**Ali Yasin Eser**

**14044058**

**CSE222**

**Homework 5**

**Part 1::**

**Problem çözümüne yaklaşım:**

Towers of Hanoi’nin algoritmasını kavramak için önce oynamam gerekiyordu. Kuralları temel alarak onlarca, belki yüzlerce kez oyunu oynadım. Farkettim ki minimum hamle için disk sayısı tek ve çift olduğunda bir etki söz konusu. Farklı peglerden başlamak gerekiyor. Bunu not alarak minimum hamle için yapmam gerekenleri yazdım. Her aşamada bir tekrar söz konusuydu. Değişen, sadece a peginden b’ye mi, yoksa tam tersi mi olduğuydu. Bunun durumu da oyuna başlarken okuduğum kuralda, küçük diskin üzerine büyük koyamamamda yatıyordu. Tüm bunları ele aldığımızda:

1. Küçük disk üstüne büyük gelmeyecek
2. Hamle geri alınmayacak
3. 3 hamlede bir tekrar var, sadece yön değişikliği mevcut
4. Amaç tüm diskleri doğru sırada destination pege yerleştirmek
5. Disk sayısına göre işlem sırası değişiyor.
6. Her aşamada tek bir disk hareket edebilir.

**Test cases:**

Source pegdeki tüm elemanların doğru sırada destination pege yerleştirilmesi gerek.

Oyunun kuralları sağlanmalı, test case olarak bu ele alındı. Oyunun kurallarını sağlamak için gerekli disk boyutu karşılaştırılmaları ve büyük diskin küçük olanın üstüne gelemeyeceği gibi testler düşünüldü. Her aşamada hedefe ulaşılıp ulaşılamadığı kontrol edildi.

**Part 2::**

**Problem çözümüne yaklaşım:**

Problem LinkedListRec(kitaptan alındı) class’ında remove metodunun ilk gordugu elemanı silip bırakması değil de tüm kopya elemanları silmesiydi. Diğer bir deyişle, verilen bir eleman listedeki tüm yerlerden silinmeliydi. Kitaptaki kodda ilk eleman kontrolü vardı. Devamını sağlayacak şekilde değiştirdim ve bulduğunda direk olarak return etmesi yerine hem bir return değeri tutması ve recursion operasyona devam etmesini sağladım. O anki recursive metodun donuş değeri ile recursion olacak metodun sonucunu bir veya bağlacı ile bağladım. Eger once return true olsa direk dondurme yapacaktı. Bu sebeple true değerini saga alarak recursive işleme devam etmesini sağladım.

**Test cases:**

İlk eleman silinmeli ve daha sonra devam edilmeli. Verilen parametredeki eleman listede kac kere geçtiyse her yerden silinmeli. Eleman kaybı yaşanmamalı.

**Part 3::**

**Problem çözümüne yaklaşım:**

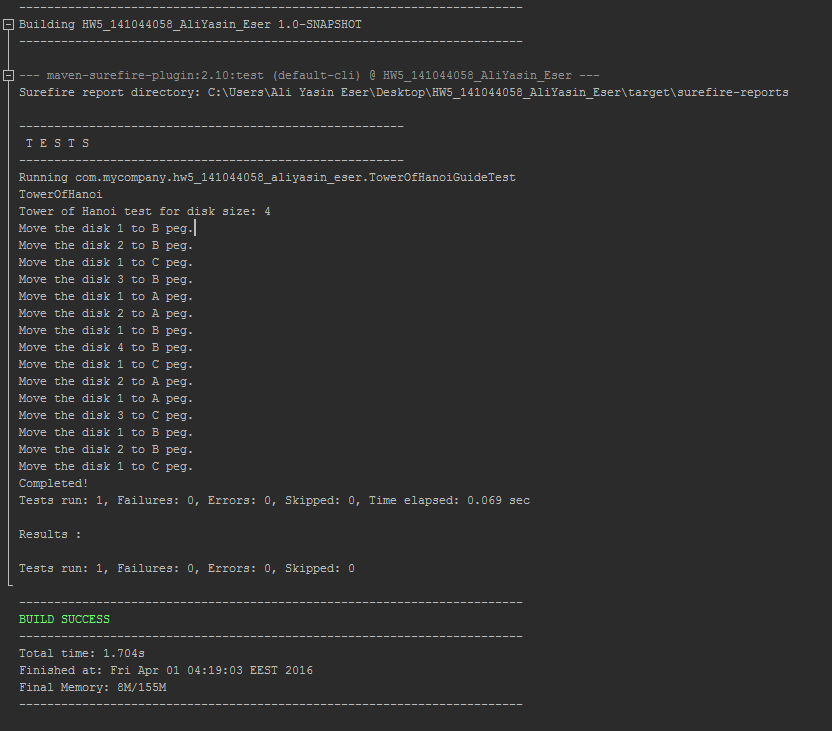
Problem iki sorted list bulunan class oluşturup aralarında recursive küme işlemleri gerçekleştirmekti. Listeleri barındırma işi kolaydı, operasyonlar recursive olarak hazırlanacağından base case design gerektiriyordu. isSubset yazarken listeyi sürekli küçülterek ilk elemanı kontrol etme işlemi gerekli recursive durumu sağlıyordu. Listede en son eleman kalmayacaktı çünkü subList metodu empty list return edecekti. Intersection ve Union aynı şekilde çalışmalı fakat içeriği farklı olmalıydı. Onlar için ekstra bir liste oluşturdum çünkü dondurmem ve recursive içinde sürekli oluşmaması için ilk gönderildiğinde oluşması gerekliydi. Daha sonra recursion metoda yolladım. Bu şekilde çalışıyordu, intersectionda bir listeyi kısaltıp ikinci listeyle aynı olanları ekliyor, unionda bir listeyi direk return edilecek listeye ekliyor ve ikinci listeyi kısaltarak kontrol ediyordum. Bu şekilde class sorunsuz çalışıyor. Sorted durumu sağlamak için eleman ekleyen ve sort eden metodlar yazdım. Class ve gerekli metodlar bitti.

**Test cases:**

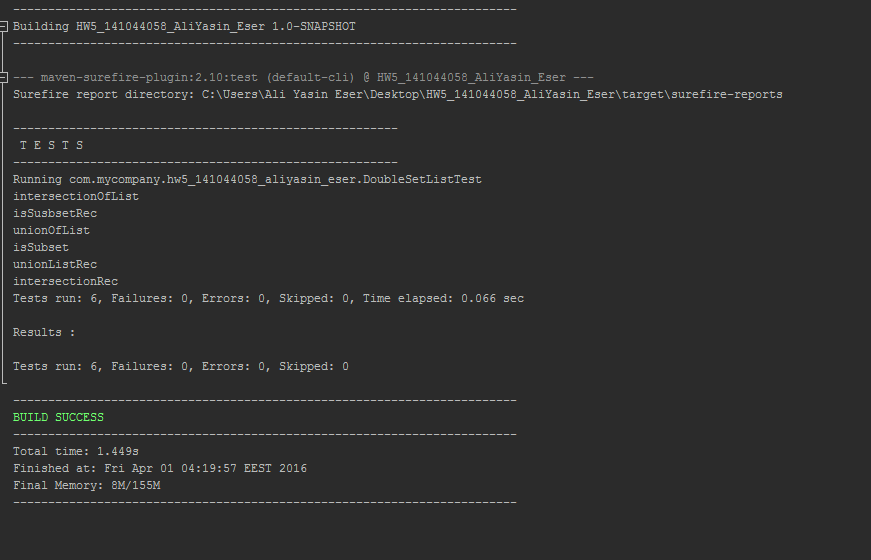
List sorted olmalı.

Intersection, union ve subset metodları çalışmalı.

**Part 1 Test:**



**Part 2 Test:**



**Part 3 Test:**

