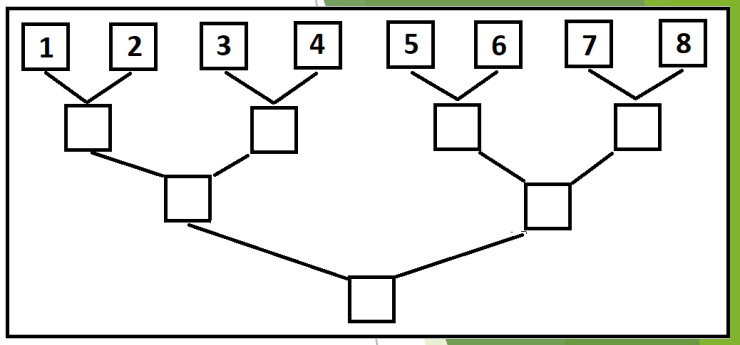
Losovanie turnajov švajčiarskym systémom

Autor: Martin Hošek

Školiteľ: doc. Dr. Tomáš Plachetka

Rôzne systémy losovaní



Round-robin

- hrá každý s každým 1x
- $\frac{n(n-1)}{2}$ partii
- $n-1$ kôl
- v prípade nepárneho počtu hráčov sa pridá jeden fiktívny hráč

„Stromový“ systém

- vyraďovací systém
- aj dobrý hráč môže byť vyradený na začiatku
- $\lceil \log_2 n \rceil$ kôl

Fáza	Počet kôl	Počet hráčov	O čo sa hrá
Finále	1	2	Víťazstvo
Semifinále	2	4	Postup do finále
Štvrťfinále	4	8	Postup do semifinále
Osemfinále	8	16	Postup do štvrťfinále

1. kolo	1 – 8	2 – 7	3 – 6	4 – 5
2. kolo	8 – 5	6 – 4	7 – 3	1 – 2
3. kolo	2 – 8	3 – 1	4 – 7	5 – 6
4. kolo	8 – 6	7 – 5	1 – 4	2 – 3
5. kolo	3 – 8	4 – 2	5 – 1	6 – 7
6. kolo	8 – 7	1 – 6	2 – 5	3 – 4
7. kolo	4 – 8	5 – 3	6 – 2	7 – 1

Švajčiarsky systém

Základné bodovanie	
WIN	1.0
DRAW	0.5
LOSE	0.0
UNPAIRED	1.0

- každé kolo sa páruje samostatne, na základe údajov z predošlých kôl
  - prvé kolá nie sú algoritmicky zaujímavé
- dá sa použiť na losovanie rôznych turnajov
  - my sme používali šachovú terminológiu (švajčiarsky systém sa využíva aj v šachu)
- hráči sa párujú tak, aby rozdiel aktuálnych bodov hráčov v jednotlivých pároch bol minimálny (najlepšie žiadny)
- nemusí hrať každý s každým
- kto je nespárovaný, vyhráva kontumačne
- každý môže byť počas turnaja nespárovaný len raz
- nikto nemôže hrať 2x s tým istým hráčom
- na **vstupe** párovania je aktuálny stav turnaja (záznam)
- na **výstupe** losovania je množina párov, ktoré budú hrať v ďalšom kole
- množina párov na výstupe dodržiava povinné pravidlá a je optimalizovaná na základe preferenčných pravidiel

Príklad pravidiel

- povinné pravidlá
  - žiadny hráč nemôže hrať s iným hráčom viac ako 1-krát
  - každý môže byť nespárovaný systémom len raz
  - pravidlá o počte farieb (rozdiel počtu farieb)
- preferenčné pravidlá
  1. **maximalizovanie počtu párov vo výslednom párovaní**
  2. hľadanie párov takých, aby rozdiel bodov hráčov v pároch bol čo najmenší, v ideálnom prípade žiadny (primárne sa minimalizuje rozdiel bodov v páre s lepšími hráčmi)
  3. vyrovnávanie počtu farieb (striedanie farieb), tak aby v ďalšom kole malo čo najmenej hráčov vynútenú farbu

Príklad vstupu a výstupu

- príklad vstupu zapísaný v štandardnom .trf súbore
  - počet kôl: 5
  - počet hráčov: 6
  - počet odohraných kôl: 3

```
XXR 5  3
001    1      Test0001 Player0001        1500     3.0    1     4 w 1    3 w 1    6 b 1
001    2      Test0002 Player0002        1500     2.0    2     6 w 1    5 w 1    3 b 0
001    3      Test0003 Player0003        1500     1.5    3     5 w =    1 b 0    2 w 1
001    4      Test0004 Player0004        1500     1.0    4     1 b 0    6 w =    5 b =
001    5      Test0006 Player0006        1500     1.0    5     3 b =    2 b 0    4 w =
001    6      Test0005 Player0005        1500     0.5    6     2 b 0    4 b =    1 w 0
```

round 1 | round 2 | round 3

2 (rating column) | 1 (points column)

záznam o odohranom zápase: 1 | 2 | 3

- ďalšie kolo (výstup párovania):
  - 1-2; 3-4; 5-6
  - tieto dvojice spĺňajú pravidlá párovania

Losovanie - algoritmus

```
Set<Pair> losovanie(Set<Player> vstup) {
      Set<Pair> pairing;
      int c = zistiKardinalituParovania(vstup);                    KROK 1
      int removeCount = vstup.playersCount() - c*2; //pocet nesparovanych
      Set<Player> nH; //nevyradeniHraci
      nH = vyradPotrebnyPocetHracov(vstup, removeCount, c);        KROK 2

      for(každý nevyradený hráč H1 (zoradení podľa priority)z množiny nH){
            if (je H1 spárovaný) continue;
            for(pre každého prípustného súpera H2 pre daného hráča H1 (zoradeni podla priority)) {
                  if (prípustný súper H2 už je spárovaný) continue;
                  if (c-1 == kardinalitaBezParu(nH bez testovaneho paru [H1; H2]) {
                        c--;
                        nH -= H1;
                        nH -= H2;
                        pairing.add(new Pair(H1, H2));
                        break;
                  }
            }
      }                                                            KROK 3
      return pairing;
}
```

```
Set<Player> vyradPotrebnyPocetHracov(hraci, int count, int c)
{
      if (maxCount == 0) return vstup;
      for(každý hráč H z množiny hraci zoradení vzostupne) {
            if (H je vyradeny) continue;
            if (c == kardinalitaBezHraca(hraci - H)) {
                  hraci =- H;
                  vyradPotrebnyPocetHracov(hraci, count-1, c);
                  break;
            }
      }
}
```

Maximum Weighted Matching

- využitie tohto podproblému pri implementácii metód z algoritmu, ktoré rátajú maximálnu kardinalitu párovania
- transformácia vstupu na graf, kde hráči sú vrcholy a prípustné páry sú hrany
- grafový optimalizačný problém
- vstup: neorientovaný ohodnotený graf bez slučiek
- výstup: množina párov, resp. hrán (párovanie)
- vlastnosti párovania:
  - pre každý vrchol zo vstupného grafu platí, že sa vyskytuje v maximálne jednej hrane vo výstupnom párovaní. Ak chceme perfektné párovanie, tak každý vrchol sa musí vyskytovať v práve jednej hrane vo výstupnom párovaní
  - súčet váh jednotlivých hrán v párovaní je podľa optimalizačného zmyslu buď najmenší možný, alebo najväčší možný

Algoritmus Blossom

- rieši podproblém Maximum Weighted Matching
- implementácia: *BLOSSOM V*
- časová zložitosť: $O(|E||V|^2)$
- Java a C++
- autor: Vladimir Kolmogorov
- ohodnotený alebo neohodnotený graf
- *MaximumWeightedMatching*
  - na vstupe je *všeobecný graf*
- *MaximumWeightedPerfectMatching*
  - na vstupe je *graf s perfektným párovaním* (graf, v ktorom existuje perfektné párovanie)

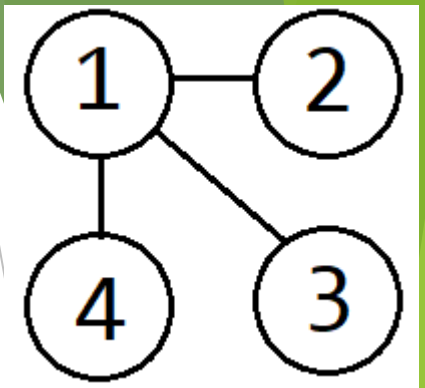
Losovanie – implementácia algoritmu

KROK 1: Nájsť minimálny počet hráčov, ktorí musia byť nespárovaní

- **vstup**: hráči spolu s prípustnými hranami (jednotkový graf), **výstup**: kardinalita párovania
- vyrobiť jednotkový graf (všetky hrany budú mať rovnakú váhu), spustiť Blossom algoritmus (metódu *MaximumWeightedMatching*) na tomto grafe

KROK 2: Určiť konkrétnych nespárovaných hráčov (odspodu – od najhoršieho)

- **vstup**: hráči spolu s prípustnými hranami (jednotkový graf)
- prehľadávať hráčov zoradených podľa bodov odspodu
  - ak sa po odobraní hráča (a jeho hrán) z grafu zníži kardinalita párovania, tak je tento výber zlý, vráti sa späť spolu s jeho hranami do grafu a pokračuje sa s nasledujúcim hráčom v poradí
- na overovanie kardinality sa použije Blossom algoritmus
- **výstup**: množina konkrétnych nespárovaných hráčov, graf hráčov spolu s ich hranami, ktorých budeme párovať (graf, v ktorom existuje perfektné párovanie)



Príklad hráča 1 spolu s jeho prípustnými hranami: 2, 3 a 4

Losovanie – implementácia algoritmu

KROK 3: Nájdenie konkrétnych párov

1. na vstupe tohto kroku bude jednotkový graf s perfektným párovaním so zoradenými prípustnými hranami podľa priority pre jednotlivých hráčov (graf už bez vyradených hráčov v predošlom kroku)
   - najprv budú zoradené hrany najlepšieho hráča podľa priority pre najlepšieho hráča
   - potom nasledujú hrany druhého najlepšieho hráča zoradené podľa priority pre druhého najlepšieho hráča ... až na konci tohto zoradeného zoznamu budú hrany najhoršieho hráča
2. algoritmus bude hľadať najbližšiu hranu (v smere od najlepšej po najhoršiu), ktorá je prípustná
3. odstráni túto hranu (hranu ***H***) spolu s hráčmi v tejto hrane (a všetkými ich hranami)
4. pomocou Blossom overí (metóda *MaximumWeightedPerfectMatching*), či sa bude dať zvyšok kompletne spárovať
5. ak áno, odstránená hrana ***H*** (v kroku 3) je už jedna hrana z množiny párovania
6. ak nie, oboch odstránených hráčov (s ich hranami) v kroku 3 vráti späť a algoritmus opakuje tieto kroky od bodu 2 a pokračuje s nasledujúcou hranou v poradí

Losovanie – implementácia algoritmu

- pri hľadaní konkrétnych nespárovaných hráčoch (krok 2) sa po nájdení konkrétneho hráča zisťuje, či sa dá zvyšok spárovať bez neho - či sa dá spárovať toľko párov, ako bolo zistené v kroku 1, pri zisťovaní maximálnej kardinality párovania (ak nie, tak treba nájsť iného hráča)
- podobne pri hľadaní konkrétnych párov (v kroku 3) sa pri nájdení konkrétneho páru hneď zisťuje, či sa dá zvyšok celý spárovať (ak nie, tak výber nie je dobrý a treba nájsť iný pár)
  - keďže na vstupe v kroku 3 bude graf s perfektným párovaním (ktorý nám zabezpečí krok 2), tak nejaké párovanie bude existovať vždy
- týmito krokmi sa vyhneme nepríjemnému **backtracku** v krajných prípadoch, kde by sa mohlo stať že až pri poslednom hráčovi zistíme, že sa už nedá spárovať a teda by sme sa mohli dostať až naspäť k párovaniu prvého hráča
- krajný prípad je graf, kde vrcholy sú malého stupňa

Krok 1:
Zistenie počtu nespárovaných hráčov

Krok 2:
Zistenie konkrétnych nespárovaných hráčov

Krok 3:
Zistenie konkrétnych párov

Vylepšenia základného algoritmu

- základný algoritmus overuje pri každom nájdení páru, či sa dá zvyšok spárovať
  - minimálny počet overovaní je teda rovný počtu párov
- vylepšenie 1 (GWM – GroupWeightedMatching)
  - každej hrane zo vstupného *grafu s perfektným párovaním* sa priradí váha rovnajúca sa súčinu bodov oboch hráčov
  - na tomto grafe sa spustí Blossom (metóda *MaximumWeightedPerfectMatching*) len raz
  - nevýhoda: v krajných prípadoch je niekedy rozdiel bodov hráčov v lepšom páre väčší ako v horšom páre
- vylepšenie 2 (SWM – SemiWeightedMatching)
  - hráči sa rozdelia do skupín podľa bodov
  - párovať sa bude postupne každá skupina samostatne, od najlepšej po najhoršiu
  - ak v niektorej skupine zostane nejaký hráč nespárovaný, označíme ho za downFloatera, ktorý prepadne do nižšej skupiny
  - downFloater môže prepadnúť maximálne o jednu skupinu, t.j. bude primárne spárovaný v nasledujúcej skupine, do ktorej prepadol
  - počet volaní algoritmu Blossom je rovný počtu skupín (počet rozdielnych bodov)
  - ak musí byť niekde rozdiel bodov, tak je vždy horšom páre

Porovnanie s JaVaFo

- *JaVaFo* je existujúca implementácia losovania švajčiarskym systémom v jazyku Java
- autor: Roberto Ricca
- vyrobili sme vstup taký, že v ňom neexistovalo kompletné párovanie (dajú sa spárovať len páry 3-4 a 5-6, pár 1-2 porušuje povinné pravidlá)
- po transformácií vstupu na graf v tomto grafe neexistuje perfektné párovanie
- náš algoritmus dal na výstupe maximálny možný počet párov (aj keď nie kompletné párovanie)
- *JaVaFo* vyhodil chybu, že neexistuje párovanie

Porovnanie časovej zložitosti (efektivity) nášho algoritmu s algoritmom JaVaFo

- na obrázku je záznam z testovania nekrajných vstupov
- čísla v tabuľke znamenajú čas v *milisekundách*
- stĺpec JVF znamená použitie algoritmu *JaVaFo*
- stĺpec NWM znamená použitie nášho algoritmu bez váhovania (základného)
- stĺpec SWM znamená použitie nášho algoritmu s váhovaním a párovaním po skupinách
- stĺpec GWM znamená použitie nášho algoritmu s váhovaním a párovaním všetkých hráčov naraz
- veľkosť vstupu je prvé číslo z názvu
  - pre vstup `in512_3_10.trf` je vstup s veľkosťou 512 hráčov a podobne

Názov vstupu	JVF	NWM	SWM	GWM
in20_1_3.trf	980	257	222	231
in52_4_9.trf	979	594	677	422
in71_3_7.trf	1378	910	554	453
in128_3_10.trf	1784	2302	1298	1236
in129_3_10.trf	1921	2084	1386	1038
in256_3_10.trf	1967	7467	2813	2254
in257_3_10.trf	2092	5930	2812	2626
in512_3_10.trf	2625	51259	6916	5491
in512_10_20.trf	4140	54392	12247	6234
in513_3_10.trf	2915	54465	8722	6030
in600_1_5.trf	3234	87793	6329	6038
in601_1_5.trf	2711	104961	8151	7538
in1024_3_10.trf	13556	554757	44229	30218
in1025_3_10.trf	4915	572479	31544	26299
in2048_2_10.trf	-	9617045	209841	74457
in2049_2_10.trf	-	10944592	227000	176905

Teoretické vylepšenie

- výroba grafu ako aj beh samotného algoritmu Blossom trvá nezanedbateľný čas
- cieľ: volať Blossom algoritmus len jedenkrát
- aby sme mohli volať Blossom len jedenkrát, tak musíme priradiť váhy hranám tak, aby sa primárne párovал najlepší hráč, potom druhý najlepší hráč, ... až sa bude párovať najhorší hráč
- váhy majú veľký rozsah v reálnych číslach (celá aj desatinná časť)
- praktické obmedzenie: aritmetika má určitú presnosť
- nutnosť využívať Blossom viackrát

Zhrnutie

- náš algoritmus generuje párovanie, ktoré dodržiava povinné pravidlá a optimalizuje preferenčné pravidlá
- JaVaFo zlyhá na vstupoch, ak neexistuje kompletné párovanie, zatiaľ čo náš algoritmus spáruje maximum párov, ktoré spárovať idú (aj keď nejdú spárovať všetky)
- na malých vstupoch (približne 1000 a menej hráčov) je JaVaFo rýchlejší ako náš algoritmus
- ak je vstup príliš veľký (konkrétne už vstup s 2048 hráčmi a dvomi kolami), tak JaVaFo nevygeneruje výstup v rozumnom čase (t.j. ani za 3 hodiny), zatiaľ čo náš algoritmus s váhovaním vygeneruje na tomto vstupe výstup za 5 minút
- v prípade nesprávnej syntaxe JaVaFo vyhodí chybu, pričom nenapíše aký je to typ chyby a kde sa vo vstupnom .trf súbore chyba nachádza. Náš algoritmus vypíše, kde presne sa chyba nachádza a aký je to typ chyby.
- v prípade logickej chyby túto chybu JaVaFo ignoruje, pričom náš program vypíše chybu a skončí sa.
  - logická chyba je porušenie povinných pravidiel v zázname turnaja zo vstupného .trf súboru

Odpovede k otázkam z posudkov

- zrýchlený švajčiarsky systém (na začiatku sa hráči rozdelia do niekoľkých skupín) a tieto skupiny sa párujú zvlášť prvé dve/tri kolá)
  - možné riešenie 1: vyrobiť 3 turnaje reprezentujúce tieto tri skupiny a po odohraní týchto dvoch/troch kôl zlúčiť turnaj tieto tri turnaje do jedného
  - možné riešenie 2: umelo zvýšiť body hráčom v daných skupinách tak, aby sa párovala každá samostatne

Rast časov jednotlivých vstupov s využitím jednotlivých algoritmov

vstup	rozdiel	128	x	256	x	512	x	1024	x	2048
JVF algoritmus		1.8	1.1	2	1.4	2.7	1.9	5	-	-
NWM algoritmus		2.2	3.0	6.5	8.2	52.8	10.6	560	17.9	10000
SWM algoritmus		1.3	2.2	2.8	2.8	7.9	4.7	37	5.7	210
GWM algoritmus		1.1	2.2	2.4	2.4	5.8	4.8	28	6.3	170

Naše algoritmy majú polynomiálnu časovú zložitosť

- načítanie vstupu a vypísanie výstupu je v polynomiálnom čase
- výroba grafu prebieha v polynomiálnom čase
- beh Blossom algoritmu je v polynomiálnom čase
- počet volaní algoritmu Blossom je polynomiálny