KSI 2014/2015

# Úloha 4-4: Podzemní honěná

Jan Horáček
Gymnázium, Brno, Vídeňská 47; jan.horacek@seznam.cz

16. března 2015

## 1 Cesta

Pro vyřešení této úlohy využijeme binárního vyhledávání. Konkrétně budeme umisťovat kolíky do vrcholů, které odpovídají středům aktuálního intervalu a interval posuneme vždy tím směrem, ve kterém se nachází myška. Konkrétní algoritmus implementuje níže uvedený pseudokód.

```
vlevo  = 0
vpravo = n-1
while vlevo < vpravo:
 stred = (vpravo+vlevo)/2
 umisti_kolik(stred)
 if myska(stred) == vlevo:
  vpravo = stred - 1
 else:
  vlevo = stred + 1
```

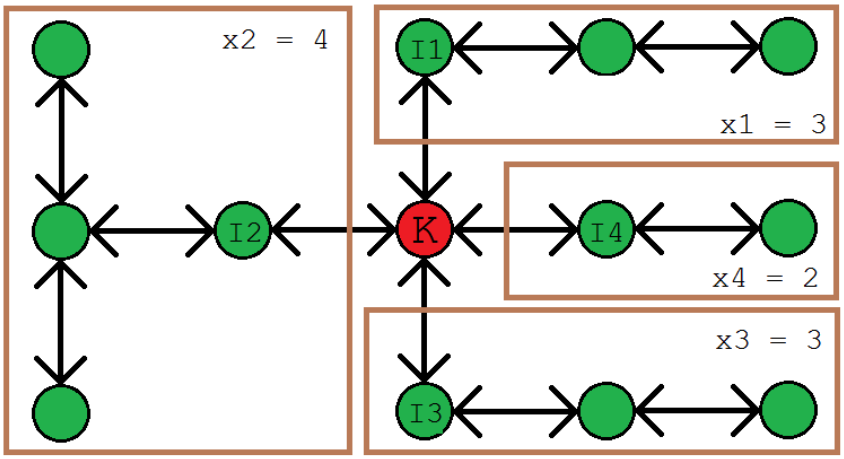
Nejmenší možný počet kolíků je tedy $log_2(n)$.

Menšího počtu kolíků bychom dosáhli při vyšším základu logaritmu, pro to bychom ale potřebovali výraz s více výstupními hodnotami, než nám poskytuje výraz pro určení pozice myšky. Zde máme pouze 2 hodnoty: `vlevo` a `vpravo`, proto se můžeme rozhodnout pouze binárně. Proto binární vyhledávání.

## 2 Neznámý strom

Zamysleme se nad tím, která díra by pro nás byla nejoptimálnějším kandidátem pro zasunutí kolíku. Logicky se jedná o takovou díru (vrchol grafu), která je, intuitivně řečeno, jakýmsi kořenem grafu. Hledáme totiž takový vrchol, který nám graf rozdělí na více pokud možno stejně velkých částí — tak docílíme oříznutí největší části grafu a to je náš cíl.

Označme vrchol $K$ jako *kořen grafu* a podívejme se na to, jak bychom jej našli. Kořen $K$ je takový vrchol, který má nejmenší součet odchylek proměnné $x_I$ od průměru $p_K$ hodnot $x_1 ... x_n$. $n$ je počet sousedních vrcholů vrcholu $K$. $x_I$ je počet vrcholů za sousedním




Obrázek 1: Ilustrace termínu *děti vrcholu*

vrcholem $I$ vrcholu $K$ včetně tohoto vrcholu $I$. Neboli jakýsi "počet dětí vrcholu $K$ ve směru vrcholu $I$". Celou situaci přehledně ilustruje obrázek 1.

Ujasněme si tedy ještě jednou výpočet kořene:

1. Hledáme takový kořen $K$, pro který je součet odchylek $s_K$ nejmenší ze všech možných vrcholů.
2. $s_K = \sum_{i=1}^{n}(x_i - p_K)$
3. $p_K$ je průměr hodnot $x_1..x_n$ pro vrchol $K$.
   $p_K = \frac{1}{n} * \sum_{i=1}^{n} x_i$

Ať už určíme hodnoty $x$ jakkoliv, je jasné, že k jejich určení bude zapotřebí projít celý graf (strom). To je v části II. zcela nereálné. Neopusťme ale naši úvahu o hledání kořene a pokusme takový kořen aproximovat.

Za daných Karlíkových znalostí definovaných zadáním je dobrou aproximací kořene takový vrchol, který má ze všech vrcholů v grafu největší stupeň. A to je informace, kterou je schopen Karlík z grafu zjistit.

Karlík by se tedy mohl řídit jednoduchým algoritmem pro vyhledání vrcholu s nejvyšším stupněm: prostě bude postupně navštěvovat všechny vrcholy a nakonec vybere ten, který má nejvyšší stupeň. Pokud není povoleno pamatovat si, které vrcholy Karlík navštívil, Karlík může prostě náhodně navštěvovat různé vrcholy a bude limitován maximálním počtem navštívených vrcholů $v_{max}$, což může být například konstanta zadaná programátorem.

Tak nebo tak, po skončení tohoto optimálně lineárního algoritmu vzhledem k počtu vrcholů Karlík vybere vrchol s nejvyšším stupněm.

Do tohoto vrcholu zabodne Karlík kolík a tím ořeže (při dobré konstelaci hvězd) poměrně vekou část grafu. Karlík lokalizuje směr, ve kterém se nachází myška a všechny ostatní směry z grafu skutečně ořízne. Nyní tedy řeší stejný problém, ale pro menší graf. Karlík tedy může opakovat celý algoritmus.

Pokud by byl Karlík hodně chytrý, může si zapamatovat stupně vrcholů do vhodné datové struktury a při další iteraci algoritmu jen vybrat nejvyšší stupeň v daném podgrafu, jít přímo do něj a zabodnout do něj kolík.

Jelikož algoritmus funguje sice iterativně, ale vždy na menším podproblému, musí zaručeně skončit.

Počet použitých kolíků závisí především na tvaru grafu a přesnosti aproximace kořene.

# 3 Předem známý strom

Známost grafu nám v mnohém pomůže — především nemusíme provádět výše zmíněnou aproximaci, ale počítat hodnoty $x$ přesně. To, co nás ve výsledku zajímá, je, jak už jsem popsal v předchozí části, takový vrchol $K$, jehož $s_K$ je ze všech vrcholů minimální. Použijeme tedy výše zmíněný algoritmus pro výpočet této hodnoty, vrchol s minimální hodnotou $s$ pak najdeme v lineárním čase vzhledem k počtu vrcholů (prostě proiterujeme vrcholy a hledáme minimum).

Zajímavou otázkou ovšem zůstává výpočet hodnot $x$, který by bylo vhodné provést co nejrychleji.

Pro každý vrchol $V$ tedy chceme vypočítat hodnoty $x_1...x_n$. To uděláme pomocí následující algoritmu spuštěného samostatně v každém vrcholu a založeného na posílání zpráv mezi vrcholy:

```
Self.child_cnt = 1
Self.prijmuto_zprav = 0
Self.x[] = {0, ...}
while (not Self.Terminated):
 Self.ReadData()

 // pokud jsme prijmuli zpravy ze vsech vrholu, skoncime
 if (Self.prijmuto_zprav == Self.n): Self.Terminate()

 // pokud jsme dostali zpravy ze vsech vrcholu krome jednoho,
 //  posleme na zbyvajici vrchol pocet deti
 if (Self.prijmuto_zprav == Self.n-1):
   // pocet vrcholu za zbyvajicim vrcholem
   //  pak musi byt roven tomuto vyrazu:
   Self.x[zbyvajici_vrchol] = pocet_vrcholu_grafu
                              - sum(Self.x[]) - 1
   Self.Send(zbyvajici_vrchol, Self.child_cnt)
   Self.Terminate()
```

Funkce `Self.ReadData` čte počty dětí odeslaných z jednotlivých vrcholů.

```
def ReadData():
 for v in Self.sousedi:
  if !Empty(pocet_deti_z_vrcholu = v.Read()):
   Self.x[v] = pocet_deti_z_vrcholu
   Self.prijmuto_zprav++
```

Lidsky řečeno: tento algoritmus propaguje počty dětí z listů grafů do všech ostatních vrcholů grafu. Po jeho skončení jsou v každém vrcholu dostupné hodnoty $x_1..x_n$.

S těmito hodnotami tedy provedeme příslušné výpočty pro získání hodnoty $s_K$ (definováno výše) a najdeme takový vrchol, jehož $s$ je minimální. Suma sumárum proběhnou všechny tyto kroky lineárně vůči počtu vrcholů $V$. Tedy $O(n)$. Nechť se celý tento algoritmus jmenuje "algoritmus pro zjištění $s$".

Dobrá věc je to, že hodnoty $s$ stačí pro celý graf vypočítat pouze jedenkrát, tento algoritmus tedy pustíme pouze na začátku. Karlík bude, podobně jako v části II., postupně ořezávat graf. Při každém ořezu ale jen znovu nalezne vrchol s maximální hodnotou $s$, tentokrát ale už v ořezaném grafu.

Karlík se tedy bude chovat takto:

1. Nechť $G$ je vstupní graf od myšky.
2. Spočítej $s$ dle *algoritmu pro zjištění s*.
3. Najdi vrchol $K$ v grafu $G$, pro který je $s_K$ minimální ze všech vrcholů.
4. Zabodni kolík do vrcholu $K$.
5. Urči směr myšky a ořízni z grafu $G$ všechny směry, ve kterých se myška nenachází. Z grafu $G$ ořízni i vrchol $K$.
6. Pokud je $G$ prázdný, skonči, jinak pokračuj na krok 3.

Časová složitost výrazně závisí na tvaru grafu, proto si dovolím určit pouze asymptotickou časovou složitost jedné iterace kroků $3 - 6$, která je $O(n)$, kde $n$ je počet vrcholů grafu.