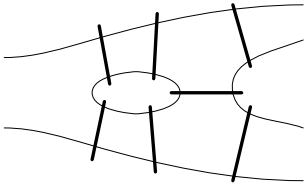
Eulerovské tahy

Historický problém

V roce 1735 se švýcarskému matematikovi Leonhardu Eulerovi na stůl dostal na první pohled jednoduchý problém, který mu předložil starosta města Královec (dnešní Kaliningrad). Královcem teče řeka Pregola, na ní je několik ostrovů a ostrovy jsou spojeny se zbytkem města mosty. Dobová ilustrace situaci vystihla takto (schématická kresba):



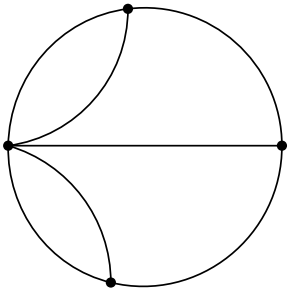
Pan starosta se pana matematika v dopise tázal, jestli je možné začít na některém z břehů (nebo ostrovů) a udělat si vycházku po městě tak, že se každým mostem projde právě jednou. Navíc chtěl procházku skončit na kusu suché země, ze kterého vyšel.

Profesor Euler jej nejprve chtěl poslat k šípku – problém jde snadno vyřešit rozborem případů, což by zvládli i tehdejší studenti střední školy (natož pak ti dnešní). Zachoval se ovšem jako pravý matematik – přišel na to, jak problém zobecnit, a mistrně vyřešil hádanku i pro všechna možná města, která kdy budou chtít pořádat podobné procházky.

Eulerovský tah

Pojďme si nyní problém popsat abstraktně a tím si připomenout grafovou terminologii. Vrcholy našeho grafu jsou kusy pevniny, ať už to budou části města nebo ostrovy. Mezi dvěma vrcholy povede hrana, pokud jsou spojeny mostem, a onen most odpovídá hraně.

V tomto zadání má smysl uvážit, že mezi dvěma kusy pevniny povede mostů více – například v Praze jich vede tolik, že se na to ptají v leckteré zeměpisné olympiádě. Graf, kde mezi vrcholy vede více hran, nazýváme *multigraf,* a pokud dvě hrany vedou mezi stejnými vrcholy, mluvíme o nich jako o *paralelních* hranách.



Obecná procházka v grafu z vrcholu A do vrcholu B (posloupnost hran taková, že cílový vrchol předchozí hrany je počáteční vrchol hrany následující) se nazývá *sled* z A do B. Ve sledu se mohou opakovat jak hrany, tak vrcholy; sled tedy není řešením našeho problému (ve sledu je možné se vrátit po hraně, ze které jsme právě přišli).

Pro naši úlohu se hodí posloupnost hran taková, že vrcholy se opakovat mohou, ale hrany nikoli. Této posloupnosti se říká *tah* z A do B. Kdyby se neopakovaly ani vrcholy, pak posloupnost označujeme jako *cestu.* Tah (respektive sled) je *uzavřený,* pokud začíná v A a končí také v A.

Podíváme-li se tedy na mapu Královce jako na multigraf, ptáme se, zdali existuje uzavřený tah takový, že každou hranu navštíví právě jednou. Takovému tahu pak říkáme *uzavřený eulerovský*.

Mimochodem, tahu se „tah“ neříká jen tak náhodou. Děti se často ve školce překonávají v umění nakreslit obrázek jedním tahem, aby se tužkou nemuselo vracet po už nakreslené čáře. Pokud si obrázek představíme jako graf (čáry jsou hrany, místa jejich setkání vrcholy), pak eulerovský tah nalezneme jen v tom obrázku, který lze nakreslit jedním tahem. V uzavřeném eulerovském tahu se pak vrátíme i do místa, kde jsme začali.

Podmínky tahu

Je na čase poodhalit řešení našeho problému s eulerovským tahem. Půjdeme na to jako matematici – nejprve ukážeme *nutnou* a hned nato *postačující* podmínku. Nutná vlastnost grafu je taková, že bez ní eulerovský tah není možné najít; postačující vlastnost je ta, se kterou vždy eulerovský tah najít umíme. Jsou-li obě podmínky stejné, pak se jedná o ekvivalenci, a tak tomu bude i nyní.

Představme si, že jsme kouzlem nějaký uzavřený eulerovský tah našli, ať už je jakýkoli. Vždy, když se dostaneme do jednoho vrcholu (a není důležité, jestli už jsme v něm byli, nebo ne), tak z něj musíme hned také odejít, abychom tah uzavřeli. A protože tah je eulerovský, každou hranou projdeme jen jednou, takže tyto dvě hrany (tu příchozí a odchozí) už nepoužijeme. U každého vrcholu mimo výchozí tedy platí, že hrany tvoří dvojice – jedna, co vedla dovnitř, a jedna, která z něj vedla ven.

Podobná věc platí i pro startovní vrchol. Sice do něj nevstoupíme poprvé pomocí hrany, takže počet navštívených hran u něj bude stále lichý – ale jen do chvíle, než se do něj naposledy vrátíme a skončíme, protože skončením jsme použili poslední hranu, která bude tvořit dvojici s hranou první.

Jakou vlastnost grafu jsme odhalili? Neplatí, že graf má sudý počet hran (protože trojúhelník jedním tahem nakreslíme a přesto má 3 hrany), ale platí, že do každého vrcholu vede sudý počet hran, tedy že graf má *všechny stupně sudé*. Nezapomeňme také na to, že graf musí být souvislý – dva oddělené obrázky jedním tahem bez zvednutí tužky nenakreslíme. Máme nutné podmínky!

Nalezení tahu

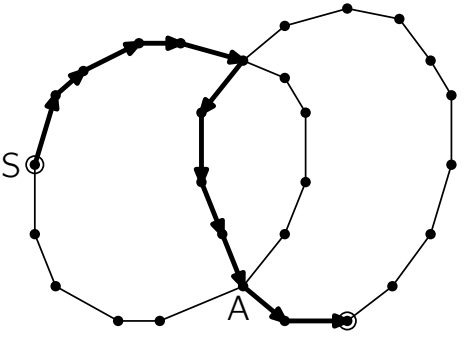
Zbývá tedy ověřit, že podmínky jsou i postačující. Mějme souvislý graf, který má všechny stupně sudé. Umíme v něm vždy najít uzavřený eulerovský tah? Ověřme to, jak se na informatiky patří – algoritmem.

Předložený algoritmus je založený na vylepšeném prohledáváni do hloubky, tedy DFS, o kterém jste si mohli přečíst v první grafové kuchařce.

Vyberme si vrchol, v něm začneme. Náš algoritmus musí umět označovat hrany jako „probrané“, jako to dělá DFS. Vyberme si tedy jednu hranu a pokračujme dále, zatím bez vypisování.

Po nějakém tom procházení se jistě stane, že jsme se zastavili – vrchol už nemá žádné nepoužité hrany. Nutně to znamená, že to je ten vrchol, ve kterém jsme začínali.

V procházení do hloubky se vracíme zpět, ale my k tomu přidáme vypisování cesty – postupně pozpátku vypisujeme hrany, kterými se vracíme zpět v prohledávání.



Na obrázku výše je příklad právě probíhajícího algoritmu. Začal ve zvýrazněném vrcholu vlevo, procházel po šipkách až do bodu A, kde volil hrany tak, že hned skončil na začátku. Dále pokračoval vypisováním hran pozpátku, až došel zase do bodu A. Zde si vybral jednu ještě nepoužitou hranu a po ní prošel celou druhou kružnici – zbytek hran – zpět do bodu A. Nyní vypisuje hrany pozpátku od bodu A.

Buď tímto výpisem dojdeme až na začátek, nebo se dostaneme do vrcholu, který má ještě nějaké nepoužité hrany (situace může vypadat třeba jako na obrázku). Potom vypisování zastavíme a pokračujeme v prohledávání DFS přes nepoužitou hranu. I tam se to může zastavit (a zastaví), i tam začneme vypisovat pozpátku. Nakonec dojdeme do původního místa rozbočení, a budeme opět pozpátku vypisovat hrany, které nás nakonec dostanou až na počátek, kde skončíme.

Najde tento algoritmus opravdu korektní uzavřený eulerovský tah? Graf byl souvislý a o algoritmu DFS se ví, že v takovém případě navštíví každou hranu právě jednou. Algoritmus opravdu vypisuje cyklus – jen je u něj trochu zvláštní způsob, jak ho vypisuje. Když dojde na křižovatku s ještě nepoužitými hranami, tak výpis zastaví, tiše po nich kráčí, označuje si je a vypisuje, až když se po nich vrací. Ověřme si, že hrany opravdu navazují.

V duchu argumentů z předcházející části víme, že jediný vrchol grafu s lichým počtem nepoužitých hran je právě ona křižovatka – a algoritmus DFS prochází graf podobně, jako jsme ho procházeli v minulé sekci, takže právě do tohoto vrcholu algoritmus dojde, až se průchod touto částí grafu zastaví.

Jakmile sem program dojde (a nezbudou mu volné hrany), začne cestovat zpět a hrany vypisovat – a opravdu, pokračuje se tedy z místa, kde naposledy přestal, a program vskutku vypíše tah přes všechny hrany v grafu – uzavřený eulerovský tah.

Věta o eulerovském tahu v celé své kráse tedy zní: *(Multi)graf obsahuje uzavřený eulerovský tah právě tehdy, když má všechny stupně sudé a je souvislý.*

Je třeba podotknout, že složitost našeho algoritmu na bázi DFS je lineární vůči velikosti grafu (počtu vrcholů a hran). Existují i jiné algoritmy pro hledání eulerovského tahu, jedna varianta například prochází grafem a vybírá si na křižovatkách takové

hrany, které souvislost grafu pokud možno nepoškodí. Tyto algoritmy už nemusí mít nutně lineární časovou složitost.

Jiné druhy procházek

Nejen kreslením obrázků ze stejného bodu živ je člověk. Co kdybychom mohli začít a skončit v jiném místě, tedy ptali se po neuzavřených eulerovských tazích, změnilo by se něco? Není tomu tak, pouze nutné a postačující podmínky si vyžádají, aby všechny vrcholy měly sudý stupeň až na právě dva vrcholy, které mají lichý stupeň. Pokud nám to nevěříte, zkuste si to rozmyslet sami, opravdu to není těžké.

Smysl také dává zkusit najít ne uzavřený tah, ale uzavřenou cestu – uzavřenou cestu přes všechny vrcholy, která navštíví každý vrchol právě jednou (říká se jí „Hamiltonovská cesta"). Bohužel, ačkoli jsou problémy příbuzné, musíme vás zklamat – není znám žádný efektivní (polynomiální) algoritmus na tento problém, a kdyby jej někdo z vás nalezl, vyřešil by otázku „P vs. NP", o níž se více dočtete v kuchařce o těžkých problémech.

V matematice se také někdy zmiňují „náhodné procházky" po grafech – můžete si je představit tak, že se po mostech města Královce motá opilec, který si hází (opilou nebo spravedlivou) mincí a podle toho se rozhoduje, přes který most jít dál. Použití mají tyto modely hlavně v matematické teorii grafů a teorii pravděpodobnosti. O tom si můžeme povědět zase někdy jindy.

Martin Böhm

Úloha 23-2-3: Projížďka

Turing před projížďkou na kole pečlivě prozkoumal terén, který si reprezentoval jako seznam rozcestí a cest mezi nimi. Silnice jsou však nevyrovnané – některé jsou krátké a klidné, dokonce vedou z kopce; jiné jsou dlouhé, klikaté a strmé, takže velmi unavují. Proto každé z nich přidělil celé číslo vyjadřující tuhle subjektivní obtížnost – na kladně ohodnocených cestách si bude odpočívat a na záporně ohodnocených tuhle nashromážděnou energii vydá.

Teď by od vás chtěl, abyste napsali program, který mu najde takovou cestu (včetně začátku – a může začínat na libovolném rozcestí), po které jednak projede všechny silnice *právě jednou*, druhak bude na každém rozcestí součet všech Turingem do té chvíle projetých silnic *nezáporný* a navíc se vrátí na rozcestí, na kterém začal. Chcete-li a myslíte-li si, že vám to pomůže, předpokládejte klidně, že z každého rozcestí vychází sudý počet silnic.

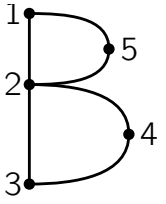
Úloha 23-3-4: Psaní písmen

Každé písmeno se skládá z bodů a linií, které je spojují. V jednom bodě může začínat i končit více linií. Při psaní perem lze psát víc navazujících linií jedním tahem, nejde-li to, musí se pero zvednout a začít jinde. Kolikrát nejméně je potřeba pero zvednout?

Na vstupu dostanete neorientovaný graf o N vrcholech a M hranách a vypište, kolika nejméně tahy lze nakreslit.

Samozřejmě šetříme, takže je zakázáno jakoukoli hranu nakreslit více než jednou. Jinak řečeno, nesmíte se vracet po již nakreslených liniích.

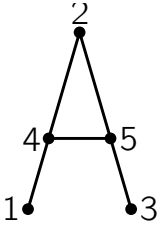
Příklad vstupu:




```
5 6
1 2
2 3
3 4
4 2
2 5
5 1
```

výstup: `1`

Jiný příklad:




```
5 5
1 4
4 2
2 5
5 3
4 5
```

výstup: `2`