— grid —
Program na řešení barevných i černobílých lušťovek

Verze 1.2 (9. 11. 2003)

Mirek Olšák + Petr Olšák (`mirek@olsak.net, petr@olsak.net`)

Program řeší lušťovky následujícího typu: je dána síť prázdných čtverečků a po stranách jsou čísly označeny délky a počty bloků černé nebo i jiné barvy. U vícebarevných lušťovek je vyznačena i barva. Má se vybarvit síť tak, aby údaje po stranách odpovídaly. Podrobněji viz například:

`www.griddlers.net`

Program pracuje s černobílými i barevnými lušťovkami se čtverečkovanou sítí. Trojúhelníková políčka ve čtverečkované síti je možné také deklarovat. Program umí řešit i tzv. triddlers (tj. trojúhelníková síť s šestiúhelníkovým obvodem) v černobílé i barevné verzi.

Program umožní buď vyřešit celou lušťovku, nebo dokáže načíst částečně vyřešenou lušťovku a poradit s dalším jednotlivým tahem. Nebo Vám poradí, kde jste udělali v částečně vyřešené lušťovce chybu (ale nemusí prozradit řešení).

Instalace

Program přeložíte ze zdrojového kódu pomocí příkazu:

```
cc -O2 -o grid grid.c
strip grid
```

Parametr `-O2` podporuje GNU `gcc` (optimalizace rychlosti). Pokud Váš kompilátor tento parametr nepodporuje, nemusíte jej použít. Příkaz `strip grid` čistí binární program od informací pro debugger. Není povinný a na některých platformách asi není ani implementován.

Zdrojový text jazyka C se opírá jen o standardní knihovnu C, tj. měl by být bez problémů přeložitelný na *libovolné platformě*.

Provoz

Program spustíte pomocí:

```
grid soubor
```

kde `soubor` je jméno vstupního souboru, ve kterém jsou zaneseny údaje o počtech blocích v řádcích a ve sloupcích, tj. zadání lušťovky. Formát tohoto vstupního souboru je popsán níže.

Při řešení lušťovky programem může nastat několik případů:

• Řešení se povedlo najít v souladu se zadáním. V takovém případě program napíše OK a vypíše řešení na terminál. Navíc zapíše toto řešení v grafickém formátu do souboru `soubor.xpm`. Na řešení se můžete podívat např. Gimpem.

• Zadání je sporné. Program před tím, než začne lušťovku řešit, zkontroluje základní konzistenci zadání: zda počet políček každé barvy, sčítáme-li je podél řádků, vychází stejně, jako když je sčítáme podél sloupců. Pokud toto neplatí, program nezačne vůbec luštit a skončí s chybovou zprávou o nekonzistenci zadání.

• Zadání je sporné, zůstala nevyřešená políčka. Nevyřešená políčka jsou označena otazníky. Program napíše KO. K této situaci dochází pouze tehdy, když je zadání sporné, ale jednoduchým testem na rovnost počtu políček v řádcích a sloupcích tento spor neodhalíme. Program upozorní na číslo sporného řádku nebo sloupce.

• Existuje více řešení. Program vypíše všechna řešení na terminál a první řešení uloží do xpm souboru. Toto chování se dá pomocí parametrů příkazové řádky změnit.

Parametry příkazové řádky

`grid [parametry] soubor-se-zadáním`

`-help`
Vypíše stručnou informaci o možných parametrech na terminál a ukončí činnost.

`-p` ⟨*number*⟩ implicitní hodnota: `-p 0`
Po ⟨*number*⟩ krocích v řešení přechází program do módu, při kterém se zastavuje u každého kroku a vypisuje stávající stav řešení na terminál. Nově objevená políčka (z posledního kroku) jsou označena znaky `#` (barva) a `-` (barva není). U vícebarevných lušťovek vypíše program stav řešení po „vrstvách“ pro každou barvu zvlášť pouze při `-out 3` a více. `-p 0` znamená, že se program nezastaví nikdy, `-p 1` znamená, že se program zastaví hned po prvním kroku a zastavuje se pak při každém dalším kroku. Pojem `krok programu` je vysvětlen níže.

`-stop` ⟨*number*⟩ implicitní hodnota: `-stop 0`
Význam je stejný, jako při `-p` ⟨*number*⟩ jenom s tím rozdílem, že po vykonání ⟨*number*⟩ kroků program vypíše stav řešení a ukončí zcela svou činnost. Při `-stop 0` program končí až v okamžiku, kdy došel ke sporu nebo našel všechna řešení nebo našel aspoň `-total` řešení.

`-total` ⟨*number*⟩ implicitní hodnota: `-total 30`
Po nalezení ⟨*number*⟩ různých řešení program ukončí činnost a další řešení nehledá. Při `-total 0` program najde a vypíše všechna řešení.

`-xpm` ⟨*number*⟩ implicitní hodnota: `-xpm 1`
Do souborů `*.xpm` uloží jen prvních ⟨*number*⟩ řešení. Při `-xpm 1` program založí soubor stejného jména, jako `soubor-se-zadáním`, ale bez jeho přípony a připojí příponu `.xpm`. Při `-xpm 2` a více program připojí ke jménu souboru ⟨*číslo řešení*⟩`.xpm`, tj. může založit více souborů a uložit do nich postupně všechna řešení daného problému. ⟨*číslo řešení*⟩ je zleva doplněno nulami na tolik cifer, kolik cifer má ⟨*number*⟩. Při `-xpm 0` program nevytváří žádný XPM výstup.

`-i`
Program bude používat pouze intenzivní algoritmus a testy.

`-log` ⟨*number*⟩ implicitní hodnota: `-log 2`
⟨*number*⟩ značí úroveň ukecanosti výstupu na terminál.
Při `-log 0` vystupují na terminál pouze řešení. Tisk řešení se dá regulovat pomocí parametru -out.
Při `-log 1` se vypíše pod řešení údaj o počtu řešení a celková statistika: počet kroků, počet úspěšných/všech vstupů do řádků.
Při `-log 2` se vypisují navíc čísla a typy kroků v průběhu výpočtu.
Při `-log 3` se navíc vypisují stavy řešených řádků-sloupců před a po aplikaci příslušného algoritmu.
Při `-log 4` se navíc vypisují interní informace použitých algoritmů.

`-lf` ⟨*file*⟩ implicitní hodnota: `stdout`
Místo na terminál bude výstup programu směrován do souboru. Pokud tento soubor existuje, bude na začátku činnosti programu vymazán.

`-out` ⟨*number*⟩ implicitní hodnota: `-out 2`
Ovlivní formát výstupu řešení na terminál:
`-out 0` ... nic se netiskne.
`-out 1` ... řešení bez čísel řádků a sloupců kolem.
`-out 2` ... řešení včetně čísel řádků a sloupců kolem.
`-out 3` ... neukončená řešení (při pausách nebo při konfliktu v zadání) jsou vypisována ve všech barevných vrstvách.
`-out 4` ... neukončená řešení se tisknou v každém kroku bez ohledu na nastavení pause.

`-of` ⟨*file*⟩ implicitní hodnota: `stdout`

Místo na terminál bude tisk řešení směrován do souboru. Pokud tento soubor existuje, bude na začátku činnosti programu vymazán.

`-cmp` ⟨*soubor*⟩

⟨*soubor*⟩ obsahuje částečně vyřešenou lušťovku ve formátu, který je shodný s tím, co program vypisuje na terminál. Podrobněji o tomto formátu viz níže. Program najde první řešení, ale nevypíše ho ani neuloží do XPM. Místo toho je porovná s obsahem ⟨*souboru*⟩ a vypíše otazníky tam, kde jsou otazníky v ⟨*souboru*⟩ a vyznačí pomocí znaků `#` a `-` místa, která se v souboru liší od vypočítaného řešení. Jedná se tedy o nápovědu, co je ve Vašem řešení špatně, ale program neprozradí řešení samotné.

`-ini` ⟨*soubor*⟩

Program začíná řešit od stavu lušťovky, která je popsána v ⟨*souboru*⟩. Přitom automaticky nastaví `-stop 1`, tj. program prozradí jen další pokračování tohoto řešení po prvním kroku. Pokud nutně chcete dořešit lušťovku až do konce, pište explicitně `-stop 0`. Tento údaj musí následovat za parametrem `-ini`. Např.: `grid -ini moje -stop 0 lustovka.g`

`-bl` ⟨*number*⟩ implicitní hodnota: `-bl 7`

Na začátku řešení lušťovky program zkouší zabývat se jen těmi řádky/sloupci, které mají stejně nebo méně bloků než ⟨*number*⟩. Teprve, pokud za tohoto omezení nedostaneme další výsledky, přechází program ke všem řádkům/sloupcům bez omezení.

Parametry mohou být uvedeny v libovolném pořadí a jsou odděleny mezerou. Pokud se stejný parametr uvede vícekrát, platí jeho hodnota, která je uvedena u posledního výskytu tohoto parametru. Pokud napíšete místo jména souboru znak `-`, pak program čte data ze standardního vstupu. Je možné i třeba toto:

```
cat zadani.g inisoubor | grid -xpm 0 -log 0 -out 1 -ini - - > napoveda
```

Zbývá vysvětlit pojem **krok programu**. Jedním krokem je průchod přes všechny řádky (krok typu **r** – rows) nebo průchod přes všechny sloupce (typ **c** – columns). Tyto dva typy kroků se střídají. Při řešení jednotlivého řádku/sloupce se používá rychlý tzv. pravo-levý algoritmus, který ne vždy poskytne z řádku/sloupce všechny informace, které by byl schopen odhalit člověk. Může se stát, že ani po průchodu všemi řádky a sloupci nevznikla žádná nová změna v rozpracovaném řešení. V takovém případě program přechází do kroků typu **R** a **C**, ve kterém znovu projde řádky a sloupce, tentokrát intenzivním algoritmem. Tento algoritmus je nepatrně pomalejší, ale nevynechá nic, co by mohl odhalit člověk zaměřující se izolovaně na řádky nebo sloupce. Po nalezení změny v lušťovce v rámci kroků typu **R** a **C** se program vrací do režimu střídání kroků typu **r** a **c** až do chvíle, kdy znovu tímto způsobem nelze odhalit další změnu.

Může se stát, že ani po aplikaci kroků typu **R** a **C** není odhalena v lušťovce žádná další změna. V takovém případě program přechází do kroku typu **t** (test), při kterém navrhne zaměnit nějaký otazník barvou pozadí. Po tomto návrhu přechází program zpět ke střídání kroků typu **r** a **c** s občasným použitím kroků typu **R** a **C** až do situace, kdy najde řešení nebo zjistí, že návrh vedl ke sporu. V obou případech pak ještě zkusí za otazník navrhnout barvy bloků a znovu testuje, zda takový návrh vede k řešení nebo ke sporu.

V případě triddlers program střídá tři základní kroky: **r** – řádky, **c** – sloupce zespodu, **e** – sloupce shora. Jinak pracuje stejně, tj. při selhání těchto tří kroků přechází do kroků typu **R**, **C** a **E** a případně také do kroku typu **t**.

Formát vstupního souboru se zadáním

Pro dvoubarevné (černobílé) lušťovky stačí do souboru napsat:

```
Jakýkoli text třeba na více řádcích. Tento text bude ignorován.
V prvním sloupci ignorovaného textu nesmí být dvojtečka ani
znak vězení #. Prvním znakem souboru nesmí být číslice.
: dvojtečka na začátku řádku vymezuje zadávání řádků
```

```
... údaje o řádcích lušťovky,
    jeden řádek souboru odpovídá jednomu řádku lušťovky
: dvojtečka na začátku řádku ukončuje řádky a zahajuje zadávání sloupců
... údaje o sloupcích lušťovky,
    jeden řádek souboru odpovídá jednomu sloupci lušťovky
: dvojtečka na začátku řádku ukončuje zadání
libovolný text
```

Zadání řádků a sloupců obsahuje čísla (kladná celá) oddělená od sebe mezerou (nebo více mezerami či tabelátory). Před prvním číslem může být také libovolné množství mezer a tabelátorů. Čísla mají stejný význam, jako obvykle po stranách lušťovky. Je významný i prázdný řádek, který označuje nula bloků v daném řádku nebo sloupci lušťovky. Příklad vstupního souboru: viz `kocka.g`.

Vícebarevné lušťovky mají podobný formát vstupního souboru, jen musejí obsahovat navíc tzv. deklaraci barev. Příklad včetně detailního popisu formátu je v souborech `oko.g` a `ruze.g`

Lušťovky s trojúhelníky (podle griddlers.net, tj. trojúhelníky se nijak nevážou na celistvé bloky) mají stejný formát vstupního souboru jako vícebarevné lušťovky. Příklad včetně popisu formátu je v souboru `alladin.g`

Lušťovky s trojúhelníky, které mohou tvořit okraje celistvých bloků (podle časopisu Maľované krížovky, vydává Silentium s.r.o., Bratislava) mají podobný formát, jako vícebarevné lušťovky, jen je potřeba deklarovat pomocí znaků `<` nebo `>` trojúhelníky, které navazují na blok zleva nebo zprava. Příklad včetně popisu formátu je v souboru `brontik.g`

Triddlers se zadávají podobně, jako černobílé či barevné lušťovky s pravoúhlou sítí. Na začátku souboru (před případnou deklarací barev) je navíc potřeba pomocí `#T` nebo `#t` v prvním sloupci dát najevo, že zadání obsahuje triddlers. Místo obvyklých dvou bloků dat oddělených dvojtečkami je třeba zadat šest bloků dat oddělených dvojtečkami. První blok se vztahuje na řádky popsané vlevo nahoře, druhý na řádky vlevo dole, třetí blok obsahuje údaje sloupců začínajících úplně dole, čtvrtý pak sloupce začínající vpravo dole, pátý blok popisuje sloupce začínající vpravo nahoře a šestý blok popisuje sloupce začínající zcela nahoře. Kolem šestiúhelníku tedy čteme údaje proti směru hodinových ručiček počínaje levým horním rohem a v tomto pořadí je zapisujeme do souboru. Upozorňujeme na drobnou zradu: údaje o blocích sloupců, které začínají dole, musíme zapisovat v pořadí *zdola nahoru*. Příklady zadání triddlers jsou v souborech `tkocka.t` nebo `vcely.g`.

Program také dokáže přečíst zadání černobílých lušťovek podle formátu, který používá program `mk` (`http://frix.fri.utc.sk/~johny/mk43frm.php`). Do módu čtení takového souboru se program grid přepíná automaticky podle toho, že prvním znakem souboru je číslice. Pak přeskočí první řádek souboru, protože tam nejsou pro program grid podstatné informace, a začíná číst až údaje o řádcích lušťovky (až po symbol `#`) a dále čte údaje o sloupcích (až po prázdný řádek nebo konec souboru). Řádek s nulovým počtem bloků se v tomto formátu značí nulou. Příklad takového souboru je `levikral.mk`

Formát vstupního souboru s částečným řešením

Tyto soubory se používají v souvislosti s parametry `-ini` a `-cmp`. Formát je navržen tak, aby mu vyhovovaly výstupy programu grid na terminál:

```
libovolný text
:::: čtyři dvojtečky zahajují vlastní popis lušťovky
   : řádky lušťovky
   : počet těchto řádků se musí shodovat s počtem řádků
   : lušťovky, který je deklarován v hlavním vstupním souboru
libovolný text
```

Každý řádek lušťovky má tvar:

⟨*lib.text*⟩⟨*dvojtečka*⟩⟨*lib.znak*⟩⟨*znaky lušťovky*⟩⟨*lib.text*⟩

přitom počet znaků lušťovky na řádku musí odpovídat počtu sloupců lušťovky, který je deklarován v hlavním vstupním souboru.

Znaky lušťovky mohou být následující:

otazník nebo tečka — zatím nevyřešené políčko
mezera nebo minus — určitě barva pozadí
hvězdička nebo křížek — určitě černá
znak pro terminál podle deklarace barev — určitě barva

Soubor s částečným řešením vytvoříte snadno:

```
grid -stop 1 lustovka.g > lustovka.p
```

Pak můžete pokračovat:

```
grid -ini lustovka.p lustovka.g
```

V případě triddlers je formát souboru stejný, jen s tím rozdílem, že řádek lušťovky má tvar:

⟨*lib.text*⟩⟨*lomitko*⟩⟨*lib.znak*⟩⟨*znaky lušťovky*⟩⟨*lib.text*⟩

kde ⟨*lomitko*⟩ je běžné lomítko (`/`) nebo zpětné lomítko (\) .

Příklad použití programu

V UNIXovém shellu:

```
for i in *.g *.mk; do grid -out 0 $i; done
gimp *.xpm
```

Princip činnosti programu

je popsán důkladně ve zdrojovém souboru `grid.c`. Je tam popsán opravdu *důkladně* – komentářů najdete více než vlastního zdrojového kódu, podobně jako třeba v `tex.web`.

Předpokládáme totiž, že bezduché používání tohoto programu nepřináší lidem žádnou radost. Podstatně více radostí si člověk užije při manuálním řešení těchto lušťovek a zcela nejvíce radosti člověk získá při studiu a pochopení algoritmů, které se pro řešení lušťovek dají implementovat do počítače. Vrchol radosti pak přichází v okamžiku, kdy se člověku podaří implementovat ještě efektivnější algoritmus, než ten, který je použit v tomto programu.

Dáváme všem uživatelům, kteří rozumějí česky (komentáře jsou v češtině), možnost si užít: otevřete si soubor `grid.c` v textovém editoru a pusťte se do čtení...

Věnovali jsme mnoho hodin optimalizaci rychloti algoritmů. Zamítli jsme metodu *brutální síly* a použili jsme metodu *brutální inteligence*. Domníváme se, že právě proto patří náš program k nejrychlejším ve své kategorii a předpokládáme, že také patří k nejlépe dokumentovaným programům na řešení těchto lušťovek.

Další výhodou programu je jeho nezávislost na platformě. Nepoužíváme systém MS Windows (protože to nemáme za potřebí), ale jsme přesvědčeni, že na této obskurní počítačové platformě bude program rovněž snadno implementovatelný. Je to tím, že se program opírá jen o základní knihovnu jazyka C a nevyužívá žádných nadstaveb šitých na míru jen pro konkrétní operační systémy. Program jsme například testovali na předpotopním PC AT 286 16MHz se systémem PC DOS a kompilace i provoz programu na tomto stroji byly bez problémů.