Reševanje igre »kakuro«

Luka Toni (63120258)
Miha Benčina (63160061)

Študijsko leto 2018/2019

Računalništvo in informatika - univerzitetni program

16. januar 2019

Kazalo

1 Povzetek **1**

2 Uvod **2**

3 Opis **4**
3.1 Neinformirano iskanje 4
3.2 Neinformiran algoritem - iskanje s sestopanjem 5
3.2.1 Korak 1 - Poišči enačbe za rešitev vsot 5
3.2.2 Korak 2 - Permutacije rešitev 5
3.2.3 Korak 3 - Pregled permutacij po enačbah 5
3.2.4 Korak 4 - Preverjanje pravilnosti rešitve 5
3.3 Informirano iskanje 6
3.4 Informirani algoritem 6
3.4.1 Korak 1 - Poišči zaporedje 6
3.4.2 Korak 2 - Presek polj 6
3.4.3 Korak 3 - Nastavi številko za polje 7
3.4.4 Korak 4 - Odstranitev podvojenih sekvenc 7
3.4.5 Korak 5 - Odstranitev nemogočih kombinacij 8

4 Predstavitev rezultatov **9**

5 Zaključek **10**

6 Opis delitve dela **10**

7 Literatura **11**

1 Povzetek

Podanih imamo 6 različnih kakuro iger z eno unikatno rešitvijo, ki jih želimo pravilno rešiti. Po krajši raziskavi je bilo jasno, da na internetu ni lahko najti načina, ki bi reševal to igro (za primerjavo je za sudoku že veliko objavljenih implementacij). Tako sva se odločila, da sama razvijeva svoje algoritme in jih nato primerjava. Za reševanje kakura sva tako razvila dva algoritma oziroma pristopa.

Prvi pristop je neinformiran algoritem iskanje v globino s sestopanjem (backtracking), drugi pa informirani algoritem, ki s pomočjo na začetku podane mreže napredno izloča številke, ki jih je na prvi pogled pri večjih kakuro igrah težko opaziti. Oba pristopa oziroma algoritma bomo primerjali s podanimi igrami.

Da lahko sestavimo veljaven algoritem (torej algoritem, katerega rešitev ne bo v neskladju s pravili igre) je bilo najprej potrebno definirati igro in njene omejitve.

Polja, ki jih ni potrebno izpolniti (torej črni kvadrati in kvadrati, ki imajo v njih napisano vsaj eno vrednost oziroma vsoto) sva predstavila z urejenim parom, polja, ki jih je potrebno izpolniti (oziroma številska polja) pa s številko.

Sestopanje ali vračanje (backtracking) je pristop v programiranju, kjer sistematično pregledujemo vse možnosti rešitve. Sestopanje lahko ponazorimo z drevesom stanj, kjer rešitev najdemo v listih drevesa ali pa je rešitev pot od korena do lista.

Ko neka pot od korena do lista ni več obetavna, se vrnemo nazaj in poskusimo po drugi poti. [1]

Informirani algoritem za reševanje kakura najprej pregleda mrežo in za vsako rešljivo polje shranjuje možne številke. Prav tako za vsako vsoto shranjuje možne kombinacije.

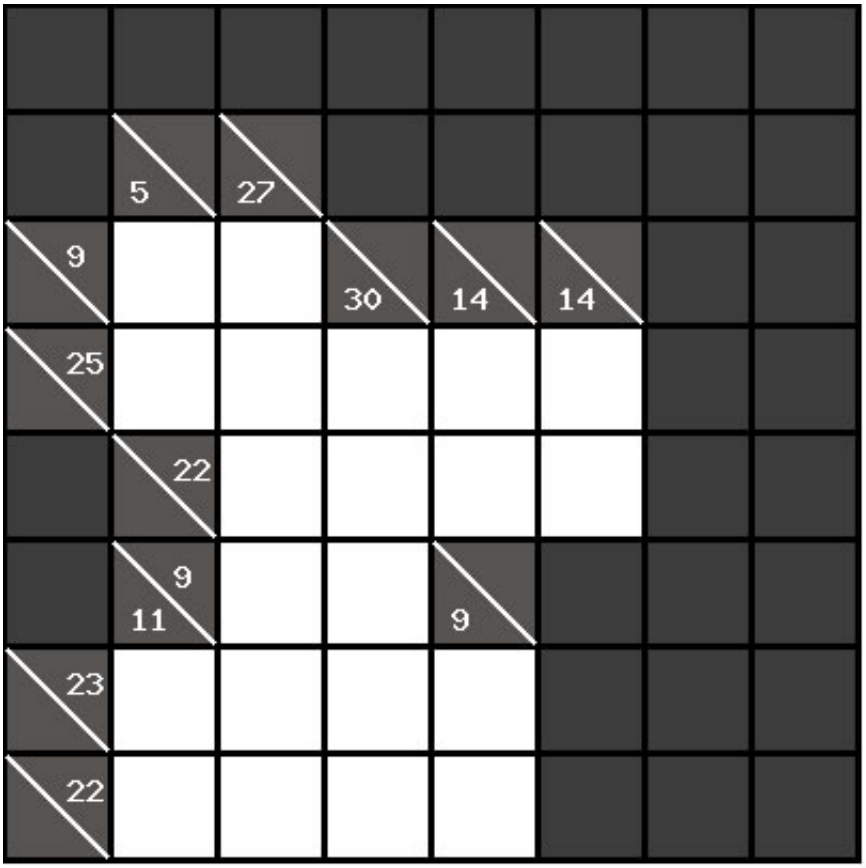
Ko sestavi to informirano mrežo postopoma odstranjuje možne številke, za katere ugotovi, da niso primerne in tako počasi rešuje igro.

Alogritma sta priložena v ločenih datotekah (backtracking.py in solver.py). Za testiranje algoritma je potebno pognati Pythonovo datoteko solver.py, ki izpiše rešitev posameznega kakura in čas, ki ga je vsak algoritem potreboval za reševanje.

Cilj naloge je primerjati algoritma, ki znata pravilno rešiti igro kakuro. Primerjali bomo v kolikšnem času vsak algoritem reši posamezno igro.

2 Uvod

Kakuro je logična igra, ki jih je bila prvič objavljena avgusta 1980 v japonskem časopisu Nikoli [2]. Spodaj je prikaz začetne igre Kakuro [3].



Slika 1: Primer začetne igre Kakuro

Uganka je opisana z naslednjimi pravili:

1. Uganka uporablja pravokotno mrežo črnih in belih celic. Črne celice lahko vsebujejo namige (celo število). Številka pod diagonalnim delilnikom je namig za spodnje celice in številka nad diagonalnim delilnikom je namig za celice na desni.

2. Naloga je vnesti številke od 1 do 9 v bele celice, pri čemer ne kršijo naslednjih omejitev:

   (a) Vsota neprekinjenega niza belih celic v vodoravni ali navpični smer mora biti enaka namigu v črni celici na levi,

   (b) Vse številke v nizu belih celic morajo biti različne.

Velikost mreže je lahko različna najmanjši primer je velikosti 3 vrstic in 3 stolpcev med večje primer pa štejemo kakuro s 30 vrsticami in 30 stolpci. Veljavna Kakuro puzzle ima vedno samo eno rešitv.

Na spletu nisva našla obstoječih uveljavljenih algoritmov za reševanje kakura, zato sva oba algoritma razvila od začetka. Kljub temu pa obstajajo taktike [6] za reševanje kakura. Na primer, da izločimo nemogoča števila, ki ne bi izpolnjevala pravil kakura. Ta taktika je uporabljena tudi v obeh algoritmih.

3 Opis

3.1 Neinformirano iskanje

Neinformirano iskanje, imenovano tudi slepo iskanje ali nevodeno iskanje, je razred iskalnih algoritmov, ki delujejo na "brute-force"način. Izraz "neinformirani"pomeni, da nimajo dodatnih informacij o stanjih, ki jih določa definicija problema. Ti algoritmi se lahko uporabijo za različne probleme iskanja, saj ne upoštevajo ciljnega problema. [5]

- iskanje v širino (breadth-first search)
- iskanje v globino (depth-first search)
- iterativno poglabljanje (iterative deepening)
- cenovno-optimalno iskanje (uniform-cost search)

Za primer neinformirananega algoritma bomo uporabili izboljšan algoritem iskanja v globino, ki namesto vseh naslednjikov generira le enega, hrati pa se v primeru neustrezne rešitve vrne po nivojih navzgor in nadaljuje v drugi vejitvi - iskanje s sestopanjem (backtracking search).

Iskanje s sestopanjem je unčikovito pri manjši globini, ko pa se ta poveča imamo hitro velika kombinatorična eksplozija možnih stanj.

3.2 Neinformiran algoritem - iskanje s sestopanjem

Pri sestopanju rešitev problema iščemo korakoma, sestavljamo podrešitev (komponento) za podrešitvijo (komponento), pri čemer sistematično pregledujemo vse možnosti.

Predpostavimo, da je prostor stanj drevo:

- globina (depth) optimalne rešitve naj bo d
- stopnja vejanja (branching factor) naj bo b na nivoju d imamo torej b^d vozlišč
- največja globina drevesa naj bo max

Pri kakuro igri imamo stopnjo vejanja 9 saj vsebuje le števila od 1 do 9. Za globino pa naj imamo podano število polj, ki jih želimo zapolniti s števili. Z kakuro, kjer iščemo 16 neznanih polj imamo $9^{16} \approx 1.853 \cdot 10^{15}$ možnih postavitev rešitev - vendar je le ena od teh pravilna.

3.2.1 Korak 1 - Poišči enačbe za rešitev vsot

Na začetku je potrebno cel kakuro sprementi v sistem enačb. Najprej podamo velikost kakuro igre (2 2 - velikost igralne ploskve je 2x2) nato pa sledi vsota števil in polja (vrstice oštevilčimo od 1,2,3 naprej, stolpce pa po abecednem vrstnem redu A,B,C,...) za to vsoto (16 A1 A2 - vsota polj A1 in A2 je 16).

3.2.2 Korak 2 - Permutacije rešitev

Za vsako enačbo dobimo možne kombinacij števil med 1 in 9 za dano vsoto enačbe. Vse dane kombinacije je potrebno še permutirati tako, da dobimo vse možne rešitve za dano vsoto.

3.2.3 Korak 3 - Pregled permutacij po enačbah

Ko imamo vse možne permutacije za dano rešitev, je te potrebno združiti v kakuro in jih postopno pregledovati (s sestopanjem).

3.2.4 Korak 4 - Preverjanje pravilnosti rešitve

Kadar vse enačbe izpolnujejo vse pogoje imamo podano pravilno rešitev, saj pravilni kakuro vsebuje le eno rešitev. Vse enačbe vsot števil vsebujejo vrednosti, tako, da imajo vse posamične spremenljivke le eno številko.

3.3 Informirano iskanje

Ključna ideja algoritmov za informirano iskanje je hevristika. Hevristika glede na posamezno vozlišče zagotavlja oceno razdalje od cilja. Za razliko od iskalnega algoritma, ki deluje za kateri koli problem iskanja, je hevristika zasnovana le za določen problem iskanja. [4]

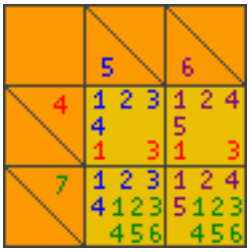
3.4 Informirani algoritem

Informirani algoritem v tej nalogi poizkuša uporabiti vse informacije, ki jih lahko dobi iz kakuro mreže. Problem igre skrči le na številke, ki so možne za posamezne vsote za določena številska polja [7], poleg tega pa pregleda še vse možne kombinacije in izloči nemogoče.

Da pa bo postopek bolj jasen, poglejmo kako algoritem pravzaprav deluje v celoti s pomočjo slik [6].

3.4.1 Korak 1 - Poišči zaporedje

V tabeli za vsako podano vsoto v diagonalnem delu (oranžni okvir z diagonalno črto) izračuna vsa možna števila (rumeni okvir) za podano vsoto. Ta števila predstavljajo potencialno rešitev polja. Prav tako pa si za posamezno vsoto zapomni vse možne kombinacije.



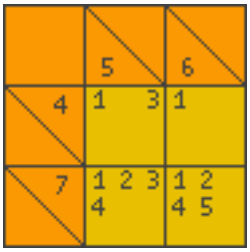
Slika 2: Iskanja zaporedij

Primer iz [Slika 2] Vsota števil prvega stolpca je 5 (označena modro) in ima dve številjski polji, kjer so možne sekvence {{1, 4} {2, 3} {3, 2} {4, 1}}.

3.4.2 Korak 2 - Presek polj

Za vsak številski del (rumeni okvir) imamo vse možne številke glede na navpični in vodoravni del. Tako naredimo presek med navpičnim in vodoravnim delom. Presek števil so možni kandidati za podano polje.

Primer iz [Slika 3]. Vsota števil v navpičnem delu (6), ima naslednje možne sekvence: $\{\{1,5\}\{2,4\}\{4,2\}\{5,1\}\}$ potem so možne navpične številke $\{1,2,4,5\}$. V vodoranem delu pa ima naslednje možne sekvence: $\{\{1,3\}\{3,1\}\}$,

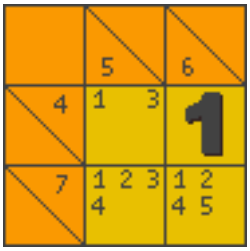


Slika 3: Presek navpičnega in vodoravnega polja

s tega sledi da so možna navpična števila $\{1, 3\}$. Ko naredimo presek med $\{1, 2, 4, 5\}$ in $\{1, 3\}$, dobimo za podan primer edino možno rešitev, ki je $\{1\}$.

3.4.3 Korak 3 - Nastavi številko za polje

Pregledati je potrebno vsak številski del (rumen okvir) in preveriti, če za podano polje obstaja le eno možno število. Kadar to velja nastavimo polje na podano vrednost [Slika 4].



Slika 4: Iskanje enoličnih rešitev za podana polja

3.4.4 Korak 4 - Odstranitev podvojenih sekvenc

Pregledamo vodoravne in navpične dele tako, da polja, ki še nimajo enoličnih rešitev ne vsebujejo števil za katere že imamo enolične rešitve. Preveriti je potrebno da je vsota enaka vodoravnim ali navpičnim poljem. Kadar zaporedja števil niso možna jih odstranimo.

Primer iz [Slika 5]. Številka v diagonali je 4 in ima dve številski polji in dana možna zaporedja (oz. kombinaciji) {{1, 3} {3, 1}}. Drugo število pa je že enolično določena s številom 1 (druga delna vrednost števila je 1). Sledi, da je je vrednost nerešenega polja enaka 3. [Slika 5]. To je v algoritmu implementirano kot prevedba problema na podproblem. Ta problem se je iz "Kako se 2 polji seštejeta v vsoto 4?" pretvoril v "Kako se 1 polje sešteje v vsoto 3?", kar pa je trivialno rešljivo.

Slika 5: Posodobitev prve vrstice

3.4.5 Korak 5 - Odstranitev nemogočih kombinacij

Ko za posamezno številsko polje ugotovimo, katera števila so zanj možna, pa si lahko dodatno pomagamo z možnimi kombinacijami s katerimi lahko izločimo odvečne številke.

Slika 6: Posodobitev druge vrstice

Primer: številka v diagonali je 7 in ima na voljo dva polja, možne kombinacije so {{1, 6} {2, 5} {3, 4} {4, 3} {5, 2} {6, 1}} toda glede na možne številke, ki smo jih definirali v koraku 2, so rešitve prvega polja lahko le vrednosti {1, 2, 3, 4}, za drugo polje pa števila {1, 2, 4, 5}. Sledi, da so edina preostala zaporedja oziroma kombinacije {{2, 5} {3, 4}} [Slika 6]. To preverjanje je v programski kodi implementirano s pomočjo permutacij.

Da bo koda lažje berljiva so deli kode v datoteki solver.py označeni s koraki. Primer:

```
# poiscemo odvecne kombinacije in jih odstranimo - STEP 5
```

4 Predstavitev rezultatov

Kakuro	Neinformirani algoritem	Informirani algoritem
8 x 8	x	64874500 ns
6 x 6	x	57889200 ns
5 x 5	1183244100 ns	3493000 ns
4 x 5	16966700 ns	997900 ns
4 x 4	156200500 ns	2496300 ns
3 x 3	6487500 ns	500900 ns

Informirani algoritmem je precej hitrejši od neinformiranega. Velika odstopnja se kmalu opazijo pri računanju večjih kakuro iger, saj neinformirani alogritem s sestopanjem preseže maksimalni čas za izvedbo algoritma torej 10s (v tabeli je predstavljen z znakom x). Sestopanje ima velik problem pri večjih kakuro igrah, ko nastane preveč možnih kombinacij. Sestopa postopoma in želi vedno priti do same rešitve. Če se nam zgodi, da je rešitev za določen stolpec ravno zadnja permutacija je to zelo časovno zahtevno.

Primer: želimo izračunati vsoto 25 na 5 polj imamo 12 možnih kombinacij. Vsako kombinacijo pa lahko permutiramo na 24 možnih načinov. Torej za le eno vsoto imamo na voljo 12 x 24 = 288 možnih stanj, ki jih moramo pregledati. Zato algoritem s sestopanjem ni primeren za večje kukure, saj bi zanj potreboval zelo veliko časa.

Informirani algoritem je precej boljši, saj namesto da števila ugiba, najprej izloči nemogoča in število v polje vpiše šele, ko je to število edino možno. Po poganjanju algoritma na največji od mrež, ki so bile v tej nalogi uporabljene (8 x 8) ugotovimo, da je tudi za to mrežo rešitev našel v manj kot sekundi.

5 Zaključek

Ugotovila sva, da sta oba algoritma primerna za reševanje manjših kakuro iger. Ko pa se mreže povečujejo, se čas reševanja algoritma iskanja v globino s sestopanjem nenadzorovano poveča.

V primeru, da bi hoteli reševati večje mreže (na primer take, ki so pogosto objavljene v raznih časopisih) je defenitivno bolj primeren informiran algoritem.

Če bi hoteli algoritma primerjati po prostorski zahtevnosti, pa bi ugotovili, da manj prostora porabi neinformirani algoritem. Razlog za to je, da informirani algoritem shranjuje za vsako polje vse možne številke.

Ko to vzamemo pod drobnogled ugotovimo, da to ne spremeni dejstva, da je boljši informirani algoritem.

Če na primer vzamemo največjo igro vidimo, da ima 36 številskih polj. Tudi če bi v vsakem polju shranjevali številke od 1 do vključno 9 (ki jih sicer nikoli ne, ampak jih vedno manj), bi to znašalo 36 x 9=324, kar pa je še vedno zelo malo.

Tako je na splošno bolj optimalen in hitrejši informirani algoritem.

6 Opis delitve dela

Delitev dela med avtorji:

1. Luka
   (a) Priprava poročila za oddajo,
   (b) Opis neinformiranega algoritma,
   (c) Opis informiranega algoritma,
   (d) Postavitev iger kakuro,
   (e) Izdelava algoritma za sestopanje.
2. Miha
   (a) Pisanje poročila,
   (b) Opis informiranega algoritma,
   (c) Izdelava informiranega algoritma,
   (d) Formatiranje izpisa.

7 Literatura

[1] Spletna stran, *http://wiki.fmf.uni-lj.si/wiki/Sestopanje* (03.01.2018)

[2] Spletna stran, *http://www.nikoli.co.jp/en* (27.12.2018)

[3] Spletna stran, *https://www.conceptispuzzles.com/picture/11/1362.gif* (27.12.2018)

[4] Spletna stran, *https://kartikkukreja.wordpress.com/2015/06/07/informed-search-algorithms/* (3.1.2019)

[5] Spletna stran, *https://en.wikiversity.org/wiki/Search_techniques* (3.1.2019)

[6] Spletna stran, *http://amit.metodi.me/oldcode/java/kakuro.php* (4.1.2019)

[7] Spletna stran, *https://www.dcode.fr/kakuro-solver* (2.1.2019)