جزوه جلسه ده داده ساختارها و الگوریتم

۴ آبان ۱۴۰۰

فهرست مطالب

۱ درخت جست و جوی متوازن قرمز و سیاه (Red-Black) (Balanced Binary Search Tree)

هر درخت قرمز و سیاه ۴ ویژگی اصلی دارد:
۱. هر راس به یک رنگ قرمز یا سیاه میباشد.
۲. معمولا ریشه و برگ ها به رنگ سیاه هستند.
۳. اگر راسی قرمز باشد، والدش حتما سیاه است.
۴. هر مسیری از ریشه به برگ ها، از تعدادی مشخصی راس سیاه میگذرد که به آن تعداد، سیاه ارتفاع درخت (Black Height) میگویند.
هر راس در این ساختار، اگر دو فرزند نداشته باشد، برای تکمیل درخت راس NIL به عنوان فرزند میپذیرد تا تعداد فرزندانش ۲تا شود. رئوس NIL به رنگ سیاه هستند ولی در شمارش سیاه ارتفاع محاسبه نمیشوند. همچنین این رئوس در محاسبه ارتفاع درخت، درنظر گرفته نمیشوند.
نکته دیگر در مقایسه درخت RB و AVL حافظه اضافی مورد نیاز برای ذخیره سازی اطلاعات میباشد. در درخت های AVL هر راس علاوه بر کلید، ارتفاع خود را نیز ذخیره میکرد که این عدد را میتوان یک int در نظر گرفت. اما در درخت RB هر راس، تنها نیاز داره رنگ خود را در یک بیت ذخیره کند که این مورد یک نقطه قوت برای BR درنظر گرفته میشود.

۱.۱ ارتفاع درخت Red-Black

اگر ارتفاه درخت را با h نشان دهیم، ادعا میکنیم:

$h \leq 2log(n+1)$
برای اثبات ابتدا رئوس قرمز را در رئوس سیاه درج میکنیم؛ بدین صورت که کلید رئوس قرمز را در والد سیاهشان درج میکنیم. در اینصورت،، هر راس سیاه یک، دو یا سه کلید را شامل میشود. در درخت جدید به وجود آمده که تمام رئوس سیاه هستند، هر راس، دو یا سه یا چهار فرزند دارد. ارتفاع این درخت برابر سیاه ارتفاع درخت اولیه میباشد.
حال اگر سیاه ارتفاع را با bh نشان دهیم، بنابر ویژگی سوم درخت قرمز سیاه داریم:

$bh \geq h/2$
میدانیم در درخت جدید حداقل تعداد فرزندان ۲تاست، پس:

$n+1 \geq 2^{bh} \implies bh \leq log(n+1)$
$bh \leq log(n+1)$ and $h/2 \leq bh \implies h \leq 2log(n+1)$

۲.۱ عملیات تعریف شده برای RB Tree

عملیات find _next() و find _prev() به سادگی و مانند درخت AVL انجام میشود.
برای عملیات درج و حذف نیز باید بعد از درج و حذف، رنگ رئوس تغییر کند. توجه شود که عملیات حذف به دلیل پیچیدگی بیش از حد، بررسی نمیشود.

۱.۲.۱ insert(key, value)

ابتدا راس را درج میکنیم و سپس رنگ آنرا قرمز میکنیم. دراینصورت لزوما ویژگی سوم برقرار نیست؛ برای حل این مشکل به رنگ والد و عموی راس درج شده نگاه میکنیم. اگر هردو قرمز باشند، رنگ هردو را با رنگ پدر بزرگ عوض میکنیم. اگر مشکل رنگ ها حل نشده بود، همین کار را برای رئوس پدر بزرگ به بالا مجدد انجام میدهیم.
اگر به مرحله ای رسیدیم که نتوانستیم رنگ هارا تغییر دهیم، از دوران استفاده میکنیم. منطق دروان و تغییر رنگ، همانند منطق دوران و تغییر کلید در درخت AVL میباشد.

۳.۱ نکاتی کلی درمورد درخت های RB و AVL

۱. درخت های RB میتوانند ارتفاع بیشتری نسبت به درخت های AVL داشته باشند اما همانطور که در ابتدا ذکر شد، حافظه کمتری مصرف میکنند.
۲. هرچه ارتفاع درخت کمتر باشد، عملیات جست و جو سریعتر و درج کند تر انجام میشود و بالعکس
۳. in-order درخت قرمز و سیاه مانند درخت AVL مرتب شده میباشد.