Losovanie turnajov švajčiarskym systémom.

Plán práce:

- Spraviť párovací algoritmus
- Spraviť spracovanie outputu z TRF interpretera
- Dokončiť aplikáciu
- Porovnanie a testovanie

Výsledok:

- Nájdenie knižnice JGraphT
- Spracovanie údajov zo vstupného súboru TRF do pracovných štruktúr (tried)
- Transformácia problému losovania na graf tak, aby som mohol využívať metódy z JGraphT
- Spravenie celého párovacieho algoritmu a porovnanie verzií
- Otestovanie funkčnosti programu a porovnanie s JaVaFo
- Vyrobenie analyzátora pre výsledné párovanie

Stránka na JavaFo: http://www.rrweb.org/javafo/aum/JaVaFo2_AUM.htm

TRF formát: https://www.fide.com/FIDE/handbook/C04Annex2_TRF16.pdf

Odkaz na riešenie (weighted) maximum matching pomocou blossoms (program v C++, bez externých knižníc, ktorý som využíval, kým som nenašiel knižnicu JGraphT:

https://pub.ist.ac.at/~vnk/software.html

Časť 4. BLOSSOM V, odkaz priamo na program:

https://pub.ist.ac.at/~vnk/software/blossom5-v2.05.src.tar.gz

Ďalšie užitočné linky:

https://depth-first.com/articles/2019/04/02/the-maximum-matching-problem/

https://en.wikipedia.org/wiki/Blossom_algorithm

https://jgrapht.org/javadoc-1.4.0/org/jgrapht/alg/matching/blossom/v5/package-summary.html

---

Pôvodný program na párovanie:

https://github.com/BieremaBoyzProgramming/bbpPairings

Funguje tak ako aj JaVaFo (akurát podstatne rýchlejšie, keďže je napísaný v jazyku C++), zvládne aj vstupy veľkosti 2000 a viac hráčov (toto JaVaFo a náš program nezvládne, pretože vyhodí chybu `Java Heap Space`)

Manuál na skompilovanie a spustenie párovacieho programu:

Priečinok **src**, ktorý sa nachádza v priečinku **pairing.zip** je pracovný priečinok.

Obsah priečinka **src** je možné dať do iného ľubovoľného pracovného priečinka.

1. Treba sa v konzole prepnúť do pracovného priečinka src (resp. do iného, ak si ako pracovný priečinok zvolíte iný) – t. j. command „`cd [path]/src`“;
2. Ak chcete aby bol pracovný priečinok iný ako **src**, tak obsah priečinka src treba skopírovať do tohto priečinka;
3. Treba skontrolovať, či máte nainštalovanú javu (myslím že by mal stačiť príkaz „**`javac -version`**“ alebo „**`java -version`**“) – ja mám verziu „**`11.0.16`**“;
4. Ak ste v pracovnom priečinku, tak program treba skompilovať príkazom „**`javac -d . BB_Pairing/getPossiblePairs`** a vytvoria sa skompilované súbory s príponou „**`.class`**“;
5. Program spustíte príkazom „**`java BB_Pairing.getPossiblePairs`“** a zobrazí sa vám manuál;

V priečinku **files** sú nejaké vstupné súbory, na ktorých som to testoval. Odporúčam zadať príkaz vo formáte: „**`java BB_Pairing.getPossiblePairs -i [vstup] -a -o [výstup]`**“.

Grafová knižnica, ktorú som v programe využíval, je v priečinku „**org**“ (všetky triedy sa nevyužívajú).

Aktuálna verzia programu má 3 funkcie (metódy), ktoré sa dajú spustiť (všetky sú v pracovnom priečinku **BB_Pairing**).

- Prvá a najdôležitejšia je funkcia je samotného párovacieho algoritmu, ktorý sa spúšťa pomocou triedy **getPossiblePairs**, tak, ako je to vyššie popísané v manuáli.
- Druhá funkcia umožňuje spustiť párovanie s využití programu JaVaFo. Táto funkcia sa spúšťa pomocou triedy **getPossiblePairsByJavaFo**.
- Tretia funkcia umožňuje spustiť náhodné generovanie turnajov (ktoré som využíval aj ja pri testovaní) pomocou triedy **getRandomGeneratedFileByJVF**.

Všetky tieto triedy sa dajú skompilovať a spustiť tak, ako je uvedené vyššie, t. j. príkazom

„**`javac -d . BB_Pairing/[class]`“** pre kompiláciu triedy **[class]**

a

„**`java BB_Pairing.getPossiblePairs [args]`“** pre spustenie triedy **[class]**

kde **[args]** sú možné argumenty

V prípade, že nie sú uvedené žiadne argumenty, tak sa po spustení zobrazí manuál.

Samotný párovací program má 3 hlavné časti:

1. Načítanie vstupu zo štandardného TRF súboru
    a. Syntaktická analýza (po riadkoch)
    b. Logická analýza (analýza po kolách)
2. Zistenie, ktorí hráči nebudú v danom kole, ktoré losuje algoritmus, hrať.
    a. Vyrobenie poľa prípustných hrán spolu so zoznamom hráčov, ktorých sme nevyradili.
3. Zistenie samotných párov:
    a. Vyrobením neorientovaného neohodnoteného perfektného grafu zo zoznamu hrán takého, že postupne nájdeme najlepšie párovanie. (Perfektný graf je graf, v ktorom existuje perfektné párovanie.)
    b. Vyrobením neorientovaného ohodnoteného perfektného grafu takého, že knižničná metóda MaximumWeightedPerfectMatching nájde najlepšie párovanie na jedno volanie tejto metódy.
    c. Rozdelením hráčov do skupín podľa počtu bodov a každú bodovú skupinu párovať zvlášť.
4. Vyrobenie výstupu. Rozhodnutie kto bude hrať s akou farbou vo výsledných dvojiciach. (Vyrobiť súbor s analýzou párovania)

(1a) V syntaktickej analýze sa kontroluje syntax v zmysle formátu TRF (kontrolujú sa len riadky podstatné pre losovanie, t. j. riadky s hlavičkou **001** (dáta o hráčoch) a **XXR** (dáta o maximálnom počte kôl).

(1b) V logickej analýze sa kontroluje dodržanie povinných pravidiel:

- Počet farieb (za sebou a priebežný počet)
- Či nie je hráč 2-x spárovaný s tým istým hráčom (existuje výnimka v prípade, že výsledok je „forfeit win/lose“ = „prepadnutá výhra/prehra“ (hra, ktorá sa ani nezačala) – vtedy môže byť hráč opakovane spárovaný)
- Ak mal v niektorom kole hráč 1 spárovaného súpera 4, tak hráč 4 musí mať v danom kole spárovaného súpera 1 a zároveň výsledky pre týchto hráčov musia byť navzájom opačné (win-lose/lose-win/draw-draw). Tiež farby pre týchto hráčov musia byť opačné (White-Black/Black-White). **Toto platí pre každé kolo a pre každú dvojicu.**
- Pre každého hráča platí, že môže byť systémom nespárovaný len raz.

(2) Vyrobiť graf so všetkými hráčmi, ktorí v danom kole môžu hrať, spolu s hranami, ktoré neporušujú povinné pravidlá (prípustné hrany).

Pseudokód 1 (na nájdenie počtu hráčov, ktorých musíme vylúčiť):

Nájsť minimálny počet hráčov, ktorí musia byť nespárovaní (vyrobiť jednotkový graf ako vstup pre MM - MaximumWeightedMatching):

- vytvoriť vrchol v grafe pre každého hráča, ktorý môže v danom kole hrať (väčšinou všetci hráči, niekedy je dopredu dané, ktorí hráči nemôžu v danom kole hrať z iných príčin ako párovanie, napr. choroba)
- vytvoriť hrany v grafe s váhou 1, ktoré reprezentujú všetky možné (prípustné) páry
- spustiť MM a výstup z MM je kardinalita grafu - maximálny počet hrán v grafe, ktoré sa dajú spárovať (t.j. počet možných párov vo výslednom párovaní, ak budeme ďalej písať o kardinalite, tak máme na mysli kardinalitu množiny párov)

Pseudokód 2 (na nájdenie konkrétnych hráčov, ktorých vylúčime):

Určiť konkrétnych nespárovaných hráčov (odspodu): prehľadávať hráčov zoradených podľa určených kritérií vzostupne, s využívaním MM. V predchádzajúcom pseudokóde zistíme, koľko hráčov musíme vyradiť, v tomto kroku ich po jednom vyradíme:

a) vymažeme z grafu najhoršieho hráča (aj s jeho hranami)
b) vyskúšame (s využitím MM) či sa kardinalita nezmenila (nezmenšila)
c) ak nie, hráča sme úspešne vymazali z grafu (a teda je pre toto kolo vyradený)
d) ak sa kardinalita zmenila, tak tohto hráča (aj jeho hranami) vrátime späť do grafu a pokračujeme od bodu (a) s nasledujúcim hráčom v poradí (podľa zoradenia odspodu)

Ak sme už vyradili taký počet hráčov, aký sme zistili v predchádzajúcej časti, tak táto časť je ukončená a ako výstup prehľadávania bude množina konkrétnych nespárovaných hráčov v danom kole. Výstup z tejto časti bude teda množina vyradených hráčov a tiež perfektný graf zvyšných hráčov (spolu s ich hranami).

(3a) Každý hráč bude mať množinu hráčov, s ktorými môže hrať (možné hrany) (aby sa dodržali pravidlá o farbách a také, že nikto nemôže spolu hrať viac ako raz).

Hráčov musí byť párny počet, čo mi zabezpečí vyraďovanie hráčov.

Pseudokód 3 (na nájdenie konkrétnych párov):

1. Vymažeme z grafu dvoch hráčov (spolu s ich hranami, t.j. hranami jedného aj druhého hráča) - hráčov, ktorí sú v najlepšom páre (podľa priority).
2. Vyskúšame (s využitím MM) či sa kardinalita grafu (po vymazaní danej hrany z bodu 1) zmenšila presne o 1, keďže sme odobrali dvoch hráčov a testujeme, či po výbere tohoto páru môžeme spárovať zvyšných hráčov tak, aby výsledná kardinalita grafu zostala nezmenená.
3. Ak áno, týchto hráčov sme úspešne vymazali z grafu (a teda sme určili jeden z párov - pár, ktorý bude vo výslednej množine párov).
4. Ak sa kardinalita zmenšila viac ako o 1, tak týchto hráčov (aj s ich hranami) vrátime späť do grafu, keďže s touto hranou sa nepodarí spárovať zvyšných hráčov tak, aby celková kardinalita zostala zachovaná a pokračujeme od bodu (1) s nasledujúcou hranou v poradí (podľa priority zvrchu).

Ak už sme ukončili túto časť t.j. zistili množinu všetkých hrán, ktorá bude ako výstup (táto množina má kardinalitu takú, akú sme zistili v pseudokóde 1), tak graf už bude prázdny a párovanie máme vyrobené, táto množina bude ako vstup pre bod (4) – vyrobenie výstupu zo zistených párov.

(3b) Každá prípustná hrana bude mať váhu rovnajúcu sa súčinu bodov hráčov z konkrétnej prípustnej hrany.

(3c) Hráči sa budú postupne párovať po skupinách (od najlepšej po najhoršiu). Ak v niektorej skupine zostane niektorý hráč nespárovaný, tak sa stane z neho downFloater a prepadne do nižšej skupiny, kde už musí byť spárovaný (downFloater môže prepadnúť o maximálne jednu skupinu).

V bode (3b) a (3c) sa dajú zohľadniť preferencie o farbách tak, že sa pri preferovaných hranách zvýši váha tak, aby neovplyvnila výber párov čo sa týka bodov (inými slovami, ak existujú dva páry $[H_1, H_2]$ a $[H_3, H_4]$ také, že $B(H_1)=B(H_2)$ a $B(H_2)=B(H_4)$, kde $B(X)$ znamená poet bodov hráča X, tak vtedy sa aplikuje preferencia o farbách)

(4) priradenie farieb hráčom vo výsledných dvojiciach podľa aktuálneho počtu posledných farieb a priebežného počtu farieb (po priradení: prvý hráč z dvojice bude biely, druhý čierny)

Porovnanie štandardných vstupov medzi rôznymi algoritmami. Hodnoty v bunkách znamenajú čas v ms. Stĺpec JVF znamená použitie implementácie JaVaFo, stĺpec NWM znamená použitie môjho algoritmu bez váhovania (algoritmus 3a), stĺpec SWM znamená použitie mojej implementácie kde sa páruje každá skupina hráčov zvlášť (algoritmus 3b), stĺpec GWM znamená použitie môjho algoritmu s váhovaním a grupovaním (algoritmus 3c). Poznámky k tabuľke sú pod ňou.

Názov vstupu	JVF	NWM	SWM	GWM	poznámka
in20_1_3.trf	980	257	222	231	
in52_4_9.trf	979	594	677	422	
in71_3_7.trf	1378	910	554	453	(1)
in128_3_10.trf	1784	2302	1298	1236	(2)
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	-	18081	5879	6357	(3) (4)
in2049_2_10.trf	-	17145	6669	5933	(3) (4)

(1) Vstup stiahnutý zo stránky z http://www.rrweb.org/javafo/aum/TRFXSample2.txt (zvyšné štandardné vstupy sú vygenerované pomocou generátora náhodných turnajov, s využitím JaVaFo).

(2) JVF, SWM a GWM mali celkový súčet diferencií 2.0, maximálnu diferenciu 0.5 a počet párov s diferenciou 4; NWM mal celkový súčet diferencií 3.0, maximálnu diferenciu 0.5 a počet párov s diferenciou 6.

(3) Generátor turnajov (s využitím JaVaFo) aj po opakovaných pokusoch nevygeneroval turnaje s 2048 a 2049 hráčmi, s tromi odohranými kolami; s dvomi kolami vygeneroval.

(4) Všetky 4 algoritmy skončili s chybou heap space (vyčerpanie miesta), preto sme skúsili náš algoritmus taký, že každému hráčovi bude generovať len troch súperov (troch najlepších súperov, z usporiadanej množiny ktorú vráti metóda getPossiblePlayers() a ktorí spĺňajú podmienky pre párovanie). Toto sme docielili pomocou argumentov „-l“ a „3“. V prípade veľkých vstupov sa výrazne zmenší graf čo sa týka počtu hrán, ale v krajnom prípade nemusí existovať platné losovanie, aj keď bez použitia tohto algoritmu s uvedenými argumentmi by to možné bolo. Algoritmus bbpPairings dokáže vygenerovať rýchlo aj párovanie s viac ako 2000 hráčmi.