Stříbrná kostra Pokud známe minimální kostru, jak najít druhou nejmenší? Dovolme si kvadratický čas, ale jde to i lineárně (velmi těžké).

Binární Vyhledávací Stromy

Co jsou BVS? K čemu slouží? Kde v nich sídlí data? Co sídlí v uzlech?

Asymptoticky kolik binárních stromů lze vybudovat nad uspořádanou množinou o n prvcích? Asymptoticky kolik jich bude vyvážených, tedy s hloubkou $\mathcal{O}(logn)$ pro nějakou rozumnou konstantu?

Jak dlouho trvá v BVS projití všech hodnot?

Jak rychle (lineárně) vybudovat dokonale vyvážený BVS? Jak z nevyváženého stromu rychle (lineárně) udělat vyvážený?

Jak v lineárním čase slít dva stromy dohromady?

Jak neopak strom roštípnout podle nějaké hodnoty na dva stromy (větších resp. menších hodnot)?

AVL

Jakými invarianty jsou definovány AVL stromy? Jak se tyto invarianty dodržují?

Kolik minimálně uzlů (hodnot) obsahuje AVL strom dané hloubky?

Operace na obecných BVS

Navrhněme rozšíření BVS (s operacemi Find, Insert a Delete) o následující operace. Uvažujme stromy s hodnotami v uzlech vs. pouze v listech a stromy obsahující dvojice (klíč,hodnota) vs. pouze klíče. Zachovávejte kompatibilitu složitost.

Min, Max a Medián celého stromu, $\mathcal{O}(1)$

k-tý prvek stromu, $\mathcal{O}(logn)$

následník a předchůdce klíče, $\mathcal{O}(logn)$

následník a předchůdce prvku daného odkazem, $\mathcal{O}(1)$

k-tý následník klíče, $\mathcal{O}(min(k, logn))$

Min, Max a Medián na intervalu klíčů (a, b), $\mathcal{O}(logn)$

Navrhněte implemntaci struktury z druhého domácího úkolu.

Domácí úkol

Mějme obyčejný BVS, který drží dvojice (klíč,hodnota) ve všech uzlech, a podporuje operace $Insert(key, val)$, $Delete(key)$, $Find(key)$ v čase $\mathcal{O}(logn)$. Dvojice jsou tedy uvnitř BVS seřazeny dle klíčů. Můžeme si představovat, že se jedná o AVL strom. Pro úplnost přepokládejme, že vložení již existujícího klíče vždy přepíše jeho hodnotu a tedy nemáme duplicity.

První část

Rozšiřte tento strom o operaci $Avg(key_1, key_2)$, která v čase $\mathcal{O}(logn)$ vydá průměr hodnot, které jsou v BVS uloženy pod klíči z intervalu $[key_1, key_2]$.

Hint: průměr jako takový udržovat nelze (rozmyslete proč), ale můžeme udržovat jiné pomocné údaje.

Zachovejte asymptotickou složitost ostatních operací. Popište stručně jaké úpravy je potřeba provést v podobě stromu, a jaké úpravy musíme zanést do již existujících operací. Můžete se odkázat na věci, které jsme dělali na cvičení. Pro jednoduchost můžete předpokládat, že klíče key_1, key_2 uvnitř BVS vždy existují.

Druhá část

Rozšiřte původní strom (bez operace z první části) tak, aby podporoval operaci $Increase(key_1, key_2, \delta)$, která v čase $\mathcal{O}(logn)$ zvýší všechny hodnoty, které jsou v BVS uloženy pod klíči z intervalu $[key_1, key_2]$ o δ (může být i záporné).

V daném čase sice můžeme vyhledat okraje daného intervalu, ale nemůžeme všechny hodnoty intervalu skutečně zvýšit. Použijeme proto líné vyhodnocování a zvýšení budeme provádět pouze implicitně, tedy tak aby se ostatní operace chovaly jako kdyby byly hodnoty zvýšené. Bude proto třeba na některé vnitřní uzly stromu pověsit poznámky o neprovedených operacích (tedy něco jako "v tomto podstromu má být všechno posunuto o δ").

Nezapomeňte, že ve stomu bude potřeba provádět rotace a podobné úpravy, které jsou součástí základních operací. Před každou takovou operací víme na které uzly budeme sahat a bude třeba z nich značky nějak odstranit. Jak to udělat aniž bychom zvyšovali asymptotickou složitost?

Podmínky vypracování stejné jako v první části.

Bonusová část nebude. Dejte dohromady předchozí domácí úkoly. Můžete zkusit vypracovat předchozí bonusovou úlohu.