**İkili Arama Ağacı Liste Üzerinde Gerçekleştirim**

* İkili arama ağacında sürekli sayıları ikiye böldüğümüz için arama logaritmik düzeye düşüyordu.
* Dizi üzerinde taşma olabiliyordu ve yeni bir dizi oluşturmak gerekiyordu. Ayrıca veri silindiği zaman bütün veriler bir üste taşınmak zorunda kaldığından performans kaybı oluyordu

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Dizi ve listeyi dolaşmak istediğimiz zaman 0. elemandan başlayıp sona kadar gidiyorduk. Ağacı ise dört tane dolaşma yöntemi vardı. İnorder, preorder, postorder, levelorder. Bütün mesaj düğümün kendisine verilir.
* Çocuklar okunurken her zaman önce sol sonra sağ çocuk okunur.
* İnorder okursan oluşacak dizi küçükten büyüğe doğru sıralı olur

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

İkili arama ağacının bunlarla pek ilgilisi yok ama arandıktan sonra yazdırılmak isteniyorsa bunlar kullanılabilir.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Her düğüm üç kısımdan oluşur. Veri, sağ çocuk, sol çocuk
* İkili arama ağacının amacı aramayı hızlandırmak. Arama hızını O(n)’den O(logn)’ e düşürebilmektir. Her seferinde yarısından kurutuluyorsun. Ama her seferinde yarısından kurtulmak tam dolu bir ağaçta olur.
* Logn’ı garantileyen ağaç red black ağaçtır.

**AVL Ağacı**

* Avl ağacının ikili arama ağacından tek farı dengeleme işleminin olmasıdır.
* Ağacın dengesinin bozulup bozulmadığını kontrol eder. Bozulmuşsa dengeleme işlemi yapar.
* Avl ağacında kök değişebilir. İkili arama ağacında ise ilk sayı her zaman köktür
* Avl ağacında bütün düğümlerin yerleri değişebilir.
* Düğüm eklendikçe oynamalar olur. Çünkü her karşılaştırmada yarısından kurtulmayı amaçlıyoruz. Bu dengenin bozulmaması lazım. O(logn)’ e çok yaklaşmak ya da garantilemek amaçtır
* Aynı şey silerken de geçerlidir. Düğüm silerken yine dengeleme olayına dikkat etmen lazım. Silme ve ekleme mantığı aynıdır
* İkili arama agacı için en kötü durum verinin sıralı gelmesidir. Büyükten küçüğe ya da küçükten büyüğe. Çünkü ağaç o zaman dengesiz olur. Listeye döner, sıralanır ve performansı O(n)’e gelir. Her aramada bir düğümden kurtulursun. Ağacın yarısından kurtulamazsın. İşte bu durumlar için redblack ağaçları tasarlanmıştır

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* İkili arama ağacının en iyi durumu tam dolu durumudur. Ve performans O(logn)’dir.



**İkili Arama Agacında Metotlar**

isEmpty -> dizi ve liste aynı. Anlık kontrol eder.ikisinde de eleman sayısı tutabilirsin

Ekleme -> dizi ve liste aynı. İkisinde de eleman sayısı oranında inmemiz gerek çünkü neresi boş bilmiyoruz

Sil -> bağıl. Çünkü dizide tüm elemanlar bir seviye yukarı çıkar. Ve yer kaplama bağıl da yine daha avantajlı

Dolaşma işlemleri -> ikisinde de aynı. (post, in, pre aynı level order dizi de avantajlı)

Yükseklik -> aynı

Arama -> aynı (avl gibi bir işlem yapmıyorsan farketmez)

Temizleme -> dizi avantajlı. Çünkü index doluluğunu direkt 0 yapıyoruz. Bagılda çöp oluşmaması için süreli solun en sağına falan iniyorsun

* Ekle her ikisinde de aynı gibi dursa da sürekli yer açtığın zaman dizi dezavantajlı olur.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**NOT**: Heap Agacında mutlaka dizi üzerinde yapılır. çünkü orada ebeveyne çıkıyorsun ve ebevyn ile çocuk sürekli değiştiriliyor falan pointerlar ile uğraşılmaz. Ama avl ağacı dizide gerçekleştirilmez. Çünkü sürekli dengeleme işlemi yapılıyor. Dengeleme demek düğümün sürekli yere değiştirmesi demek. Yani avl deki amaç ikili arama ağacının asıl performansına erişmek.

Ödevde şablon yapı içinde kişi nesnesi tutacaksın düğümler sürekli yukarı aşağı ineceği için iki tane yığıt yerleştireceksin. Biri her düğümün yükseklik değişikliğini tutması için diğeri de derinlik değişikliklerini tutması için yığıt pointeri eklemelisin. Sınıf içindeki tüm nesnelerin new ile heapte oluşturman lazım. Ve büyük dosyalarda deneme yap. Normalde ikili arama ağacında eşit veri olmaz ama özel senaryolarda bu gerekir. Ödevde de eşit veri olayı olabilir. Küçük her zamanda solda bulunmayabilir. Çünkü dengeleme işlemlerinde belli olmaz. İlk eklerken sola ekliyoruz ama eşit geldiğinde dengelem işlemi sağa atayabilir. Ama önemli değil. Önemli olan ilk sola atanmasıdır. Whatttt?