Ezoteryczne Kartki
Skojarzenia

Jakub Bachurski

wersja 1.0.1

1 Wstęp

Po przepływach będziemy rozważali inne sposoby modelowania problemów optymalizacyjnych. Cała zabawa będzie działa się w większości w grafach dwudzielnych, bo posiadają przydatne dla nas własności. Tym razem będą to mniej ogólne rozważania, dzięki czemu powstałe algorytmy będą prostsze do napisania, a problemy bardziej schematyczne.

Jednak w grafach dwudzielnych nie jesteśmy ograniczeni do samych skojarzeń: jest w nich kilka przydatnych relacji (przynoszących na myśl Max-Flow Min-Cut), dzięki którym jednym algorytmem rozwiążemy wiele problemów.

Na tej kartce, wbrew tradycyjnemu sposobowi przedstawieniu skojarzeń, korzystam ze sformułowania maksymalnego przepływu i metody Forda-Fulkersona.

2 Czym są skojarzenia?

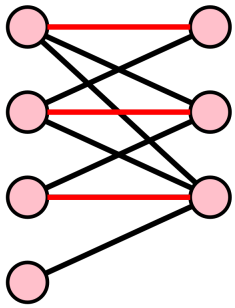
2.1 Intuicja i definicja

Skojarzenia mają na celu modelować parowanie obiektów, przy czym każdy obiekt należy do co najwyżej jednej pary. Modelujemy je grafami (bo tak jest wygodnie), tak, że wierzchołki są obiektami, a krawędź symbolizuje możliwość sparowania ze sobą dwóch obiektów.

Przechodząc do definicji: *skojarzeniem* nazywamy taki podzbiór krawędzi, że każdy wierzchołek ma co najwyżej jedną sąsiadującą krawędź, która zawiera się w skojarzeniu. Jeżeli taka krawędź istnieje dla pewnego wierzchołka v, to jej drugi koniec jest wierzchołkiem *skojarzonym* z v (jeżeli wierzchołek jest skojarzony, to znaczy że ma incydentną krawędź w skojarzeniu). Wielkością skojarzenia jest oczywiście liczba krawędzi, która się w nim zawiera. Maksymalne skojarzenie definiuje się samo: jest to skojarzenie o największej możliwej wielkości. Rzecz jasna, będziemy chcieli szukać maksymalnych skojarzeń.

Przypomnijmy jeszcze, że *grafy dwudzielne* to takie, że ich wierzchołki można pokolorować na dwa kolory – dwa wierzchołki tego samego koloru nie mogą być połączone krawędzią. Możemy narysować taki graf dzieląc go na dwie części: z lewej możemy umieścić pierwszy kolor, a z prawej drugi. Tak też często będziemy

robić. Łatwo się przekonać, że warunek na dwudzielność grafu jest równoważny nieistnieniu cykli nieparzystych (co też się później przyda). Żeby przywyknąć do widoku skojarzeń w grafie dwudzielnym (bipartite matching), spójrzmy na poniższy obrazek z przykładowym skojarzeniem (zaznaczonym na czerwono).[1]



2.2 Skojarzenie doskonałe. Twierdzenie Halla

Jednak zanim przejdziemy dalej, warto poznać trochę matematycznej teorii związanej z tym problemem – czasem może się ona okazać kluczowa w znalezieniu rozwiązania zadania. Już teraz pierwszy raz zobaczymy przydatność grafów dwudzielnych, bo to właśnie w nich zachodzi **twierdzenie Halla** (znane też jako twierdzenie o kojarzeniu małżeństw).

Sugerując się jego drugą nazwą, zmieńmy trochę terminologię. Powiedzmy, że wierzchołki lewej strony grafu to *algorytmicy*, a prawej *zadanka*. Będziemy chcieli każdemu algorytmikowi dać jedno zadanie (ale każdy musi dostać inne, żeby się nie kłócili, kto pierwszy zrobił). Krawędzie oznaczają umiejętność zrobienia zadania przez algorytmika (nie ma sensu dawać mu zadania, którego nie umie zrobić, bo będzie mu smutno). Dany zbiór algorytmików *umie zrobić* zbiór wszystkich zadań, które umie zrobić dowolny z algorytmików mieszkających w zbiorze. Możemy już wprowadzić twierdzenie Halla:

Twierdzenie (Halla)**.** *Można dać zadanka wszystkim algorytmikom* ***wtedy i tylko wtedy****, gdy dla każdego podzbioru algorytmików, liczba algorytmików w podzbiorze jest nie mniejsza niż liczba zadań, które umieją zrobić.*

Na pierwszy rzut oka twierdzenie wygląda mało przydatnie – przecież nie będziemy sprawdzać wykładniczej liczby przypadków, żeby stwierdzić istnienie skojarzenia. Jednak, jak zwykle, siła takich twierdzeń tkwi w szczególnych przypadkach – gdy graf będzie bardzo specyficzny, twierdzenie może pomóc nam kompletnie przeformułować problem ze skojarzeń do analizy struktury grafu.

Poprawność tego twierdzenia jest dosyć oczywista w stronę warunku na wielkości podzbiorów – trochę trudniej jest je udowodnić w drugą stronę. Można skorzystać z indukcji, taki dowód jest np. [tutaj].

[1] Obrazek z Wikipedia Commons: `https://pl.m.wikipedia.org/wiki/Plik:Bipartite_graph_with_matching.svg`

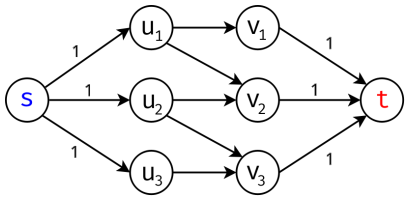
3 Algorytmy w grafach dwudzielnych

3.1 Czemu dwudzielne?

Można wymyślić kilka powodów – w grafach dwudzielnych naturalnie występuje asymetria, dzięki czemu bardzo łatwo wykorzystać koncept źródła i ujścia z przepływów, jak za chwilę zrobimy. Jednak rozważając ogólny przypadek, przydatny okaże się przede wszystkim brak cykli nieparzystych w grafach dwudzielnych.

3.2 Ścieżki powiększające w przepływie

Zaczniemy od zbudowania klasycznej sieci przepływowej rozwiązującej maksymalne skojarzenie. Na podstawie działania Forda-Fulkersona skonstruujemy uproszczony algorytm. Oznaczmy wierzchołki z lewej strony jako u_i, a z prawej v_j. Weźmy do sieci początkowy graf, a jego krawędzie skierujmy od lewej do prawej i nadajmy im **przepustowości** ∞ (ten wybór będzie miał znaczenie w przyszłości, ale działałby też wybór 1). Przejście jednostki przepływu przez parę wierzchołków oznacza, że krawędź jest w skojarzeniu. Aby doprowadzić przepływ, źródło łączymy z u_i, natomiast v_j z ujściem, za każdym razem z przepustowością 1.[2]



Zastanówmy się, jak działa metoda Forda-Fulkersona na takiej sieci. Początkowo będziemy po kolei brali proste ścieżki $s \to u_i \to v_j \to t$, natychmiast powiększając skojarzenie. Jednak w pewnym momencie będzie to niemożliwe: wtedy, mknąc po sieci rezydualnej, zaczniemy cofać się krawędziami w skojarzeniu. Rozumując dalej: zaczynając ścieżkę powiększającą od wierzchołka u_i (z założenia nieskojarzonego) możemy:

- Przejść do nieskojarzonego wierzchołka v_j i natychmiast powiększyć przepływ (skojarzenie).
- Przejść do skojarzonego wierzchołka v_j, skąd musimy wrócić się na lewo krawędzią w skojarzeniu (bo w sieci rezydualnej istnieje dla niej krawędź stowarzyszona w drugą stronę) do u_k, następnie do pewnego v_l nieskojarzoną krawędzią. Jeżeli v_l jest skojarzone, to powtarzamy proces, w innym wypadku możemy powiększyć przepływ.

Czas przekuć ten algorytm w coś szybszego i prostszego.

[2] Zmodyfikowany obrazek z Wikipedia Commons: `https://commons.wikimedia.org/wiki/File:Maximum_bipartite_matching_to_max_flow.svg`

3.3 Turbo matching

Powyższy schemat działania można przepisać na pseudokod:

> *Powiększ(u_i)* – spróbuj stworzyć ścieżkę powiększającą zaczynając od nieskojarzonego u_i:
>
> 1. Jeżeli jest nieskojarzony sąsiad v_j, to skojarz u_i i v_j ze sobą.
> 2. W przeciwnym wypadku: wybierz któregoś sąsiada v_k skojarzonego z u_l i sprawdź, czy da się zrobić *Powiększ(u_l)*. Jeżeli tak, to „rozkojarz" z u_l i zamiast tego skojarz ze sobą (u_i).

Zastanówmy się, jak usprawnić tę metodę:

- Nie ma sensu robić cykli: wielokrotnie odwiedzać tego samego wierzchołka. W przeciwnym wypadku możemy się zapętlić.
- Warto wiedzieć, czy skojarzenie udało się powiększyć w rekurencji, aby wcześniej przerwać działanie. *Powiększ* może zwrócić prawdę, jeżeli skojarzenie się powiększyło (czyli skończyło się na kroku 1.). Jeżeli doszło do kroku 2., to zwraca prawdę, gdy wywołanie rekurencyjne zwróciło prawdę. W przeciwnym wypadku zwraca fałsz. Skojarzenie w kroku 2. modyfikujemy tylko wtedy, gdy wywołanie rekurencyjne zwróciło prawdę.
- Nawiązując do poprzedniej modyfikacji, jeżeli nie udało się powiększyć skojarzenia w kroku 2., to możemy dalej z tego samego wierzchołka poprosić inny wierzchołek o próbę od-skojarzenia.
- Do tego zauważmy, że nie trzeba wcześniej modyfikować skojarzenia w kroku 2., bo można to zrobić dopiero wtedy, gdy jesteśmy pewni, że taka zmiana pozwala powiększyć skojarzenie. Funkcja może po prostu zakładać, że u_i jest nieskojarzony.

Pamiętajmy, że cała ta metoda działa dlatego, że tak na prawdę puszczamy algorytm przepływowy, a jednostkowe przepustowości i struktura grafu pozwalają nam na dużo uproszczeń.

Otrzymujemy bardzo prosty kod funkcji `bool augment(int u)`. `match[v]` oznacza obecny skojarzony wierzchołek `u`, przy czym `-1` to brak.

```
bool augment(int u)
{
    vis[u] = true;
    for(auto v : graph[u])
        if(match[v] == -1)
            { match[v] = u; match[u] = v; return true; }
    for(auto v : graph[u])
        if(not vis[match[v]] and augment(match[v]))
            { match[v] = u; match[u] = v; return true; }
    return false;
}
```

Funkcję `augment` wystarczy wywoływać w taki sam sposób, jak zrobiłby to Ford-Fulkerson – na wierzchołkach będących sąsiadami źródła w sieci rezydualnej, czyli nieskojarzonych. Trzeba też zawczasu czyścić tablicę odwiedzonych. Wtedy z każdym wywołaniem, które zwróciło prawdę, wielkość skojarzenia zwiększa się o 1. Voilà!

...no dobra, nie do końca. To, co pokazałem, to klasyczny bezimienny algorytm szukania ścieżek powiększających w celu znalezienia maksymalnego skojarzenia (w rosyjskich źródłach spotkałem się z nazwą „algorytm Kuhna"). **Turbo matching** to podobny algorytm, którego opisu trudno szukać, lecz wygląda na to, że jest to popularny polski trik. Jest to modyfikacja powyższego algorytmu, niezwykle prosta, bo różnica jest jedynie w pętli szukającej powiększeń: *tablica odwiedzonych powinna być czyszczona jedynie po sprawdzeniu wszystkich wierzchołków do próby skojarzenia.* Czyli reszta turbo matchingu wygląda tak:

```
while(true)
{
    for(int u = 0; u < n; u++)
        vis[u] = false;
    bool any = false;
    for(int u = 0; u < n; u++)
        if(match[u] == -1 and augment(u))
            any = true;
    if(not any) break;
}
```

Użyłem tutaj pewnej sztuczki: zauważmy, że cały dotychczasowy algorytm przechowywał dane w symetryczny sposób, niezależnie od tego, z której strony jest wierzchołek. Z tego powodu nie musimy nawet dwukolorować grafu.

3.3.1 Jak najszybciej

Ale co ze złożonością czasową? Cóż, nawiązując do Forda-Fulkersona albo prostego uzasadnienia, $O(V)$ razy powiększymy skojarzenie zmodyfikowanym przeszukiwaniem wgłąb w $O(V + E) = O(E)$ – teoretyczna złożoność to $O(VE)$. Ale w praktyce skojarzenia działają, podobnie jak przepływy, bardzo szybko – jeszcze szybciej w grafach szczególnej postaci.

W grafach, w których maksymalny stopień jest bardzo mały (2, 3...) efekt jest wyjątkowo duży. W zadaniu Cło z [I etapu XV OI] proste rozwiązanie wzorcowe działało w czasie $O(n + m)$ ($n \leqslant 10^5$, $m \leqslant 2 \cdot 10^5$). Jednak rozwiązanie korzystające z turbo matchingu wywołanym na grafie, którego wierzchołki miały stopień 2 z $|V| = O(n + m)$ wierzchołkami, mimo starań układającego testy działało **podobnie do rozwiązań wzorcowych**. Jednak cudów nie ma: wrzucenie dziesiątków milionów krawędzi do grafu raczej nie jest dobrym pomysłem.

Bardzo zabawne są też optymalizacje randomizowane do tego algorytmu. Losowanie kolejności odwiedzania wierzchołków cieszy się sporą popularnością wśród krnąbrnych uczestników. Poza tym, można też przed wykonaniem algo-

rytmu arbitralnie wybrać krawędzie do skojarzenia stosując różne zachłanne heurystyki.

Krążą legendy o ruskich testach uwalających turbo matching. Mało kto je widział, ale sama idea ich istnienia jest doprawdy przerażająca.

3.4 Algorytm Hopcrofta-Karpa

Na koniec historia o smutnym algorytmie, o którym wie większość osób znających skojarzenia, ale prawie nikt go nie pisze. Działa w czasie $O(E\sqrt{V})$.

Jego zasada działania jest dosyć znajoma: konstruowany jest graf poziomowy, a następnie wyszukuje się jak najwięcej rozłącznych krawędziowo ścieżek powiększających („skojarzenie blokujące”?). W istocie, podobieństwo nie jest przypadkowe: jest to „zdegenerowany” algorytm Dinica na sieci przepływowej takiej samej, jak wprowadziliśmy.

W praktyce turbo matching jest znacznie szybszy od algorytmu Hopcrofta-Karpa. Ale zawsze można się pochwalić, że ma się algorytm którego pesymistyczna złożoność jest $O(\sqrt{V})$ razy mniejsza.

4 Skojarzenia, zbiory niezależne i pokrycia

Sekcja jest zainspirowana pomocnym artykułem z czasopisma Delta „W grafach dwudzielnych jest łatwiej" autorstwa doktora Jakuba Radoszewskiego.

4.1 Niby co innego, a jednak to samo

Nauczymy się teraz, jak skojarzenia są związane z innymi wielkościami opisującymi strukturę grafu. Zacznijmy od ich wprowadzenia:

- Minimalne pokrycie krawędziowe $\mathcal{C}_E$ (Min Edge Cover) – najmniejszy zbiór krawędzi, taki, że każdy wierzchołek ma co najmniej jedną sąsiadującą krawędź w pokryciu.
- Minimalne pokrycie wierzchołkowe $\mathcal{C}_V$ (Min Vertex Cover) – najmniejszy zbiór wierzchołków, taki, że każda krawędź ma co najmniej jeden sąsiadujący wierzchołek w pokryciu.
- Maksymalny zbiór niezależny $\mathcal{I}_V$ (Max Independent Set) – największy zbiór wierzchołków, taki, że żadne dwa nie sąsiadują ze sobą.

Zastanówmy się jeszcze chwilę nad skojarzeniem (Max Matching) – w pewnym sensie jest to wariacja zbioru niezależnego, ale w definicji są krawędzie zamiast wierzchołków. Z tego powodu oznaczmy je $\mathcal{I}_E$.

Problemy wyglądają z natury podobnie, lecz z pewnością zaskakujący będzie fakt, że ich rozwiązania są ze sobą blisko powiązane. Poniżej są wypisane te relacje w grafie z n wierzchołkami, wraz z zarysem uzasadnienia (podane fakty wykorzystuje się, by pokazać nierówność $\leqslant$ w obie strony). Co ważne, zachodzą one tylko wtedy, gdy w grafie **nie występują wierzchołki izolowane** (o stopniu zerowym).

- $|\mathcal{C}_V|+|\mathcal{I}_V| = n$ – bo każdy zbiór niezależny to dopełnienie pewnego zbioru niezależnego, i na odwrót. Dzięki temu oba problemy można rozwiązać w ten sam sposób. Wierzchołki izolowane psują tę własność.
- $|\mathcal{C}_E| + |\mathcal{I}_E| = n$ – ponieważ pokrycie krawędziowe musi tworzyć las, a maksymalne skojarzenie można zachłannie uzupełnić krawędziami tak, żeby otrzymać pokrycie krawędziowe. Graf z wierzchołkami izolowanymi nie ma pokrycia krawędziowego. Warto wiedzieć, że to zachłanne uzupełnianie daje minimalne pokrycie krawędziowe.

Obie relacje łączą ze sobą problemy, których rozwiązania łatwo do siebie sprowadzić.

4.2 W grafach dwudzielnych jest łatwiej

Nie pozostaje nic innego, jak wprowadzić cudowną równość z **Twierdzenia Königa**, dzięki której w grafach dwudzielnych powyższe równości spinają się w piękną całość.

Twierdzenie (Königa)**.** *W grafie dwudzielnym wielkość maksymalnego skojarzenia jest równa wielkości minimalnego pokrycia wierzchołkowego.*

$$\mathcal{I}_E = \mathcal{C}_V$$

Dowód. Twierdzenie można udowodnić za pomocą Max-Flow Min-Cut i konstrukcji pokrycia z minimalnego przekroju sieci przepływowej. Dowód można znaleźć w materiałach uniwersytetu Chicago [tutaj]. □

Warto wiedzieć, że konstrukcja minimalnego pokrycia wierzchołkowego przebiega następującego: bierzemy maksymalny przepływ odpowiadający skojarzeniu, konstruujemy minimalny s-t przekrój (A, B), a następnie bierzemy do pokrycia wierzchołki z prawej strony (od ujścia) w A oraz z lewej w B. Wypada też zauważyć, że twierdzenie nie wymaga w założeniach braku wierzchołków izolowanych. Zaznaczę, że zwykle znajomość tych wszystkich metod konstrukcji nie jest konieczna.

Wnioskujemy, że w grafie dwudzielnym umiemy rozwiązać wszystkie powyższe problemy w czasie wielomianowym! W grafie ogólnym Min Vertex Cover tudzież Max Independent Set jest w NP (co przydaje się do redukcji, może kiedyś do tego wrócimy). Co więcej, możemy zrobić piękną sieć równości, bo $|\mathcal{C}_V| = |\mathcal{I}_E| \Rightarrow |\mathcal{I}_V| = |\mathcal{C}_E|$:

$$\begin{array}{ccccc} |\mathcal{C}_V| & + & |\mathcal{I}_V| & = & n \\ \shortparallel & & \shortparallel & & \\ |\mathcal{I}_E| & + & |\mathcal{C}_E| & = & n \end{array}$$

4.3 Uwaga o zbiorze dominującym

Często myli mi się Min Vertex Cover z Min Dominating Set – są to podobne problemy, z czego ten drugi *zawsze* jest NP (czyli dosyć trudny do rozwiązania). Pierwszy pokrywa krawędzie, a drugi pokrywa wierzchołki – zbiór dominujący to taki zbiór wierzchołków, że każdy wierzchołek sąsiaduje z wierzchołkiem w zbiorze, bądź sam w nim jest.

5 Zadanka

5.1 [PREOI 2020] Skracanie

Treść Dany jest słownik n krótkich słów (długości co najwyżej 10 znaków). Dla każdego z nich chcemy wybrać *skrót*, czyli słowo będące jego podciągiem (niekoniecznie spójnym). Skróty muszą być parami różne oraz mieć długość z przedziału $[1, 4]$.

Rozwiązanie Zadanie w dosyć oczywisty sposób sprowadza się do poszukiwania maksymalnych skojarzeń w grafie dwudzielnym. Każde słowo ma co najwyżej $S = \binom{10}{1} + \binom{10}{2} + \binom{10}{3} + \binom{10}{4} = 385$ możliwych skrótów, zatem możemy stworzyć graf z $|V| = |E| = O(nS)$. Daje to złożoność rzędu $O(nS\sqrt{nS})$, czyli liczba operacji rzędu $4 \cdot 10^7$. Graf jest dosyć specyficznej postaci, a jedna ze stron grafu jest bardzo mała (jedynie n wierzchołków), więc algorytm będzie szybki.

5.2 Ogród księżniczki

Treść Mamy kratkę (taką jak szachownica, ale pola nie są pokolorowane) $n \times n$. Na niektórych polach jest trawka, a na innych ziemia. Pola na których jest ziemia możemy zmienić w jeziorka, ale nie można zrobić dwóch sąsiadujących jeziorek (czyli o wspólnym boku). Chcemy zmaksymalizować sumaryczną liczbę stworzonych przez nas jeziorek.

Rozwiązanie Gdy spotkałem się z tym zadaniem, było ono przedstawione jak zadanie optymalizacyjne (czyli pojawiała się sugestia, że nie da się go zawsze szybko rozwiązać optymalnie). Dosyć łatwo się przekonać w bolesny sposób, że pomysły zachłanne raczej nie działają.

Sformułowanie z zakazem stawiania sąsiednich jeziorek i maksymalną liczbą jeziorek powinno przynieść na myśl problem Max Indepedent Set. W istocie, grafy zbudowane z relacji sąsiedztwa na kratce są dwudzielne (tutaj warto pomyśleć o szachownicach, żeby natychmiast zrozumieć dowód). Wystarczy wykorzystać algorytm maksymalnych skojarzeń, a wiemy, że $|\mathcal{I}_E| = |V| - |\mathcal{I}_V|$ (czyli wynik to liczba wierzchołków minus liczba krawędzi w maksymalnym skojarzeniu). W rozważanym grafie umieszczamy jedynie pola z ziemią. W tym konkretnym zadaniu należało odtworzyć wynik, zarys sposobu zrobienia tego był już przedstawiony (liczymy pokrycie wierzchołkowe z minimalnego przekroju, a potem bierzemy dopełnienie tego zbioru).

5.3 [XVI OI] Łyżwy

Treść Prowadzisz wypożyczalnię łyżew. Rozmiarów łyżew jest n i są ponumerowane liczbami całkowitymi od 1 do n (zgodnie z rosnącą wielkością). Łyżwiarze mogą założyć tylko łyżwy o rozmiarze z pewnego przedziału: każdy łyżwiarz ma rozmiar stopy $r \in \mathbb{N}$, i wtedy może nosić łyżwy z przedziału $[r, r + d]$, gdzie $d \in \mathbb{N}$ to wspólna dla wszystkich łyżwiarzy stała tolerancji. Przy tym łyżwiarz

zakłada obie łyżwy tej samej wielkości, bo inaczej będzie mu niewygodnie. Masz k par łyżew każdego rozmiaru. Chcesz napisać system, który obsługuje wydarzenia postaci „liczba łyżwiarzy z danym r zmieniła się o $x \in \mathbb{Z}$" i stwierdza, czy da się wydać łyżwy każdemu łyżwiarzowi.

Ograniczenia $n \leqslant 2 \cdot 10^5$, $m \leqslant 5 \cdot 10^5$, $k, x \leqslant 10^9$

Rozwiązanie Od razu widać, że mamy do czynienia ze specyficzną modyfikacją dynamicznego problemu stwierdzania, czy w grafie dwudzielnym istnieje skojarzenie doskonałe (każdy łyżwiarz dostaje parę łyżew). Teraz będzie trochę z innej beczki, bo skorzystamy z twierdzenia Halla (w końcu limity zdecydowanie nie pozwalają na ciągłe puszczanie algorytmu do skojarzeń/przepływów, a graf jest *bardzo* specyficzny).

W tym wypadku pan Hall mówi, że gdy spojrzymy na dowolny podzbiór łyżwiarzy A, to musi być ich co najwyżej tyle, ile łyżew, które może założyć dowolny z nich (oznaczmy tę liczbę przez $f(A)$). Zatem chcemy, by zachodziło $F(A) = f(A) - |A| \geqslant 0$. Żeby ograniczyć liczbę rozważanych podzbiorów A, pokuśmy się na parę obserwacji.

- Zauważmy najpierw, że A powinno zawierać wszystkich łyżwiarzy o rozmiarach stopy należących do pewnego zbioru – jeżeli nie byłoby jakiegoś łyżwiarza, który ma taki r jak ktoś w A, to jego dodanie do A jedynie utrudni spełnienie nierówności (zmniejszy $F(A)$).

- Zauważmy też, że A nie powinno mieć małych „dziur" – gdy w A są łyżwiarze o rozmiarach stopy r i $R \leqslant r + d + 1$, to dodanie wszystkich rozmiarów stopy z $[r, R]$ jedynie pogorszy wynik, bo $f(A)$ nie zmieni się.

- Natomiast gdy A ma dziurę większą, to również się okazuje, że nie warto go rozpatrywać. Załóżmy, że A nie spełnia warunku z twierdzenia ($F(A) < 0$) i da się je podzielić na rozłączne zbiory A_1 i A_2, tak, że łyżwiarze z A_2 mają rozmiary stopy większe o co najmniej $d + 1$ od tych z A_1. Nie ma pary łyżew, którą mogą założyć łyżwiarze z obu zbiorów, więc $F(A) = F(A_1) + F(A_2)$. Jeżeli ta wartość jest ujemna, to w szczególności jedna z wartości w nawiasach jest ujemna i warunek łamie A_1 lub A_2.

Z obserwacji wnioskujemy, że wystarczy rozpatrywać jedynie rozmiary stóp ze spójnych przedziałów. Reszta rozwiązania sprowadza się do dynamicznego problemu poszukiwania spójnego podciągu o minimalnej sumie, co jest poza zakresem tego kółka – można to osiągnąć drzewem przedziałowym. Wskazówka dla dociekliwego czytelnika: zastanów się, jak rozwiązać spójny podciąg o minimalnej sumie za pomocą dziel i zwyciężaj. Co się zmienia, gdy jeden z liści zmienia swoją wartość?

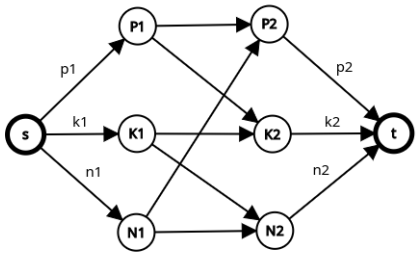
5.4 Papier, kamień, nożyce

Na koniec zabawkowe zadanie, które zdecydowanie da się rozwiązać prościej, ale warto pomyśleć o rozwiązaniu w ten sposób, bo daje nam to pewną perspektywę.

Treść Wiadomo, że w czasie pewnej gry w „Papier, kamień, nożyce" gracz pierwszy zagrał papier, kamień i nożyce odpowiednio p_1, k_1, n_1 razy, a drugi p_2, k_2, n_2. Ile minimalnie razy mógł przegrać pierwszy z graczy (strategia graczy mogła być dowolna)? Liczba zagranych rund jest rzędu 10^9.

Rozwiązanie Minimalizacja liczby przegranych to maksymalizacja liczby wygranych plus remisów. Wystarczy jedynie pamiętać, że remisujemy/wygrywamy, gdy gracz pierwszy zagra papier kontra papier/kamień, kamień kontra kamień/nożyce albo nożyce kontra nożyce/papier. Można tutaj pozwolić sobie na zbudowanie odpowiedniego grafu i uruchomienie maksymalnych skojarzeń w grafie dwudzielnym, lecz niestety liczba zagranych runda mogła być bardzo duża i nie zmieścimy się pamięciowo.

Z pomocą przychodzą przepływy – weźmy klasyczną sieć przepływową rozwiązującą maksymalne skojarzenie w grafie dwudzielnym (tą, w której przepustowości na środku to ∞). Zauważmy, że nasza sieć jest nieciekawa – bardzo dużo wierzchołków ma dokładnie takie same zestawy krawędzi. Zastosujemy pewien trik: aby pozbyć się redundancji będziemy łączyć podobne wierzchołki. Dokładniej: gdy para wierzchołków u_1, u_2 ma dokładnie takie same krawędzie wchodzące oraz wychodzące, to gdy je zwiniemy w jeden wierzchołek u (przepustowości się sumują), to wartość maksymalnego przepływu nie zmieni się. Gdy zrobimy to dostatecznie dużo razy na naszej sieci, otrzymamy coś takiego:



Maksymalny przepływ w takiej sieci to maksymalna liczba rund nieprzegranych przez gracza pierwszego. Komentarz do tego zadania jest taki, że na tak małej sieci możemy próbować uprościć algorytm, by nie trzeba było pisać przepływów – a takie rozwiązanie przychodzi szybciej, jeżeli nie wpadnie się na zachłanne. Mówię z doświadczenia (niekoniecznie jest to taktyka optymalna, ale jest lepsza niż brak taktyki).

5.5 Myślenie skojarzeniami

Powróćmy na chwilę do zadań, które rozwiązywaliśmy przy przepływach. Pełne omówienia są na kartce Przepływy.

5.5.1 [AMPPZ 2011] Iloraz inteligencji – biklika

Rozpatrzmy trochę inną odmianę problemu z tego zadania. Chcieliśmy tam znaleźć podzbiór wierzchołków w grafie dwudzielnym, że wierzchołki z różnych stron grafu są wszystkie połączone krawędziami, przy czym chcieliśmy również wybrać zbiór o maksymalnej sumie wag. Tutaj zastanówmy się tylko nad zbiorem maksymalnej wielkości. W istocie, biklika sprowadza się do znalezienia maksymalnego zbioru niezależnego w dopełnieniu grafu (bez dodawania krawędzi z tej samej strony). Zatem tę uproszczoną wersję można rozwiązać z pomocą Twierdzenia Königa i odpowiedniej konstrukcji.

5.5.2 [XVII OI, etap III] Mosty – zadanie na... skojarzenia?

W tym zadaniu na koniec mamy sieć, która uderzająco przypomina sieć przepływową rozwiązującą maksymalne skojarzenie. W istocie, pewnie łatwiej to rozwiązanie rozwiązać myśląc właśnie o skojarzeniach, ale umieściłem to zadanie na tamtej kartce, bo Mosty są znane jako zadanie na przepływy. Opisana sieć jest „po przejściu" trikiem ze zwijaniem wierzchołków, który jest opisany powyżej. Możemy ten proces odwrócić – wbrew pozorom liczba powstałych wierzchołków nie będzie zbyt duża, bo jest ograniczona przez sumę stopni, czyli liczbę krawędzi (ale nie zawsze tak będzie, i czasem trik będzie konieczny). Napisanie skojarzeń jest znacznie prostsze od przepływów, więc warto uważać, czy nie strzelamy ze zbyt dużych dział.

Warto zwrócić uwagę na ten zabieg: przepływy w zadaniach na skojarzenia zwykle występują, gdy chcemy duplikować wierzchołki o takich samych zestawach krawędzi.

6 Skojarzenia w innych środowiskach

Skojarzenia możemy spotkać nie tylko w grafach dwudzielnych. Jako że niewiele mogę powiedzieć o bardziej zaawansowanych algorytmach związanych ze skojarzeniami, to przedstawiam zarys ich działania oraz ich zastosowań, by zaspokoić ciekawość czytelnika.

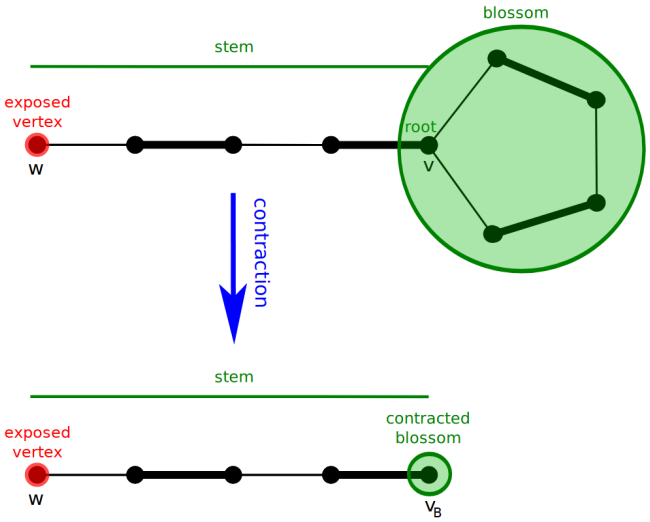
6.1 W ważonym grafie dwudzielnym

- Algorytm nosi nazwę **algorytmu węgierskiego** (Hungarian), bądź algorytmu Kuhna-Munkresa.
- Rozwiązuje problem przyporządkowania, który jest wariacją maksymalnego skojarzenia w grafie dwudzielnym, ale wybierane krawędzie mają wagi. Chcemy zmaksymalizować sumę wag w skojarzeniu danej wielkości (albo po prostu maksymalnej możliwej).
- Algorytm węgierski działa w $O(n^3)$. Problem przyporządkowania (assignment problem) można rozwiązać Max-Flow Min-Cost w $O(n^4)$ lub $O(n^3 \log n)$ (ćwiczenie dla czytelnika!), lecz będzie to zauważalnie wolniejsze od węgierskiego.
- Istnieje sformułowanie macierzowe, które jest w praktyce wolniejsze, ale znacznie łatwiejsze do zrozumienia i gwarantuje $O(n^3)$. Sformułowanie grafowe jest bardziej wydajne, ale korzysta z modyfikacji klasycznego algorytmu skojarzeń, która szukając ścieżek powiększających generuje tzw. „feasible labeling". Jest też trudniejsze do zrozumienia.
- Czytaj więcej: `http://timroughgarden.org/w16/l/l5.pdf`, `https://brilliant.org/wiki/matching-algorithms/`
- Przykładowe zastosowanie: mając dane dwie listy punktów w przestrzeni, które reprezentują np. położenia detektorów w dwóch bliskich momentach czasu spróbuj tak sparować punkty, że średnia odległość między punktami w parze jest jak najmniejsza. Uzasadnienie jest takie, że możemy nie wiedzieć, który detektor jest który, a chcemy śledzić ruch całego obiektu, do którego przyczepione są detektory. Poza tym klasyczny problem przyporządkowania mówi o przydzielaniu prac pracownikom tak, żeby ich sumaryczna cena była jak najmniejsza.

6.2 W dowolnym grafie

6.2.1 Znowu scieżki powiększające. Lemat Berge

- **Algorytm Blossom** (kielichowy) Edmondsa pozwala na obliczanie maksymalnego skojarzenia w grafie ogólnym.
- Korzysta z Lematu Berge'a, który mówi, że w dowolnym grafie skojarzenie jest maksymalne wtedy i tylko wtedy, gdy nie istnieje maksymalna ścieżka powiększająca. Możemy go udowodnić rozważając różnicę symetryczną (`xor`) między pewnym maksymalnym skojarzeniem i obecnym skojarzeniem, korzystając z tego, że tworzy ona pewien graf, w którym wszystkie stopnie są równe co najwyżej 2. Takie grafy są bardzo specyficznej postaci i wystarczy rozpatrzyć parę przypadków.
- Szukanie ścieżek powiększających jest o wiele łatwiejsze, gdy w grafie nie ma cykli nieparzystych (które nie istnieją właśnie w grafach dwudzielnych). Wynika to z faktu, że gdy takie cykle istnieją, to nie da się wprost określić, czy zastosowanie ścieżki powiększającej w przyszłości nie okaże się nieopłacalne.
- Algorytm kielichowy rozwiązuje ten problem „zwijając" (kontrakcja) cykle nieparzyste w wierzchołki. Cykl nieparzysty to kielich, a ścieżka do niego prowadząca to łodyga. Na koniec kielichy są rozwijane i wtedy określane jest, jak powinno wyglądać na nich skojarzenie.
- Czytaj więcej: `https://en.wikipedia.org/wiki/Blossom_algorithm` (stąd obrazek), `https://brilliant.org/wiki/blossom-algorithm/`



6.3 Niespodziewana algebra liniowa

- **Algorytm Muchy-Sankowskiego** posługuje się algebrą liniową, by w kompletnie inny sposób rozwiązywać problem maksymalnego skojarzenia w grafie ogólnym.
- Konstruuje rozszerzoną macierz sąsiedztwa M, której elementy są mod p, gdzie p jest dużą liczbą pierwszą. Początkowo wszystkie elementy są zerowe. Następnie, dla każdej krawędzi $\{u, v\}$ losowana jest liczba x, i ustawiamy $A_{u,v} = x$ oraz $A_{v,u} = -x$.
- Rząd skonstruowanej macierzy to dwukrotność wielkości maksymalnego skojarzenia (działa to w dowolnym grafie). Co za tym idzie, skojarzenie doskonałe istnieje wtedy i tylko wtedy, gdy wyznacznik macierzy jest niezerowy. Obie te wartości można policzyć w oczekiwanym czasie $O(n^3)$.
- Algorytm jest w stanie odtworzyć maksymalne skojarzenie.
- Czytaj więcej: `https://www.mimuw.edu.pl/~mucha/pub/mucha_sankowski_focs04.pdf`

7 *Zadanka do kminienia*

- [KI] Taxi
- Kartka Marka Sommera: `https://mareksom.w.staszic.waw.pl/kolko/2013-2014/kartki/27.05.2014.pdf`