Homework 3 - splay_experiment

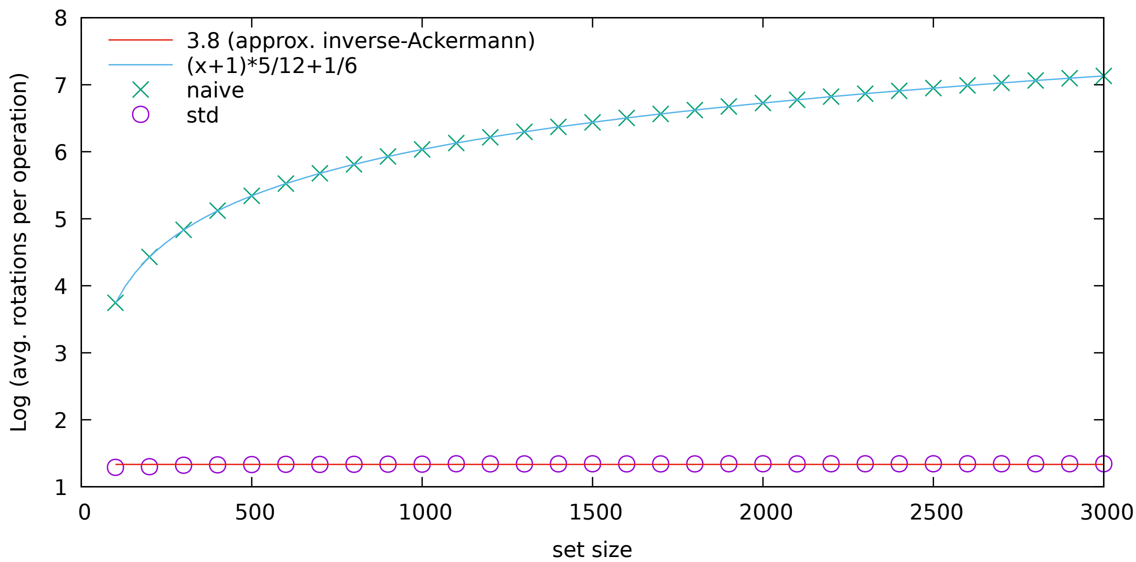
Jiří Klepl

Prolog

Následují diskuze tří zadaných testů (se seedem 80) a analýzy pozorovaných domnělých závislostí. Pro úvod by bylo vhodné říct, že všechny testy měří a průměrují počet provedených rotací naivní a standardní implementace splay stromu. Tento test je poměrně dobrým měřítkem časové složitosti práce nad těmito strukturami, neboť u splay stromů počet rotací odpovídá počtu čtení a dereferencí prvků, které vedou k přidanému/hledanému prvku (právě přidání a hledání prvku testy měří).

Sequential test

Figure 1: Sequential test results



Ve výsledcích tohoto testu znázorněných ve figuře 1 můžeme vidět velký rozdíl v počtu provedených rotací naivní a standardní implementace.

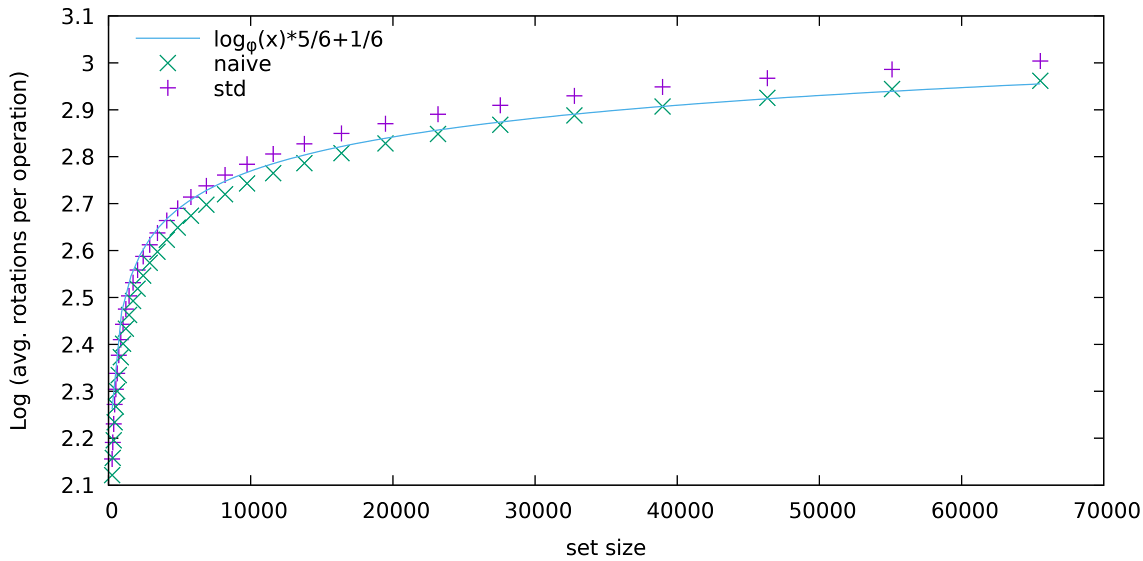
U obou implementací je složitost přidávání n prvků v $\Theta(n)$, tedy konstantní na jeden prvek, s tím i počet rotací. Výsledkem je vždy lineární seznam (při vzestupném přidávání propojen po levých synech). Tento výsledný tvar budou mít obě implementace po každém dokončení (přidávací/hledací) sekvence.

Složitost hledání prvku (a jím způsobený počet provedených rotací) v naivní implementaci je v $\Theta(n)$. Strom má během hledání vždy tvar dvou hlavami spojených seznamů, z nichž jednomu odtrháváme listy a přidáváme je v hlavě do druhého. Jednoduchou matematickou úpravou proto dostaneme velice přesnou lineární aproximaci závislosti průměrného počtu rotací na velikosti pole.

Složitost hledání prvku (a jím způsobený počet provedených rotací) ve standardní implementaci je v $\Theta(\alpha(n))$, kde α je inverzní Ackermannova funkce, důkaz toho lze nalézt na internetu. Tato analýza odpovídá mazání krajních bodů, když splay stromem reprezentujeme deque.

Random test

Figure 2: Random test results (seed 80)

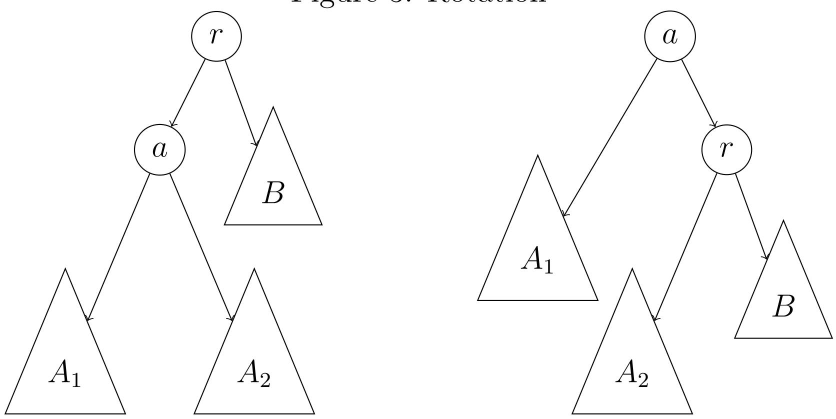


V testu s náhodným přístupem k prvkům dosahují obě implementace obdobného počtu rotací (pozn. znázorněná funkce je pouze k porovnání, není výsledkem žádné analýzy), na jeden prvek amortizovaně v $\Theta(n)$ (kde n je počet prvků). U standardní implementace toto není vůbec překvapivé, na přednáškách to bylo rigorózně dokázáno.

Jediným překvapením zde je podobnost výsledků naivní a standardní implementace. V dalším testu najdeme vysvětlení, které by částečně mohlo vysvětlovat i toto (včetně toho, proč naivní implementace dává lepší výsledky). Ono vysvětlení však zde není uspokojivé.

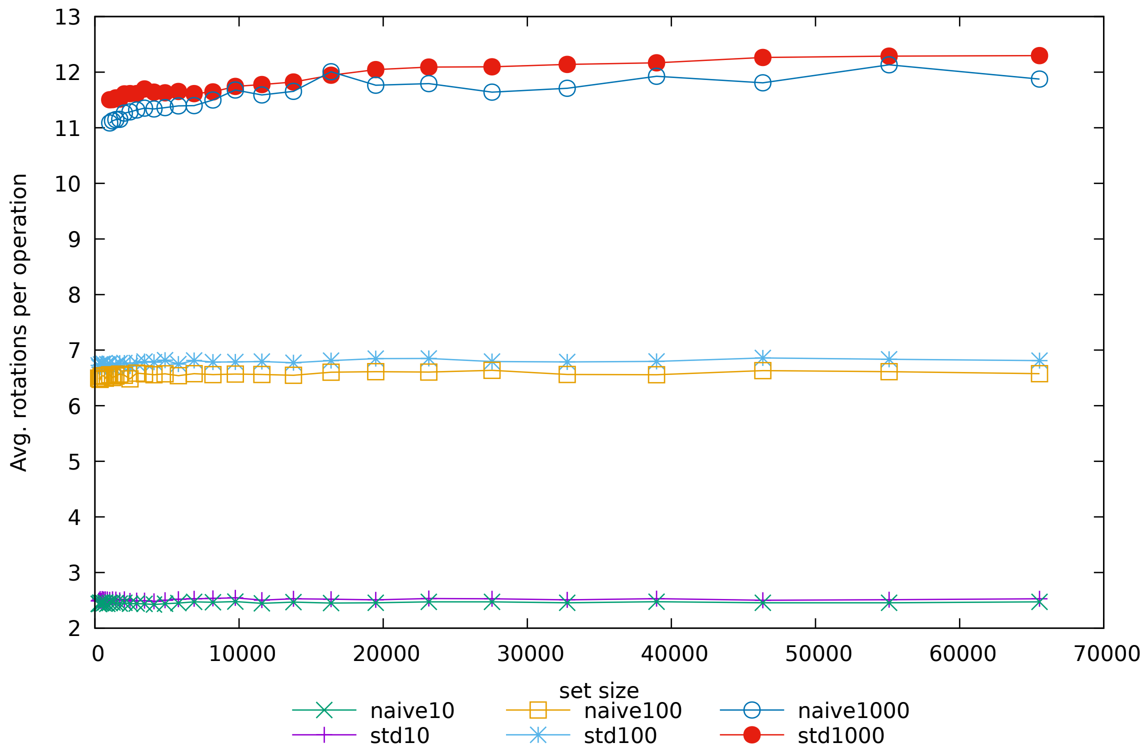
Co nám dá vysvětlení logaritmického počtu rotací je, že v libovolném nevyváženém podstromě budeme pravděpodobněji vyrotovávat prvky z větší (mohutnější) strany (na figuře 3 to je levá (a, A_1, A_2)). Když tedy budeme uvažovat tomu odpovídající rotaci prvku a, tak prvky z A_1 posuneme výš, prvky z B níž a prvky z A_2 budou stále ve stejné výšce, sníží se velikost levého podstromu a pravděpodobnost přístupu k jeho prvkům, opačně s pravým podstromem. Tedy naivní splay stromy se mají tendenci samy vyvažovat. A pak tedy tradičně podle průměrné hloubky.

Figure 3: Rotation



Subset test

Figure 4: Subset test results (seed 80)



V testu s náhodným přístupem do malé množiny prvků (počet v popisku v legendě, křivka jen pro přehlednost) uvnitř velké množiny přidaných opět vidíme velice podobný počet provedených rotací naivní a standardní implementace (pro danou konfiguraci testu).

Dále vidíme, že velikost stromů nemá v tomto testu signifikantní vliv na počet rotací. Naproti tomu velikost navštěvované podmnožiny má stejný vliv jako ona velikost v minulém testu (kde totiž velikost stromu odpovídala velikosti přistupované podmnožiny).

Vysvětlení pro naivní implementaci

Snažíme se vysvětlit, proč nemá velikost nadmnožiny vliv na práci se stromem a důležitá je jen přistupovaná podmnožina.

Uvažujme **"podmnožinovou špičku" stromu** (budeme značit A_i, či neformálně A; i značí počet provedených přístupů), což je v kořeni začínající maximální spojitý podgraf prvků, které jsou v podmnožině. (pozn. A_0 nemá smysl uvažovat, byl by prázdný, ale A_1 je poté alespoň kořen).

Špička A je (z definice) vždy spojitá a obsahuje prvky pouze z podmnožiny, tedy je-li přistupovaný prvek v tomto podgrafu, nikdy ani nesáhneme na prvek, který by v A nebyl (pozn. rotace neuvede na cestu od prvku z A ke kořeni prvek, který by nebyl z A). Tedy zbytek stromu je irelevantní při přistupování k prvkům z A.

Teď jen ukážeme, že prvky A_i jsou i v A_{i+1}. Nikdy nenastane vyrotovávání prvku, který není z A. Tedy jediná relevantní situace je, když vyrotovávaný prvek (nutně z podmnožiny) je těsně pod A (jinak se nic nerozbíjí, protože jsme v nějakém podstromě až dále pod A a na žádný prvek z A v rotacích nesáhneme). První situace, kdy vyrotovávaný prvek (který ještě není v A) manipuluje s nějakým prvkem z A, je, když už v A sám je (neboť manipuluje jen se svým rodičem), tedy jeho rodič je nutně v A. Tedy jakákoliv snaha o rozbití množiny A by nám dala spor.

Předešlé dva odstavce nám dohromady dají, že přistoupíme-li k prvkům z nějaké podmnožiny a poté nepřistoupíme k žádnému jinému a poté opět přistoupíme k prvkům z této podmnožiny, tak již vůbec nesáhneme na prvek, který by nebyl v této podmnožině.

Vysvětlení pro standardní implementaci

Pro standardní splay strom vysvětlení naivní implementace platí jen, když připustíme "skoro spojitost" výše zmíněné špičky A. Tedy připustíme, aby na cestě mezi prvkem v A a jeho prvním předkem z A byl maximálně jeden prvek, který není v podmnožině. Problematické totiž jsou *zigzig* rotace.

zig rotace a zigzag rotace nijak neohrožují špičku A, stále pro ně platí naivní vysvětlení. Zigzag rotace jediná může "Vytrhnout" prvek z A a nahradit jej vyrotovávaným, jenže ten je taktéž z podmnožiny a díky připuštění "skoro spojitosti" je vytrhnutý prvek stále v A.

Maximální vliv nadmnožiny je tedy určitě omezen dvojnásobkem vlivu podmnožiny, tedy nadmnožina asymptoticky významná není a i tohoto dvojnásobku docílíme jen ve velice konkrétních situacích a většinou bude špička A poměrně podobná špičce naivní implementace (pozn. A se nikdy nezhoršuje, pokud se konkrétně přes její hranici neprozigziguje nový prvek A - toto je jediný možný způsob uvedení mezery do A, ostatní rotace ji jistě nezhoršují).

Tyto mezery v A pravděpodobně vysvětlují v grafu viditelný nepatrný rozdíl implementací.