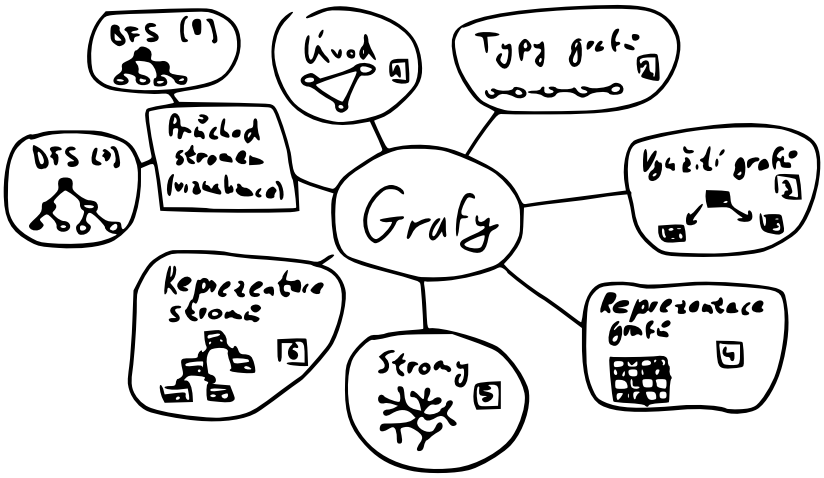
EKSI

Korespondenční Seminář z Informatiky

Grafy

První část úvodníku bude tentokrát ve formě několika krátkých videí, jejichž přehled vidíte na následujícím grafu:



Rozcestník k videím naleznete na webové stránce:
`http://effaacademy.cz/informatika`
Jedná se o následující videa:

- Úvod do teorie grafů[1]
- Typy grafů[2]
- Využití grafů[3]
- Reprezentace grafů v paměti[4]
- Stromy[5]
- Reprezentace stromů v paměti[6]
- Průchod stromu do hloubky (DFS) - vizualizace[7]
- Průchod stromu do šířky (BFS) - vizualizace[8]

Pojmy

Graf – Dvojice skládající se z množiny vrcholů (označenou V) a množiny hran (označenou E), kde každá hrana spojuje dva vrcholy. Pro účely zadání vede z jednoho vrcholu do druhého vždy buď jedna, nebo žádná hrana, zejména jich tedy nikdy nemůže vést více. Můžete také počítat s tím, že hrana nemůže vést z vrcholu x zpět do vrcholu x (tedy graf neobsahuje smyčky).

[1] `http://www.youtube.com/watch?v=cZdlDgDhm70`
[2] `http://www.youtube.com/watch?v=2FLi8dV7DO8`
[3] `http://www.youtube.com/watch?v=Dp9mD2QPO_g`
[4] `http://www.youtube.com/watch?v=02vfx_crXSw`
[5] `http://www.youtube.com/watch?v=6kYYOc7VJw0`
[6] `http://www.youtube.com/watch?v=rZTQQbguPLw`
[7] `http://www.youtube.com/watch?v=0uz3Nj8ZnUY`
[8] `http://www.youtube.com/watch?v=a6Mx8vhHvS4`

Neorientovaný graf – Hrany spojují své koncové vrcholy a není určeno, který vrchol je začátek hrany a který konec. Důležité je, že jsou dva vrcholy spojeny hranou. Hrana z A do B je totéž jako hrana z B do A.

Orientovaný graf – Hrany jsou šipky z jednoho vrcholu do druhého. Hrana z A do B je jiná hrana než hrana z B do A.

Sousední vrcholy v grafu – Jsou spojeny hranou.

Podgraf – Obdoba pojmu podmnožina. Podgraf vznikne vymazáním některých vrcholů původního grafu, všech hran do těchto vrcholů zasahujících a případně některých dalších hran.

Ohodnocený graf – Graf, u něhož jsou hrany označeny nějakou informací, typicky číslem (např. vzdálenost mezi městy).

Cesta délky n – n vrcholů, které jsou uspořádány za sebou, z každého vrcholu kromě posledního vede hrana do následujícího vrcholu (celkem $(n-1)$ hran).

Stupeň vrcholu – Počet hran, které z vrcholu vychází.

Úplný graf – Každý z vrcholů tohoto grafu je spojen hranami se všemi ostatními vrcholy.

Cyklus (kružnice) – Skoro jako cesta, s tím rozdílem, že první vrchol je stejný, jako poslední vrchol. Nejmenší kružnicí je trojúhelník, tedy úplný graf o třech vrcholech.

Acyklický graf – Neobsahuje žádný cyklus.

Souvislý graf – Takový graf, v němž platí, že pro každé dva vrcholy x, y existuje alespoň jedna cesta z x do y.

Strom – Souvislý graf neobsahující cykly.

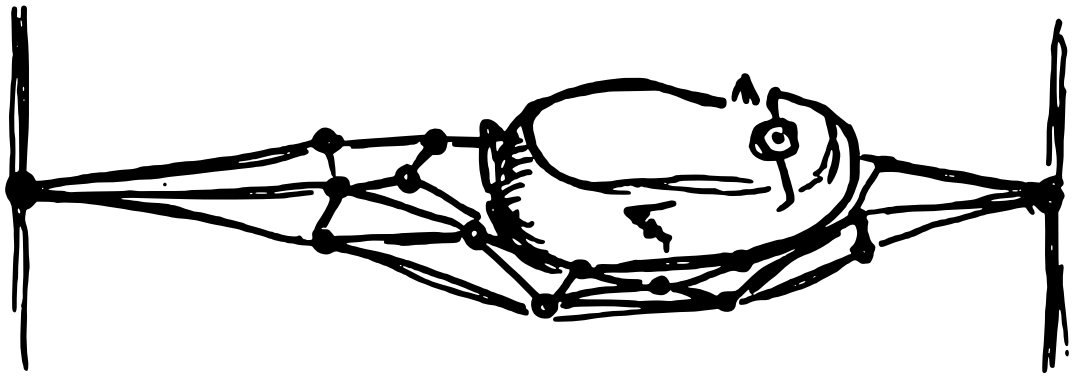
Matice sousednosti – Tabulka vyjadřující, které vrcholy spolu sousedí. Konkrétně buňka $M_{i,j}$ obsahuje hodnotu `true` právě tehdy, když je hrana mezi vrcholem i a j. Pokud reprezentujeme ohodnocený graf, buňky místo hodnot `true` obsahují hodnoty příslušných hran.

Seznam sousednosti – Obsahuje pro každý vrchol seznam jeho sousedů.

Vzdálenost dvou vrcholů – Délka nejkratší cesty mezi dvěma vrcholy.

Matice vzdáleností – Tabulka vyjadřující vzdálenost mezi každými dvěma vrcholy. Konkrétně buňka $M_{i,j}$ obsahuje vzdálenost mezi vrcholy i a j.

Excentricita vrcholu – Nejdelší vzdálenost z daného vrcholu do libovolného jiného vrcholu, tedy pro vrchol i je to nejvyšší číslo z i-tého řádku matice vzdáleností.



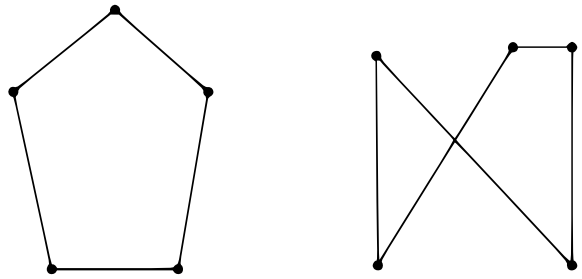
Průměr grafu – Nejdelší vzdálenost mezi libovolnými dvěma vrcholy grafu, tedy největší číslo v matici vzdáleností (zároveň také největší excentricita).

Poloměr grafu – Nejmenší ze všech excentricit. Povšimněte si, že u grafových kružnic je poloměr stejný jako průměr - pokud má kružnice n vrcholů, pak mezi libovolnými dvěma vrcholy existuje cesta délky nejvýše $n/2-1$.

Chromatické číslo – Představte si, že každý vrchol grafu může mít nějakou barvu. Chromatické číslo potom udává, kolik nejméně různých barev potřebujeme na obarvení všech vrcholů tak,

aby spolu nesousedily dva vrcholy stejné barvy. Například pro kružnici délky 3 potřebujeme tři barvy, ovšem pro kružnici délky 4 nám stačí jen 2.

Izomorfismus – Dva grafy $G = (V, E)$ a $G' = (V', E')$ nazýváme izomorfní, jestliže existuje bijektivní (vzájemně jednoznačné) zobrazení $f : V \to V'$ tak, že platí: $\{x, y\}$ je prvkem E, právě když $\{f(x), f(y)\}$ je prvkem E'. To jednoduše řečeno znamená, že těmto dvěma grafům dokážeme vrcholy pojmenovat tak, že mají stejnou matici sousednosti i vzdálenosti. Řečeno ještě jednodušeji, můžeme jejich vrcholy poposunovat tak, že oba grafy vypadají stejně (viz obrázek). Všimněte si, že izomorfní grafy mají shodné například počet vrcholů, chromatické číslo, průměr, poloměr a spoustu dalších vlastností.



Obrázek 1: Izomorfní grafy

Procházení grafu

Představme si, že máme graf a potřebujeme najít nejkratší cestu z vrcholu A do vrcholu B. Nebo chceme jen zjistit, jestli mezi těmito vrcholy nějaký cesta existuje. Nebo chceme zjistit, jestli je daný graf stromem, jinými slovy, jestli obsahuje nějaký cyklus. Ve všech případech musíme jakýmsi způsobem procházet grafem, abychom našli odpověď na naše otázky. Představíme si proto dva standardní způsoby procházení, které nám na většinu problémů vystačí. Jedná se o *procházení do hloubky*, aneb DFS (Depth-first Search) a *procházení do šířky*, aneb BFS (Breadth-first Search).

Obě metody jsou v jádru podobné a jejich myšlenka je velice jednoduchá. V obou případech používáme jakousi pomocnou datovou strukturu pro ukládání vrcholů pro pozdější zpracování. Na začátku si to této struktury vložíme jeden, daný počáteční vrchol a dále pokračujeme jednoduchým cyklem, jak je znázorněno v následujícím pseudokódu:

```
pruchod(vrchol v) {
    vlož v~do datové struktury
    while (struktura není prázdná) {
        označ x za navštívený
        vlož nenavštívené sousedy x do struktury
    }
}
```

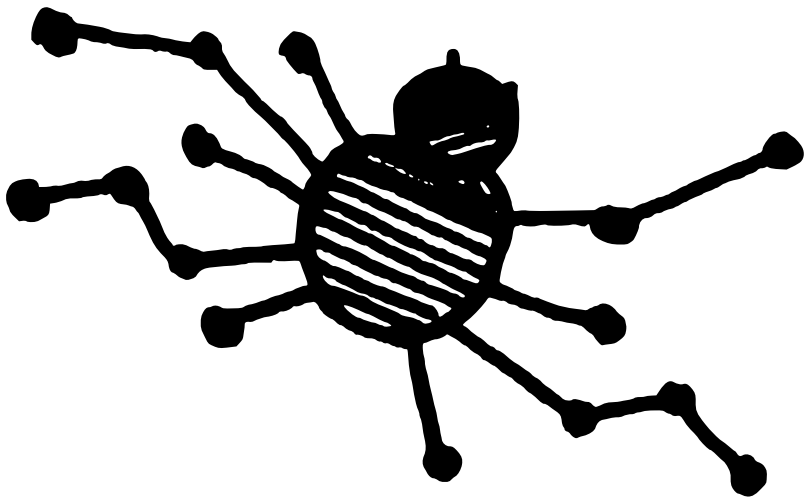
Rozdíl mezi průchodem do šířky a průchodem do hloubky je pouze ve volbě oné datové struktury. V případě průchodu do šířky používáme tzv. *frontu*. Frontu si představte jako obyčejnou frontu (např. lidí) – když přidáme prvek do struktury, zařadí se na konec; pokud chceme vybrat prvek ze struktury, vybereme zepředu. Použití fronty způsobí, že nejprv projedeme všechny sousedy počátečního vrcholu, pak sousedy jeho sousedů atd. V podstatě tedy vrcholy procházíme podle vzdálenosti od počátečního vrcholu (tento způsob tedy může být výhodný při hledání nejkratší cesty).

V případě procházení do hloubky používáme *zásobník*. Zásobník je datová struktura, kde prvky vložené jako poslední budou odebrány první. Při vkládání prvku do zásobníku ho zařadíme na konec, při odebrání prvku ovšem odebíráme také z konce. Tato metoda má hlavní výhodu v implementaci, neboť lze výhodně využít rekurzi. Díky rekurzi nemusíme zásobník nijak implementovat, protože využijeme zásobník pro volání funkcí. Prohledávání do hloubky pak může vypadat takto:

```
pruchod_do_hloubky(vrchol v) {
    označ v~jako navštívený
    foreach (soused s~vrcholu v)
        if (s~není označen jako navštívený)
            pruchod_do_hloubky(s)
}
```

Pomocí průchodu do hloubky lze snadno označit všechny dosažitelné vrcholy či zjistit, jestli je v (neorientovaném) grafu nějaký cyklus (pokud v cyklu foreach najdeme vrchol, který již byl navštívený, můžeme říct, že zde je cyklus, neboť jsme se do něj prve museli dostat jiným způsobem).

Tímto náš letmý úvod do tajů grafů končí. Přejeme vám hodně zdaru při řešení úloh 4. sady!



Zadání 4. sady úloh KSI (termín odevzdání: 16. 3. 2015)

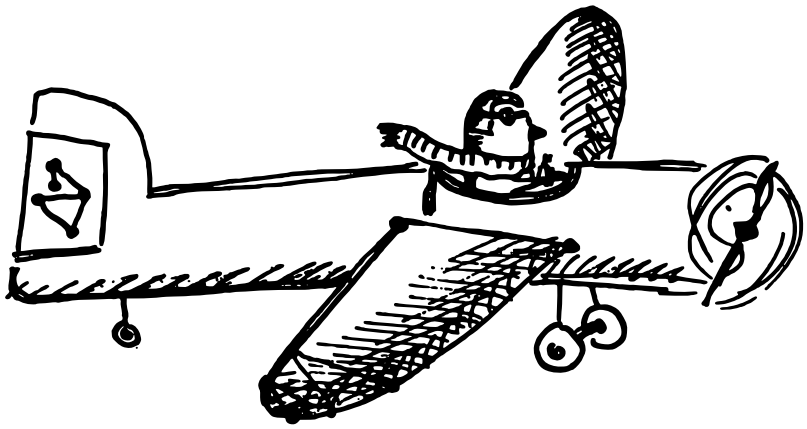
Řešení zasílejte pomocí internetového systému na adrese `http://ksi.fi.muni.cz`.

Příklad 1: Karlík a letenky (10 bodů)

Karlík rád chodí na dovolenky, preto plánuje dovolenku aj toto leto. Z minulých rokov sa už poučil a vie, že ako aj samotnú dovolenku, tak aj letenky musí vybavovať už v zime – teraz, inak už bude všetko plné. Karlíkovi sa zapáčila najbližšia destinácia a rozhodol sa, že toto leto pôjde tam. Keď si chcel zarezervovať letenky zistil, že priama letenka je drahšia ako letenka s prestupom v inom meste. Toto ho dosť zarazilo a preto začal skúmať letecký poriadok.

V krajine sa nachádza n letísk. Letecký poriadok obsahuje ceny letov medzi všetkými letiskami. Karlíka by zaujímali všetky letiská x také, že keď letí do susedného letiska $x+1$, tak existuje letisko y také, že let z x do $x+1$ cez y je lacnejší ako priamy let z x do $x+1$. Skúste Karlíkovi pomôcť a nájsť všetky také letiská. Toto je teoretická úloha, takže odovzdávate popis a pseudokód. Pri riešení úlohy využite maticovú reprezentáciu grafu.

Hodnotiť budeme ako časovú tak aj pamäťovú zložitosť a aj samotnú elegantnosť riešenia. Nezabudnite zdôvodniť prečo váše riešenie dá vždy správny výsledok.



Příklad 2: Zvířátka (10 bodů)

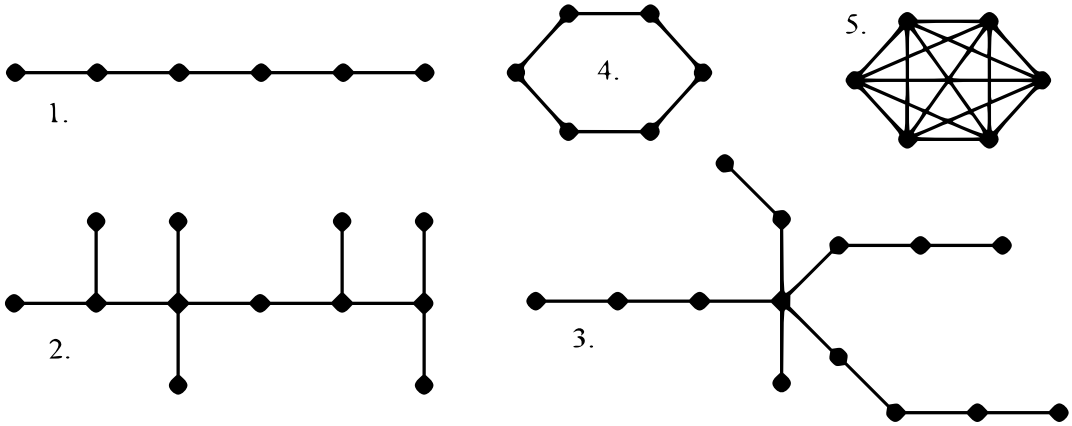
Karlík se vydal do zahrádky fotit zvířátka. Byl tam celé odpoledne a nafotil spoustu fotek. Jenže ouha – večer zjistil, že fotky jsou rozmazané a nekvalitní a jsou na nich vidět jen obrysy. Pomozte Karlíkovi a vytvořte program, který podle obrysů pozná, o které zvíře se jedná.

Konkrétně: Naimplementujte funkci `detekujZvire`, která jako parametr dostane libovolný vrchol grafu (zadaného seznamem sousedů) a vrátí číslo zvířete. Karlík fotil těchto 5 druhů zvířat:

1. *Žížala*: Má tvar cesty délky alespoň 2 hrany.
2. *Housenka*: Opět cesta délky alespoň 2 ("páteř"), avšak z každého vrcholu páteře můžou vyrůstat maximálně 2 cesty délky 1.
3. *Pavouk*: Acyklický graf se středovým vrcholem, ze kterého vyrůstá libovolný počet libovolně dlouhých cest (které se již nijak nerozvětvují).
4. *Brouček*: Cyklus délky alespoň 3 vrcholy.
5. *Larva*: Jedná se o úplný graf mající alespoň 4 vrcholy.

Jestliže bude graf vyhovovat více definicím, funkce vrátí nejmenší vyhovující číslo. Pokud naopak nevyhovuje žádné z definic, vraťte číslo 0. Detekci jednotlivých zvířat provádějte každou ve zvláštní funkci.

Vstupní graf bude souvislý (tj. obsahuje maximálně 1 zvíře) a mezi každými 2 vrcholy povede maximálně 1 neorientovaná hrana.



Obrázek 2: Ilustrace tvarů jednotlivých zvířat

Tato úloha zaměřena primárně na vyzkoušení práce s grafem reprezentovaným seznamem sousedů. U každého typu zvířete však požadujeme napsat též stručnou myšlenku jak daný typ zvířete detekujete a časovou složitost detekce jednotlivých zvířat v závislosti na počtu vrcholů V a hran E.

Příklad 3: Karlík jede domů (10 bodů)

Karlík sice studuje u nás, ale pochází z úplně opačného konce zeměkoule – z daleké Antarktidy. Proto dokončil všechny zkoušky o chvíli dřív a vyrazil navštívit rodiče na svoji rodnou ledovou hroudu. Když přistáli, tak zjistil nepříjemnou věc. Ledy tají a jeho rodná hrouda se rozpadla na spoustu malých ker, které si volně plavou po okolí. A jeho rodný domek si také někam odplaval, takže ho nemohl najít.

Po chvilce přemýšlení přišel Karlík se skvělým nápadem: Je to přece tučňák, tak domů doplave! Vytáhl tedy svůj moudrý telefon a stáhl si nejnovější satelitní snímky svého okolí. Na nich našel iglů, kde bydlí jeho rodiče, a zjistil, že je to pořádně daleko. A aby toho nebylo málo, tak jeho kufr není úplně vodotěsný, takže vydrží ve vodě jenom chvilku, a potom ho Karlík musí vytáhnout z vody a trochu posušit, aby se mu nenamočily učebnice.

Sestavil si tak graf, kde jednotlivé kry reprezentoval jako vrcholy. Hranu mezi dvěma vrcholy udělal pouze pro ty dvojice vrcholů, mezi kterými stihne doplavat dřív, než se mu promočí kufr. U každé hrany si také poznamenal, jak daleko je to z jedné kry na druhou. Začal si plánovat cestu a pak si vzpomněl, že by mu maminka vyhubovala, kdyby do vody chodil moc často a byl z toho nemocný. Řekl si tedy, že víc než třikrát do té vody raději nevleze. To už na něj ale byl trochu moc složitý úkol, takže by potřeboval pomoct od vás.

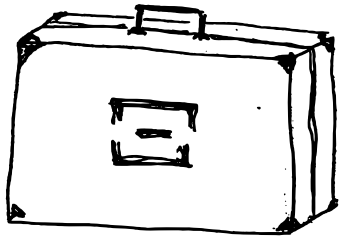
Vaším úkolem (I.) je navrhnout algoritmus, který pomůže Karlíkovi najít cestu domů. Cesta Karlíkovým grafem musí začínat na letišti (vrchol A) a končit u Karlíka doma (vrchol B). Po cestě můžete projít nanejvýš přes další dva vrcholy.

Druhým úkolem (II.) je pak najít takovou cestu, která je co nejkratší. Protože si Karlík poznačil vzdálenosti mezi jednotlivými krami, tak by rád znal cestu, při které bude muset uplavat co nejkratší celkovou vzdálenost. Stále ale také platí to, že Karlík nechce plavat více než tři úseky.

Pak se ale Karlík zamyslel a řekl si, že jeho kufr není úplně nejnovější, takže by se mu mohl promočit i dřív, než bylo uvedeno v návodu. Řekl si proto, že by bylo super znát takovou cestu, po které má nejmenší šanci, že se mu kufr promočí (III.). Třetí algoritmus, který by od vás Karlík chtěl navrhnout, bude hledat takovou cestu, která má „co nejkratší nejdelší hranu“. Pokud vám přijde že Karlíkovi nedělá polární prostředí dobře a už ani neví jestli chce nejdelší nebo nejkratší cestu, tak věřte že tomu tak není. Zkusíme to ale raději vysvětlit o něco podrobněji. Představte

si, že máte nějakou cestu c. Ta má několik hran. Z nich vybereme tu nejdelší (její délka je m) a řekneme, že cesta c má nejdelší hranu s délkou m. Když takhle označíme všechny cesty, tak je můžeme srovnávat mezi sebou – prostě řekneme, že nejdelší hrana jedné cesty je kratší než nejdelší hrana druhé, neboli jedna cesta má kratší nejdelší hranu než druhá. Cesta s nejkratší nejdelší cestou je tedy taková cesta, jejíž nejdelší hrana je kratší než nejdelší hrany všech ostatních cest. A najednou vidíte že se Karlík nepomátl. Ještě je asi dobré dodat, že někdy může existovat více cest se stejně dlouhou nejdelší hranou. Pokud jsou zároveň nejkratší, pak obě dvě prohlásíme za cesty s "nejkratší nejdelší hranou" a stačí naleznout jenom jednu z nich.

Pro analýzu časové a prostorové složitosti použijte n, které značí nejvyšší stupeň vrcholu v grafu, neboli všechny vrcholy mají nanejvýš n sousedů. Pokud vyřešíte více podúloh, tak složitost stačí zdůvodnit pouze pro tu s nejvyšším indexem.



Příklad 4: Podzemní honěná (10 bodů)

Karlík si často hraje se svou kamarádkou myškou na honěnou. Ale ne obyčejnou. Karlík běhá po povrchu a myška se prohání pod zemí ve svém rozsáhlém systému nor složených z uzlů (vrcholů) a cestiček (hran) – jedná se o souvislý graf.

Z každého vrcholu může vést libovolný počet cest. Myščiny nory jsou však vždy acyklické. Karlík chodí po povrchu a sleduje systém nor – přechází pouze mezi dvěma vrcholy spojenými hranou. Jelikož je Karlík tučňák, neumí moc rychle utíkat. Zato myška je pod povrchem rychlá jako blesk a Karlíkovi může vždy utéct – než Karlík přejde z vrcholu do jiného vrcholu, myška stihne přeběhnout kamkoliv do grafu. Oba dva disponují dobrým sluchem, a tak navzájem slyší své dupání a ví, kde se ten druhý nachází. Když tedy Karlík stojí nad vrcholem, tak ví, jaké hrany z něj vedou, ze které hrany přišel, a ve směru které hrany se nachází myška. Nepamatuje si nic více z předchozího přesunu. Myška má dobrou intuici a umí předvídat každý Karlíkův krok.

Karlík má před každou hrou daný počet kolíků, které zapichuje do země skrze vrcholy. Takto označený vrchol je pro myšku neprůchozí. Jakmile Karlík zabodne kolík, tak jej až do konce hry nesmí vytáhnout. Karlík si může být jistý, že mezi momentem, kdy zjistil kterým směrem se nachází myška, a zabodnutím kolíku, mu myška nepřeběhne jinam. Karlík vyhrává v momentě, kdy je myška uvězněná – všechny vrcholy okolo ní jsou zablokované a tedy musí zůstat na současném vrcholu nebo hraně. Naopak myška vyhrává tehdy, když Karlíkovi dojdou kolíky a ona stále může přebíhat alespoň mezi dvěma vrcholy.

Karlíkovi se v této hře doposud příliš nedařilo. Blíží se jaro, kdy ji opět budou hrát, a tak se zamýšlí nad herní strategií. Pomoz mu ji vymyslet!

- Tradičně začínají hrou, ve které si utahují z jednoho organizátora KSI a jeho oblíbeného typu příkladu. Tehdy se myška pohybuje v noře, která má tvar cesty délky n. Jaký je nejmenší počet kolíků, které Karlík potřebuje, aby pro něj existovala výherní strategie? Vysvětli, proč nelze použít menší počet kolíků a popiš Karlíkovu strategii.
- Běžná hra se odehrává v libovolné noře, která splňuje výše uvedené podmínky (acyklický graf). Karlík ví o jedné výherní strategii – do každého až na jeden uzel zabodne kolík. Tím myšku vždy uvězní, ale potřebuje na to $n - 1$ kolíků (n je počet uzlů v noře). To se mu zdá

mnoho. Pomoz mu vymyslet herní strategii, která využívá menší počet kolíků! Tuto strategii popiš. Na čem závisí počet použitých kolíků?

- Myška nabídla Karlíkovi, že mu na jednu hru nakreslí, jak vypadají její nory – Karlík tedy dopředu zná graf a může tedy přecházet mezi libovolnými dvěma vrcholy. Pomůže to Karlíkovi? Jak se změní jeho herní strategie?

Odevzdávej přesný popis Karlíkových strategií (ideálně v podrobném pseudokódu) a vysvětli, proč tvá strategie funguje a proč je optimální.

Příklad 5: Grafové domino (10 bodů)

KSI vzniklo před necelými devíti lety, a tak není divu, že otcové zakladatelé a matky zakladatelky přešli ze zakládání soutěží na zakládání rodin. Někteří potomci zakladatelů se už naučili mluvit, je tedy nejvyšší čas naučit je taky informatiku. Jejich oblíbenou hrou se proto v poslední době stalo grafové domino, které vám posíláme v dopise společně se zadáním, případně ho můžete najít i na našich stránkách ke stažení, vytisknutí a rozstříhání.

Každý z 29 dominových kamenů (z papíru) je rozdělen na dvě poloviny a na každé z těchto polovin je právě jeden graf. Kromě dominových kamenů hra obsahuje ještě kartičky vlastností (např. izomorfismus, počet vrcholů, chromatické číslo atp.).

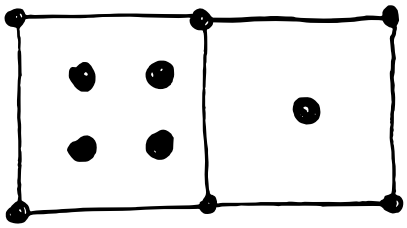
Pravidla

Na začátku hry se jeden z nich náhodně vybere a položí doprostřed hrací plochy (grafy směrem nahoru). Zbytek dominových kamenů se rozdá mezi hráče tak, aby jich měl každý stejný počet (dolní celá část z 28 děleno počet hráčů). Kartičky vlastností se zamíchají a celý balíček se položí rubem (zadní stranou) nahoru.

Začíná hráč, který naposledy programoval nějaký grafový algoritmus. Dále se pak hráči střídají v záporném směru (tj. po směru ručičkových hodin). První hráč před začátkem svého tahu otočí vrchní kartičku z balíčku grafových vlastností. Ta určuje, které grafy lze v tomto kole k sobě přikládat – musí mít tuto vlastnost stejnou (např. musí být izomorfní, musí mít stejný počet vrcholů, stejné chromatické číslo, atp.). V případě nejkratších a nejdelších kružnic vycházíme z definice kružnice, která požaduje, aby kružnice obsahovala alespoň 3 vrcholy a povolujeme k sobě lze přikládat také grafy, které žádnou kružnici neobsahují, tj. acyklické grafy (protože délka jejich nejkratší/nejdelší kružnice je *nedefinovaná*, a tedy stejná).

Hráč, který je na tahu, hledá možnost, jak přiložit jeden ze svých kamenů na jednu ze dvou stran cesty tvořené již položenými dominy. Kameny lze přikládat pouze na tyto 2 konce (tj. nelze větvit) a není ani povolené cestu svým kamenem uzavřít. Pokud hráč nemůže přiložit žádný ze svých kamenů na žádnou stranu cesty, povolenou operací je odebrat libovolný počet kamenů z jedné, nebo druhé strany cesty tak, aby na nový konec cesty už jeden ze svých kamenů položit mohl. Odebrané kameny jdou do ruky tohoto hráče. V aktuálním tahu samozřejmě nemůžu použít kameny, které jsem zrovna v tomto tahu odebral. Každý hráč končí svůj tah tím, že umístí v souladu s aktuální vlastností jeden ze svých kamenů, tahu se nelze vzdát. První hráč (pouze tento) vždy na začátku svého tahu obrací novou kartičku vlastnosti. Pokud už byly otočeny všechny, balíček se zamíchá a použije znovu stejným způsobem.

Úkolem je zbavit se co nejdříve všech svých dominových kamenů. Hraje se ale do té doby, dokud se kamenů nezbaví – a tedy nevyhrají – úplně všichni. Remíza není povolená. Takže od jistého okamžiku (zejména když už se chce všem jít na oběd) se jedná o týmovou hru. V této hře tedy nemůžete prohrát (ale můžete být tím, který vyhrál jako poslední).[9]



Několik pozorování

Stojí za povšimnutí, že klasické domino je speciálním případem našeho obecného grafového domina, v němž se vyskytuje pouze 6 typů grafů (a to dost nudných, úplně bez hran!) a po celou dobu hry je vyžadovaná vlastnost přikládaných grafů stejná, totiž izomorfismus.

Izomorfismus je ze všech použitých vlastností dost speciální, zkuste si rozmyslet proč.

Jednak se na rozdíl od všech ostatních nejedná o numerickou vlastnost, ale o vzájemnou vlastnost dvojice grafů (takže o nemá smysl mluvit o vlastnosti izomorfismu grafu samotného, vždy pouze vzhledem k nějakému jinému grafu).

Ještě zajímavější je, že z izomorfismu mezi 2 grafy vyplývá, že budou mít i všechny ostatní numerické vlastnosti (jako je počet hran, chromatické číslo nebo délka nejkratší kružnice) stejné. Pokud jsou tedy 2 grafy izomorfní, můžete je k sobě přiložit v jakémkoliv kole. Samozřejmě k sobě obvykle můžete přiložit i spoustu dvojic neizomorfních grafů. Takže skutečně nejtěžším kolem je (trochu překvapivě) izomorfismus, protože existuje nejméně možností, které grafy lze k sobě přikládat.

Zadání

Vaším úkolem je ke každé kartičce vlastnosti napsat skupiny grafů, které mají stejnou hodnotu požadované vlastnosti (např. mají stejný počet vrcholů, jsou navzájem izomorfní atp.), tj. grafy, které můžou být v kole s touto kartičkou vlastnosti přiloženy k sobě. Grafy identifikujte pomocí čísel na kartičkách. Pro každou vlastnost stačí napsat rozklad grafů podle této vlastnosti, v této úloze není potřeba nic vysvětlovat.

Než se vyděsíte, že musíte rozkládat 58 grafů podle 9 různých vlastností, uvědomte si, co jsme si řekli o izomorfismu. Pokud rozložíte grafy na množiny izomorfních grafů (a těch je v našem dominu 15), stačí dále pracovat už jen s těmito množinami – nebo pro jednoduchost s *reprezentanty* těchto

[9] V případě nejasností ohledně pravidel grafového domina se ptejte na diskuzím fóru.

množin, což může být číslo libovolného grafu v množině izomorfních grafů, my pro jednoznačnost budeme používat vždy to nejmenší číslo.

Rozklad na množiny izomorfních grafů může například vypadat takto (jedná se jen o ilustraci zápisu, nesouvisí se skutečným řešením úlohy):

```
izomorfismy = {
        1: {1, 2, 5, 10, 11},
        3: {3, 4, 8, 15},
        6: {6, 12}
        ...
}
```

Takový zápis např. říká, že grafy 1, 2, 5, 10 a 11 jsou navzájem izomorfní (a 1 byla zvolena jako reprezentant, protože je z nich nejmenší).

Co se týče dalších vlastností, budeme už pracovat pouze s reprezentanty. Pokud bychom měli reprezentanty 1, 3, 6, 7, 9, 13 a 14 mohl by rozklad podle vlastnosti *divnost*, která nabývá hodnot 11, 12 a 13, dopadnout takto:

```
divnost = {
        11: {1, 7},
        12: {3, 14},
        13: {6, 9, 13}
}
```

Vaše řešení zapište do souboru *task4_5.py*, ve kterém je způsob zápisu podrobně popsán. Jestli jste při zadávání něco nepopletli, můžete zkontrolovat spuštěním souboru *run4_5.py* (tento program kontroluje pouze správnost zápisu, nikoliv správnost řešení jako takového).

Bonusová úloha

Natočte video se zápasem v grafovém dominu. Nahrajte ho třeba na YouTube a pošlete nám na něj odkaz. Každý zúčastněný dostane 1 bod, další body lze získat za vítězství v následujících kategoriích.

1. místo – nejneobvyklejší místo na hraní grafového domina
2. čas – nejrychleji zahraná partie 2 hráčů
3. zápas – hodnotí se napětí, zajímavost, atmosféra, velkolepost
4. počet osob – co nejvíce osob
5. osobnosti – hodnotí se subjektivní zajímavost hráčů (dospělí < senioři < malé děti; spolužáci < učitelé < ředitelé < prof. Petr Hliněný)
6. nejvíce shlédnutí (v době hodnocení)

Ve všech případech se musí jednat o kvalitní partie, žádné vzdávání, remízy atp. Kromě kategorie *čas* však může být zápas libovolně sestříhaný, zrychlený a upravený.

Přejeme hodně zábavy při hraní grafového domina!

A to je z této sady KSI vše. Přejeme ti hodně úspěchů při řešení, a když budeš mít jakékoliv otázky, neváhej se na nás obrátit e-mailem na adresu `ksi@fi.muni.cz` nebo v diskuzním fóru na našich webových stránkách.

Termín odevzdání 4. sady úloh KSI: 16. 3. 2015

`http://ksi.fi.muni.cz`