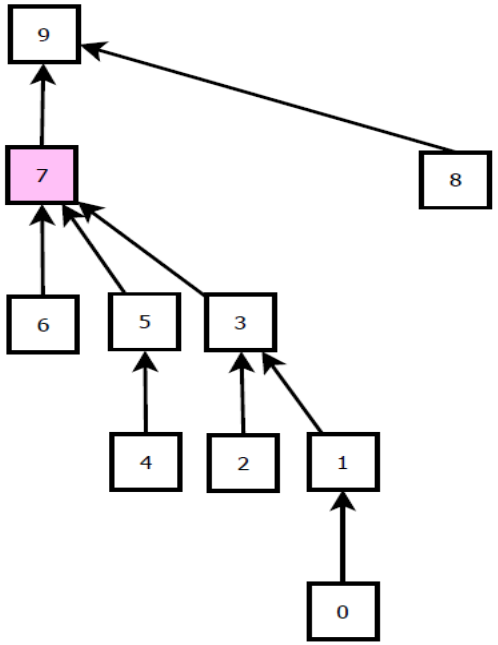
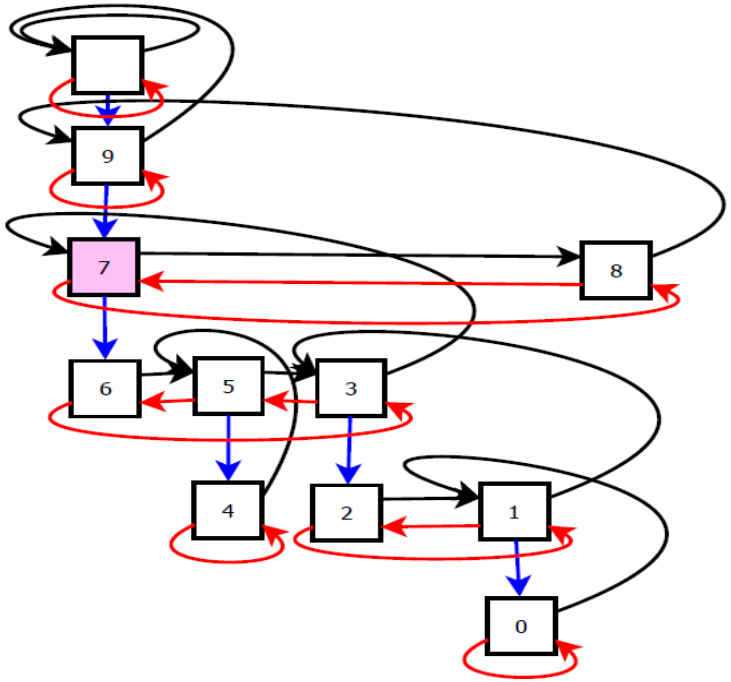
Padovan heaps

Vladan Majerech





Základní princip – velmi drahé porovnávání

- Porovnáváme jen když musíme (v podstatě veškerou práci dělá FindMin)
- Výsledky porovnávání nezahazujeme
- Nesmíme organizací porovnávání ztratit řádově víc času než porovnáváním

Insert

- Do seznamu prvků haldy vložíme nový prvek
- O(1)
- Budeme muset ještě nějaký čas naspořit na budoucí operace – uvidíme později

FindMin

- Porovnáváme „kandidáty“ na minimum, tedy prvky, pro něž neznáme menší prvek
- Výsledek každého porovnání zaznamenáme (symbolicky hranou směrem k menšímu prvku), reprezentaci probereme později
- Na konci máme jediného kandidáta na minimum.
- Potenciál Φ_0 rovný počtu kandidátů na minimum může všechna porovnání zaplatit a navýší cenu Insert o O(1)

DeleteMin

- Předpokládáme, že předcházel FindMin, případně jej pro jistotu zavoláme.
- Odstraníme minimum, jeho předci se stanou novými kandidáty na minimum.
- Je možno implementovat v O(1) čase, ale počet předků musíme přidat do potenciálu $\Phi_{0.}$
- Potřebujeme aby stromy byly úzké, aby cena DeleteMin nebyla velká.

Invariant c-q úzkosti

Každý prvek haldy bude mít definovaný řád ($rank_v$) (nový prvek 0).

- Pro c>0 a 1<q≤2 platí:

Libovolný vrchol řádu k má alespoň cq^k velký podstrom předchůdců.

Řád a stupeň nebude nutně totéž. Abychom v DeleteMin měli z čeho zaplatit do potenciálu Φ_0, musíme mít účet pro situace, kdy je stupeň větší než řád.

$$\Phi_1 = \sum \deg\_v - rank\_v$$

Suma je jen přes kladné rozdíly.

DeleteMin podruhé

- Pokud prvky haldy budou splňovat invariant c-q úzkosti, pak největší možný řád je log (n/c)/log q, tedy po odstranění minima m bude počet kandidátů deg_m = rank_m+(deg_m-rank_m), tedy kromě přeúčtování z $\Phi_1$do Φ_0, musíme do Φ_0 přidat jen O(log n).
- Ačkoli worst case pro DeleteMin může být O(1), v našem amortizovaném rozboru bude cena DeleteMin O(log n).

Zajištění invariantu c-q úzkosti

- Při FindMin nemůžeme porovnávat kandidáty náhodně, mohly by vznikat široké malé stromy.

1. Spojování stromů stejného řádu je OK $2cq^k \geq cq^{\{k+1\}}$, můžeme kořenu spojeného stromu zvýšit řád, takže rozdíl $\deg\_v - rank\_v$ zůstane zachován.
2. Pouze pokud je již od každého řádu nejvýš jeden kandidát, můžeme je porovnat libovolně. Vytvoříme dvojice, ty porovnáme, pak opět vytvoříme dvojice ... nechceme porovnávat jednoho kandidáta se všemi. V druhé fázi neměníme rank, zvětšujeme Φ_1.

FindMin cena

- První fázi zaplatíme z Φ_0.
- Druhou fázi … příspěvek do Φ_1 nemáme z čeho zaplatit.
- Počet kandidátů na začátku druhé fáze je nejvýš O(log n) (1+maximální dosažitelný řád). Zároveň je to nejvýš počet kandidátů na začátku FindMin. Zavedeme proto potenciál Φ_2 rovný minimu z maximálního dosažitelného řádu + 1 a z počtu kandidátů.
- Insert musí platit O(1) jak do Φ_0, tak (možná) do Φ_2, DeleteMin zaplatí do Φ_2 nejvýš O(log n).

Implementační detaily

- Kandidáty na minimum udržujeme v obousměrném seznamu (doprava acyklický, doleva cyklický), abychom mohli přidávat na oba konce a odebírat libovolný prvek v O(1).
- Stejně jako kandidáty na minimum udržujeme seznam prvků porovnaných s daným kandidátem (větších).

FindMin implementační detaily

- Pro každý řád máme synchronizační místo (prázdné).
- V první fázi procházíme seznam kandidátů doprava a v synchronizačních místech jednotlivých řádů aktualizujeme odkazy na navštívené kandidáty. Pokud narazíme na místo odkazující na kandidáta, můžeme porovnat, odstranit většího z kandidátů připojit menšímu jako nejpravějšího přímého předchůdce a zvětšit řád …
- Důležité je, že seznam kandidátů je pouze aktualizován. V čase dle jeho délky můžeme na konci fáze vyprázdnit synchronizační místa.
- V druhé fázi procházíme doleva a vždy porovnáme dvojici a většího přesuneme na nejlevějšího předchůdce menšího a postoupíme doleva. Končíme, když máme jediného kandidáta.

Decrement (zatím máme Binomiální haldy)

- Snížením prvku přestane být spolehnutí na hranu z tohoto prvku a musíme ji tedy odstranit.
- Porušíme tím ale invariant c-q úzkosti?
- Pokud snížíme všem tranzitivním následníkům řád, tak bude c-q úzkost spravena, ale aktualizace by trvala hloubku prvku, což je až $\Omega(n)$.
- Tarjan – Fredman stačí, když budeme propagovat až každé druhé snížení řádu. Zavedení černých a bílých vrcholů.

Decrement analýza

- Vrcholy budou na začátku bílé. Při prvním snížení řádu přebarvíme na černé (kořeny necháváme bílé). V druhé fázi FindMin prvek který přestává být kandidátem zčervená.
- Zavedeme Φ_3=počet všech černých přímých předchůdců.
- V průběhu Decrement nejvýš jeden bílý vrchol přebarvíme na černo, můžeme ale černý vrchol přebarvit na červeno a propagovat snižování řádu dál. Tato propagace je ale zcela zaplacena ze snížení potenciálu Φ_3.
- Protože potřebujeme mít aktuální odkazy na následníky, worst case čas pro DeleteMin odpovídá nárůstu Φ_0, tedy $\theta(n)$.

Fibonacciho haldy

- Pro nejmenší M_k velikost stromu řádu k platí rekurence

$$M_k = 1 + 1 + M_0 + M_1 + \cdots + M_{k-2}$$

Tedy $M_{k+1} = M_k + M_{k-1}$ a dostáváme Fibonacciho posloupnost a c-q úzkost platí pro $q = \frac{1+\sqrt{5}}{2}$.

Tím je rozbor Fibonacciho hald ukončen.

P.S.: Rozbor by fungoval i pokud bychom místo červené barvy používali bílou. Jen bychom museli zabránit snižování řádů pod nulu.

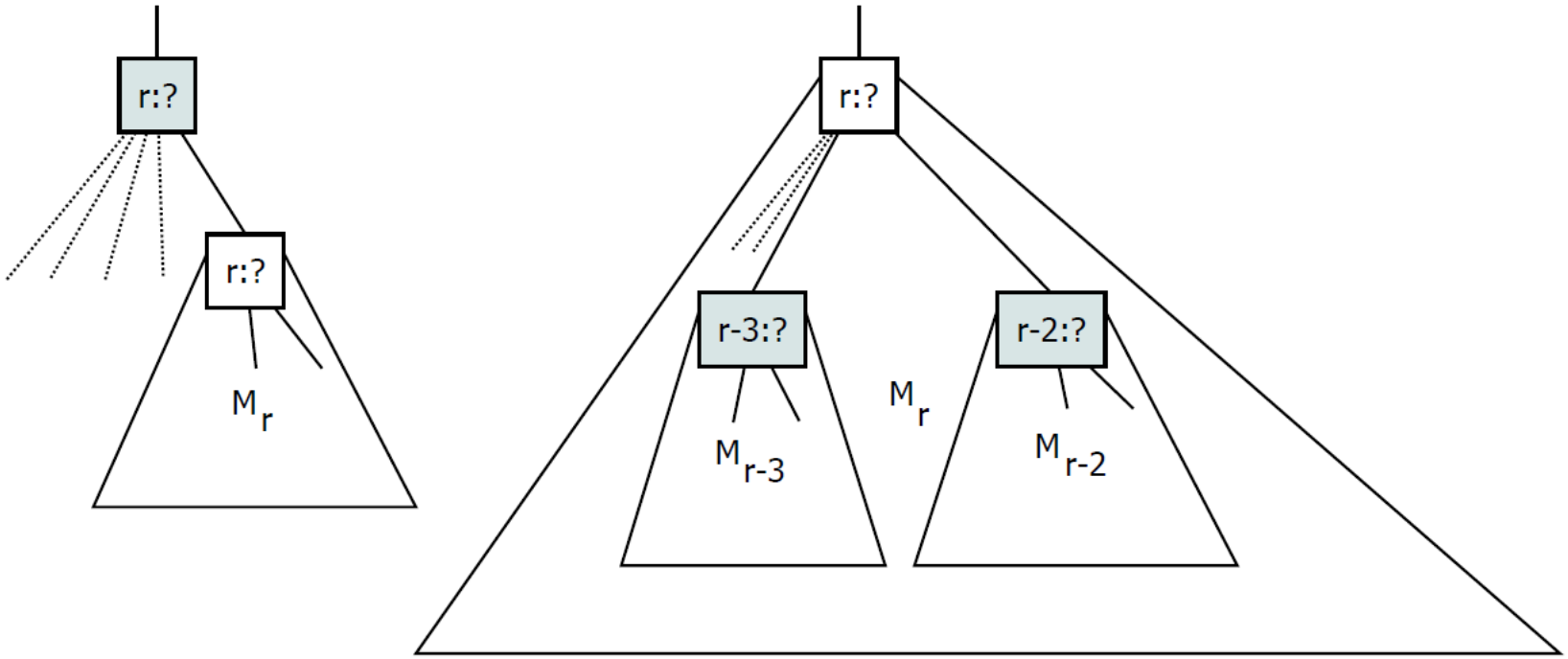
Ušetření pointeru

- Kvůli Decrement Fibonacciho haldy potřebují pointer na následníka prvku, abychom mohli propagovat snižování řádů.
- Šlo by tento pointer ušetřit a použít odkaz jen v posledním předchůdci (místo odkazu na dalšího)?
- Pak bychom nemohli propagovat snižování řádu s výjimkou prvků na konci seznamu, které mají k následníkovi O(1) přístup.
- Propagaci provádíme jen v posledních dvou prvcích seznamu (předchůdcích největšího řádu).
- Worst case čas DeleteMin klesne zpět na O(1).

$rank\_v$ a $wrank\_v$

- Pro černý vrchol je $wrank = 1 + rank$, pro bílý vrchol je $wrank = rank$, pro null (chybějící vrchol) je $wrank = -1$.
- Seznam předchůdců (dětí) má všechny červené nalevo, pak neklesá $wrank$ (pokud je definovaný).
- Vezmeme z bílých a černých přímých předchůdců (dětí) vrcholu v dva nejpravější … zprava w_0 a w_1. (mohou chybět)
  S. Bezpečný: $wrank_{w_1} + 1 = wrank_{w_0}$: $rank_v = wrank_{w_0} + 1$.
  D. Nebezpečný: $wrank_{w_1} + 1 < wrank_{w_0}$, a w_0 je bílý: $rank_v = wrank_{w_0}$. (v nemůže být bílé)
  F. Zakázaný: $wrank_{w_1} + 1 < wrank_{w_0}$, a w_0 je černý: zažluť w_0 a přepočítej.

Rekurence minimálních stromů



Úzkost

- Pro minimální velikost M_k stromu řádu k máme

$$M_k = 1 + M_{k-2} + M_{k-3}$$

- Úzká souvislost s Padovan posloupností a tedy c-q úzkost pro

$$q=\sqrt[3]{\left(\frac{1}{2}\left(1+\sqrt{\frac{23}{27}}\right)\right)}+\sqrt[3]{\left(\frac{1}{2}\left(1-\sqrt{\frac{23}{27}}\right)\right)} \approx 1.324718.$$

- $(q^3 = q + 1)$

Decrement zcela vybarvený

- Utrhneme prvek (kandidát na minimum), a pokud byl jedním z dvou nejpravějších předchůdců, vyvoláme kaskádovou aktualizaci řádů na jeho následníkovi.
- Při kaskádové aktualizaci vždy přepočítáme rank. Pokud nedošlo ke změně, výpočet končí.
- Při snížení ranku bílého prvku o 1 prvek začerníme.
- Pokud došlo ke snížení ranku černého prvku, je obarven žlutě, stejně tak při snížení ranku bílého prvku aspoň o 2.
- Je-li prvek mezi pravými dvěmi, pokračuje kaskádové přepočítávání.
- Při přepočítávání ranku prvku nejprve žluté vrcholy mezi pravými dvěmi přebarvujeme na červeno a přemísťujeme na levý kraj seznamu. Pokud narazíme na červený prvek, víme, že další bílý či černý prvek neexistuje.

Decrement

- Analýza Decrementu stále ještě nedává O(1).
- Přepočítání ranku může vést k poklesu až o 2, v takovém případě i pro bílý vrchol dojde k propagaci aktualizace. Tuto propagaci Φ_3 nezaplatí.
- Potřebujeme potenciál

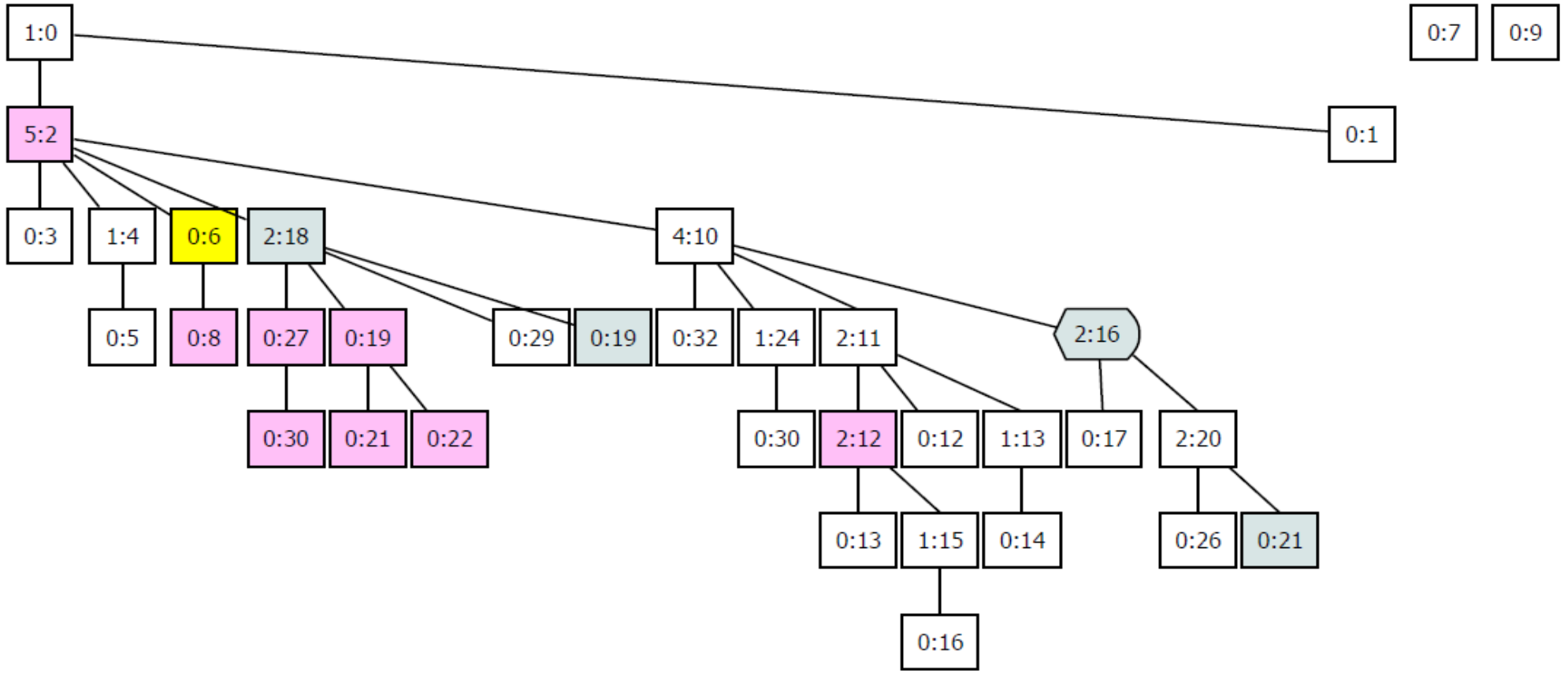
$$\Phi_4 = \sum rank_v \ - \text{počet přímých předchůdců } v \text{ bílé či černé barvy.}$$

- Odstraněním bílého či černého předchůdce šetříme na budoucí (větší) pokles ranku. Spojování stromů při FindMin nemá vliv na Φ_4. Jediné co jej mění je Decrement.
- Nyní již za pokračování kaskádovitého přepočítání řádů vždy zaplatí buď Φ_3 nebo Φ_4. Kromě toho máme konstantní příspěvek (až na přemísťování žlutých vrcholů).

Červená, žlutá a nebezpečné

- Kvůli odkládání přesunu žlutých vrcholů na začátek seznamu potřebujeme potenciál Φ_5 = počet všech žlutých předchůdců.
- DeleteMin místo $\sum \deg_v - rank_v$ stačí Φ_1=počet všech červených předchůdců.
- $\deg\_v =$ počet červených + počet žlutých + počet (bílých a černých), kde počet bílých a černých $\leq rank$, takže do Φ_0 zaplatíme z Φ_5, Φ_1 a nejvýš $O(\log n)$.
- Potřebujeme předcházet bílým nebezpečným vrcholům. Proto na začátku FindMin z nebezpečných vrcholů děláme bezpečné. To je důvod zavedení potenciálu Φ_6 = počet všech nebezpečných vrcholů.

Obrázek se všemi barvami (klíče mají mínus)



Shrnutí

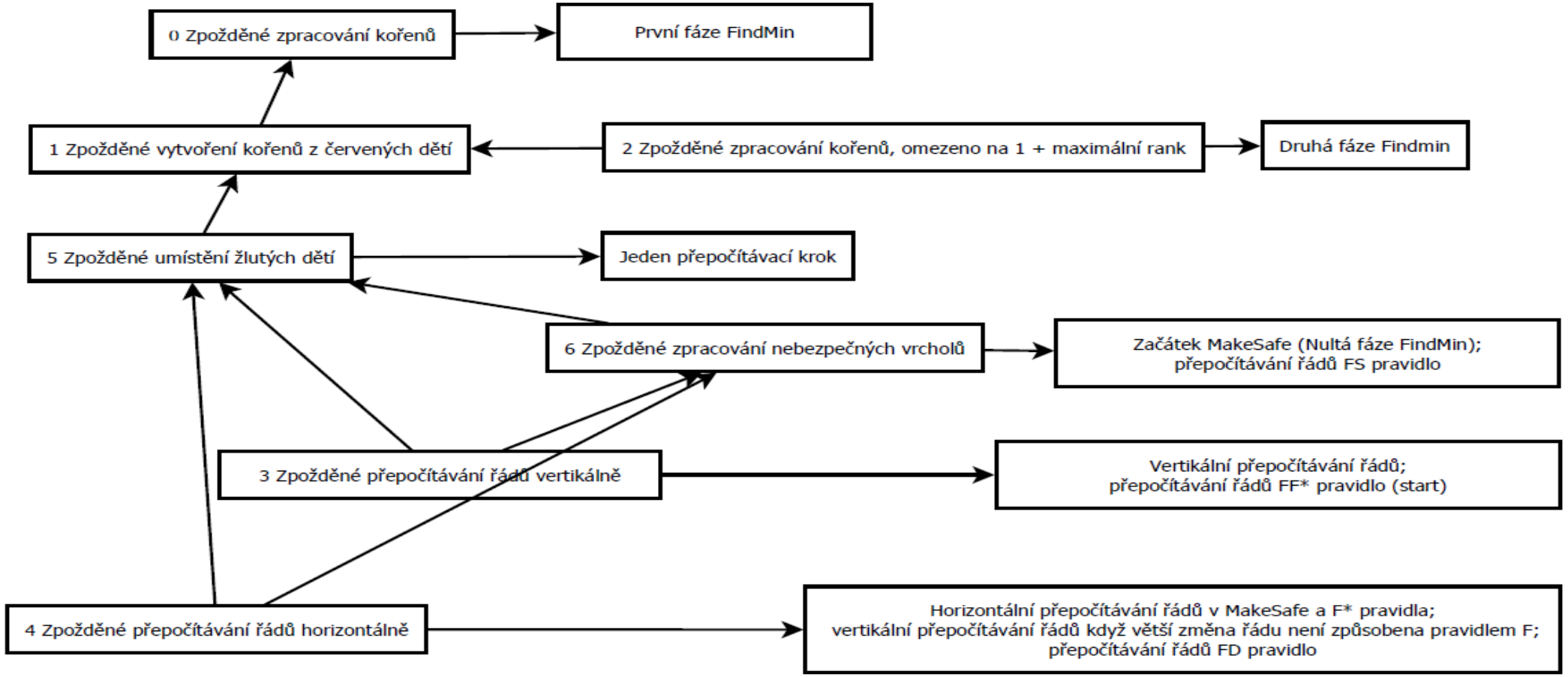
- Potenciál vůči němuž je analýza je tedy

$$\Phi_0 t_0 + \Phi_1 t_1 + \Phi_2 t_2 + 2\Phi_3 t_3 + \Phi_4 t_4 + \Phi_5 t_5 + \Phi_6 t_6$$

kde $t_0 \leq t_1 < t_2, t_1 < t_5 < t_6, t_5 + t_6 < t_3, t_5 + t_6 < t_4$ jsou vhodné konstanty.

- Φ_0 je počet kandidátů.
- Φ_2 je počet kandidátů, nejvýše však nejvyšší dosažitelný řád + 1.
- Φ_1, Φ_3, resp. Φ_5 je počet všech červených, černých, resp. žlutých předchůdců.
- Φ_4 je součet rozdílů mezi rankem a počtem bílých a černých přímých předků.
- Φ_6 je počet nebezpečných vrcholů

Schéma placení



Soutěž struktur

- Obhájce jedné struktury vygeneruje posloupnost volání metod, obě funkce provedou danou posloupnost a změří se poměr jejich časů.
- Haldy dle principu velmi drahého porovnávání vítězí nad standardními haldami hodnotou $\theta(\log n)$ pro posloupnost

  i=0;Repeat (Insert(-i), Insert(-(i+1)), FindMin, DeleteMin, i++),

  protože maximální řád, který vznikne bude 4.
- Standardní haldy mohou vynutit jedině poměr $\theta(1)$.