Vedno bo veljalo, da sta a and b že odkriti točki in da $a \neq b$. Če je pot lihe dolžine (srednja točka takrat ni dobro definirana), naj funkcija vrne tisto od obeh srednjih točk, ki je bližje točki a (glej tudi primer). Ta funkcija bo poklicana vsaj enkrat.

Tvoja knjižnica **ne sme** brati ničesar, niti s standardnega vhoda niti iz kakršnekoli datoteke; prav tako **ne sme** ničesar pisati, niti na standardni izhod niti v kakršnokoli datoteko. Knjižnica **sme** pisati na `stderr`, vendar se zavedaj, da to porablja čas, namenjen izvajanju tvojega programa.

Če pišeš v C/C++, knjižnica **ne sme** vsebovati funkcije `main`. Če pišeš v Pascalu, moraš določiti `unit` (glej primer na disku).

Prevajanje

Tvoja knjižnica — `tre.c`, `tre.cpp` ali `tre.pas` — bo prevedena skupaj z ocenjevalnim programom s sledečimi ukazi:

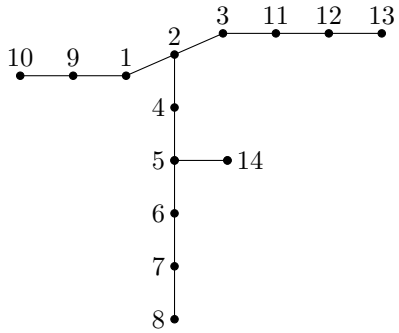
- C: `gcc -O2 -static -lm tre.c tregrader.c -o tre`

- C++: `g++ -O2 -static -lm tre.cpp tregrader.cpp -o tre`
- Pascal:
  ```
  ppc386 -O2 -XS -Xt tre.pas
  ppc386 -O2 -XS -Xt tregrader.pas
  mv tregrader tre
  ```

Primer izvajanja

Spodnja tabela prikazuje primer zaporedja klicev funkcij ter za klice funkcije `dig` še pripadajoče pravilne rezultate klicev. Struktura stezic v parku, ki ustreza temu primeru, je prikazana na sliki.

Funkcijski klic	Rezultat	Novododane točke
`init();`		1
`path(1, 2);`		2, 3
`dig(1, 3);`	2	
`path(2, 5);`		4, 5, 6, 7, 8
`dig(7, 3);`	5	
`dig(3, 7);`	4	
`path(1, 2);`		9, 10
`path(3, 3);`		11, 12, 13
`dig(10, 11);`	1	
`path(5, 1);`		14
`dig(14, 8);`	6	
`dig(2, 4);`	2	



Omejitve

- Ocenjevalni program bo izvedel največ 400 000 klicev funkcij (`init`, `path` in `dig`). Bajtko in prijatelji bodo odkrili največ 10^9 točk v parku.
- V testnih primerih, skupaj vrednih 50 točk, bo odkritih največ 400 000 točk v parku.
- V testnih primerih, skupaj vrednih 20 točk, bo odkritih največ 5 000 točk v parku, ocenjevalni program pa bo izvedel največ 5 000 klicev funkcij `init`, `path` in `dig`.

Preizkušanje

Da boš lahko preizkusil svoj program, imaš na razpolago vzorčen ocenjevalni program v datoteki

`tregrader.c`, `tregrader.cpp`, oziroma `tregrader.pas`

v direktoriju **/home/zawodnik/tre/** na tvojem računalniku. Če ga želiš uporabiti, moraš svojo rešitev zapisati v datoteko

`tre.c`, `tre.cpp`, oziroma `tre.pas`

v pripadajočem direktoriju (`c`, `cpp` ali `pas`). Na začetku tekmovanja lahko v vsaki od teh datotek najdeš vzorčno nepravilno rešitev naloge. Svojo rešitev prevedeš z ukazom

`make tre`,

ki deluje natanko tako, kot je opisano zgoraj v razdelku Prevajanje, če se pred tem postaviš v ustrezen direktorij (`c`, `cpp` ali `pas`). Prevajanje C/C++ zahteva še datoteko `treinc.h` v ustreznem direktoriju (`c` ali `cpp`).

Binarna datoteka, ki je rezultat takšnega prevajanja, sprejema na standardnem vhodu imena in argumente funkcij, pokliče ustrezne funkcije iz tvoje knjižnice in zapiše rezultate klicev funkcije `dig` na standardni izhod. Seznam funkcij na standardnem vhodu mora biti oblikovan takole: prva vrstica vsebuje število ukazov, q. Sledi q vrstic; vsaka vsebuje znak `i`, `p` ali `d`, ki mu sledita dve nenegativni celi števili. Znak določa, katera funkcija naj se kliče: `i` za `init`, `p` za `path` in `d` za `dig`. Števila predstavljajo argumente funkcij: a in s za `path`, a in b za `dig`. Pri znaku `i` naj bosta obe števili enaki 0. Bodi pozoren na to, da testni ocenjevalni program **ne** preverja, ali so vhodni podatki pravilno oblikovani, niti, ali zadoščajo zahtevam iz razdelkov Komunikacija in Omejitve zgoraj.

Dana je tudi datoteka `tre0.in`, ki predstavlja podatke za zgoraj opisani primer zagona programa:

```
12
i 0 0
p 1 2
d 1 3
p 2 5
d 7 3
d 3 7
p 1 2
p 3 3
d 10 11
p 5 1
d 14 8
d 2 4
```

Da ocenjevalnemu programu na standardni vhod podtakneš podatke iz te datoteke, uporabi ukaz

```
./tre < tre0.in
```

Rezultati klicev tvoje funkcije `dig` bodo izpisani na standardni izhod. Pravilen izpis za zgoraj vpisane vhodne podatke (zapisan tudi v datoteki `tre0.out`), je:

```
2
5
4
1
6
2
```

Da preveriš, ali so izhodni podatki tvoje knjižnice pravilni za osnoven testni primer, lahko svojo rešitev pošlješ na ocenjevalni sistem (SIO).