Mittuniversitetet
MID SWEDEN UNIVERSITY

DT028G Introduktion till programmering

Projekt:
Ett härligt parti *Bulls and Cows*

Daniel Bosk*

bullcow.tex 516 2012-12-19 10:34:56Z danbos

Innehåll

1 Introduktion

Bulls and Cows, av vilket *Mastermind* är en känd variant, är ett klassiskt kodknäckningsspel för två spelare där det går ut på att gissa sig vilket tal som motståndaren tänker på. De två rollerna i spelet är alltså en *kodmakare*, som hittar på talet, och en *kodknäckare*, som ska lista ut talet.

Spelet fungerar enligt följande. Jag tänker på ett fyrsiffrigt tal där varje siffra endast får förekomma en gång, upprepningar är alltså inte tillåtna, och du ska gissa på vilket tal det är. Utifrån din gissning svarar jag med ett antal *bulls* eller *cows*. Du får en bull för varje siffra i talet som är på rätt plats och en cow för varje siffra som är rätt men på fel plats. (Notera att du får inte veta vilka tal som gav dig bull eller cow.) För siffror som inte är med i talet får du ingenting. Därefter får du gissa igen utifrån den information du har fått. Detta illustreras enklast med ett exempel.

Exempel 1. Jag tänker på talet 1234. Din första gissning är 1437. Du får då som svar *two bulls* för ettan och trean, och du får *one cow* för fyran. Du gissar härnäst på 2437 och får då svaret *one bull* för trean och *two cows* för tvåan och fyran. Sedan fortsätter spelet tills du fått *four bulls* som svar.

Notera återigen att du inte får veta för vilka siffror du får *bulls* respektive *cows*, detta får du veta här enbart för att exemplet ska vara tydligt.

Det finns lite olika angripssätt för att lista ut talet, se avsnitt 3, och i detta projekt kommer du att titta på ett av dem, men fokus kommer att ligga på att implementera spelet.

Notera att Mastermind och Bulls and Cows skiljer sig något. I Bulls and Cows får siffrorna inte upprepas, vilket de får göra i Mastermind. I Mastermind används också (oftast) bara talen 1-6 medan talen 0-9 används i Bulls and Cows.

indata En mängd $G = \{(g_i, a_i)\}$ av gissningar g_i med respektive svar a_i.
utdata En ny gissning g som inte redan finns i G.

```
1: funktion BullsCows(x, y)
2:     returnera Antalet bulls och cows när x jämförs med y. ▷ Detta ger samma resultat oavsett om x eller y används som hemligt värde.
3: slut funktion

4: funktion GiltigGissning(g, G)
5:     för alla g_i ∈ G genomför
6:         om BullsCows(g, g_i) ≠ a_i då
7:             returnera falskt
8:         slut om
9:     slut för
10:    returnera sant
11: slut funktion

12: funktion GenereraGissning(G)
13:    om G = ∅ då                                   ▷ Detta garanterar att g ∉ G.
14:        Låt g = 1023.
15:    annars
16:        Låt g vara den största gissningen i G adderad med ett.
17:    slut om
18:    medan GiltigGissning(g, G) = falskt genomför ▷ Genomför detta tills att vi hittar en giltig gissning.
19:        Öka g till nästa tal med distinkta siffror.
20:    slut medan
21:    returnera g
22: slut funktion
```

Algoritm 1: En enkel algoritm för att generera en gissning för *Bulls and Cows*.

2 Syfte

Syftet är att fördjupa dina kunskaper inom och utveckla din vana för programmering. I detta projekt kommer du att fokusera på algoritmer, datastrukturer och i viss utsträckning representera dessa data i filer.

3 Teori

Knuth [2] föreslog något år efter att spelet Mastermind först släptes en algoritm för att knäcka koden med som mest fem försök. Många andra har också föreslagit olika algoritmer för att lösa problemet att knäcka koden [exempelvis 4; 6; 1; 3; 5]. I detta projekt ska du till en början titta på en enkel algoritm för att lösa problemet, den algoritm du ska använda dig av ges i algoritm 1.

Algoritm 1 har funktionen `GenereraGissning` för att generera en gissning utifrån tidigare gissningar, eller en initial gissning om inga tidigare gissningar finns. Algoritmen garanteras avslut om gissningarna och svaren i mängden G stämmer överens. Exempelvis om kodmakaren fuskar genom att under spelets gång byta sitt hemliga tal är resultatet av algoritm 1 odefinierat.

4 Uppgift

Projektet är uppdelat i olika betygssteg. För betyget D krävs att du genomför grunduppgiften perfekt utan anmärkningar enligt granskningsprotokollet. Vid maximalt två anmärkningar ges betyget E, vid fler anmärkningar betyget F.

För betygen C-A krävs de extrauppgifter som ges efter grunduppgiften, för ett betyg krävs att samtliga föregående extrauppgifter även är genomförda. (Även dessa ska vara utan anmärkningar, vid anmärkning

sänks betyget fortfarande till E.)

4.1 Grunduppgift

I grundutförandet ska du implementera en version av spelet där programmet gissar på talet som användaren tänker på. Det vill säga programmet agerar kodknäckare och användaren kodmakare. Detta kan du göra genom att implementera algoritm 1 på föregående sida. Användaren tänker alltså på ett tal, programmet skriver ut gissningarna på skärmen och användaren matar in antalet bulls och cows.

4.2 Extrauppgift C

Ditt program ska ha felhantering. Det ska kunna detektera när användaren har fuskat (eller bara tänkt fel) för att programmet inte ska fortsätta i all evighet. En lösning i likhet med att införa en maxgräns på 1000 gissningar är dock inte giltig, programmet ska säga till precis när det går att avgöra att användaren gjort fel – helst med en kommentar i stil med *Ye cheatin' scum!*.

Med felhantering menas även att programmet ska hantera om en användare matar in fel, exempelvis om denne matar in bokstäver där det förväntas heltal.

4.3 Extrauppgift B

Utöka ditt program så att användaren kan agera kodknäckare och programmet är kodmakare, det vill säga ombytta roller jämfört med grunduppgiften. Du ska även skapa en highscorelista.

4.4 Extrauppgift A

I denna extrauppgift ges flera alternativ. Du väljer ett av dessa som du implementerar i ditt program.

- Använd funktionerna i ditt program för att låta programmet spela mot sig självt och med hjälp av highscorefunktionen skapa statistik för vilka tal som är svårast för algoritmen att knäcka.
- Implementera en förbättrad kodknäckningsalgoritm.
- Alla val ska kunna göras via argument till programmet från terminalen. Du kan använda biblioteket getopt(3). Exempelvis om den exekverbara filen heter `bullcow` och vi vill specificera att highscorelistan ska lagras i filen `highscore.txt`:

```
1 $ ./bullcow -f highscore.txt
```

5 Examination

Ditt färdiga projekt ska först granskas av en annan student på kursen med hjälp av det granskningsprotokoll som återfinns på lärplattformen. Efter att du åtgärdat alla påpekanden som uppkommit under granskningen ska du tillsammans med granskaren redovisa ditt projekt och granskningen för en av kursens lärare vid något av de inbokade redovisningstillfällena. När du redovisat och fått godkänt laddar du upp din källkod med tillhörande byggskript[1] till inlämningslådan i lärplattformen.

Referenser

[1] Goddard, Wayne. Mastermind revisited. *J. Combin. Math. Comin. Comp.*, 2004.

[2] Knuth, Donald E. The computer as a master mind. *J. Recreational Mathematics*, 9(1), 1977.

[3] Kooi, B. Yet another Mastermind strategy. *ICGA Journal*, 2005.

[4] Neuwirth, E. Some strategies for Mastermind. *Mathematical Methods of Operations Research*, 1982.

[1] Notera att det är obligatoriskt att ha ett byggskript för programmet.

[5] Slovesnov, Alexey. Search of optimal algorithms for bulls and cows game. URL: `http://bulls-cows.sourceforge.net/bullscows.pdf`, nov 2011.

[6] Vomlel, J. Bayesian networks in Mastermind. *Proceedings of the 7th Czech-Japan Seminar*, 2004.