Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

HOMEWORK 7 REPORT

FÜRKAN YILDIZ
141044031

Course Assistant: ŞEYMA YÜCER

1. Problem Solutions Approach

Q1)

Önce Enrty tipinde Binary Search Tree oluşturuldu fakat bu entry'nin Comparable olması gerektiğinden (Binary Search Tree yalnizca comparable objeler ile instance edilebilir) Map.Enrty ve Comparable interfacelerinden özel bir myEnrty class'ı implement edildi. Bu class key ve value'den oluşuyor, Comparable sıfatından dolayı compareTo methodu override edildi, bu method iki adet key alıyor ve eğer BinaryNavMap classımıza comparator verilmiş ise comparator ile verilmemiş ise compareTo ile bu iki key'i karşılaştırıyor. Class'a comparator verilebilmesi ve bu verilen comparator ile karşılaştırma yapılabilmesi için comparator data fieldi eklendi, buna göre constructureler yazıldı. Boş olan methodları implement edebilmek için ise Navigable Map'i bir sıraya koymak gerekti. BinarySearch Tree alt yapıda bunu yapıyor. Bize düşen ise sıralı bir şekilde bu treeden elemanları çekebilmek. Bunu yapabilmek adına in-order iterator yazıldı Navigable Map için, zira Binary Tree de in-order traverse elemanları küçükten büyüğe sıralı bir şekilde verir.

Implement edilecek methodlarda ise tamamen bu iterator kullanıldı, örneğin yol gösterici olması için bazı methodlar aşağıda açıklanmaktadır:

floorEntry methodu verilen key'e eşit yada ondan küçük en büyük map'i return eder, bu method yazılırken Treede iterator ile tek tek gezildi (bu gezilme treedeki elemanlara göre küçükten büyüğe doğru yapılıyor) ve gezilen her eleman methoda gelen key ile karşılaştırıldı bu karşılaştırma sonucu treede gezdiğimiz key değerinin, parametre olarak gelen key değerinden büyük olduğu an tree'yi dolaşma sonlandırıldı ve son next yapılarak elde edilen elemandan bir önceki eleman return edildi. floorKey'de ise bu method çağırılarak key'i return edildi.

ceilingEntry methodu verilen elamandan büyük olan en küçük değeri yada kendisini dönderir. Bunu sağlayabilmek için yine iterator ile tree gezildi ve iteratorden alınan değer, aradığımız değerden(parametre olarak gelen değer) büyük yada eşit olduğu an o değer return edildi.

Bu şartlara uyulmuyorsa yani örnek vermek gerekirse girilen elemandan daha küçük yada ona eşit eleman yoksa tree'de exception fırlatıldı.

ceilingKey 'de ise yine ceilingEntry methodu çağırılarak key'i return edildi.
firstEntry methodunda 1 kere next çağırılarak return edildi,
lastEntry'de ise tüm tree sonlanana kadar next çağırılarak en son gelen eleman return edildi.
pollFirstEntry methodunda, firstEntry methodu çağırılarak ilk eleman alındı ve ardından bu değer ile Binary Search Tree'nin delete methodu kullanıldı.

subMap methodunda önce fromKey ve toKey parametreleri karşılaştırıldı, eğer fromKey, toKey'den büyükse exception fırlatıldı. Öyle bir durum yok ise, iterator oluşturuldu ve tree deki elemanlar tek tek fromKeyle to Key arasında mı, fromKey'e yada toKey'e eşit mi (fromInclusive ve toInclusive parametrelerine göre) diye karşılaştırma yapıldı ve bu şartı sağlayanlar için yeni bir BinaryNavMap oluşturularak return edildi.

headMap, baştan verilen key'e kadar geleceğinden subMap mehoduna firstKey verilerek çağırıldı.
tailMap verilen key'den son key'e kadar geleceğinden subMap mehoduna lastKey verilerek çağırıldı.

Q2)

HashTableChaining classı bir HashtableOpen table'ı tutmalı, bu sebeple HashtableOpen adında bir class oluşturuldu.

HashtableOpen classında Enrty arrayi tutuldu, başlangıç kapasitesi olarak 101, load threshold olarak 0.75 seçildi. Silinen elemanların yerine koymak için DELETED entry data fieldi yazıldı. Aktif olan maplarin ve silinen maplarin sayısını tutmak için integer değerler tutuldu. Size methodunda bu akif olan (yani silinmemiş, tabloda hala değeri olan) maplarin sayısı return edildi. Find metdodu yazıldı, bu methodda gelen key'in hasCode'si çağırılarak table size'a göre modu alınıyor ve indexine bakılıyor eğer bu index tabloda dolu ise ilk boş index bulunana kadar arttırılıyor. Bu method put methodunda kullanılmak üzere yazıldı. Zira put methodunda find methodu çağırılıyor ve ondan dönen index değerine yeni bir entry oluşturuluyor verilen key ve value ile. Bunun hemen ardından ise hangi index'e ekleme yapıldığını indexList fieldine ekliyor. Bu listeyi toString methotunu yazarken kullanacağız, tablonun dolu olmayan yerlerini gezmeyeceğiz sadece dolu olan yerleri stringe çevirip return edeceğiz. Put methodunda ekeme yapıldıktan hemen sonra loadFactor hesaplanıyor ve eğer 0.75 den büyük ise load factor rehashing yapılıyor. Rehashing'de eski table temp bir yere atılarak, eskisinin 2 katından 1 fazla capasiteye sahip yeni bir table oluşturuluyor ve ardından aski tabledeki null olmayanlar ve silinmeyenler yeni tabloya put methodu yardımıyla ekleniyor. Remove ve get methodlarında ise verilen key'in tabloda bulunan ilk (eklenen ilk map'a karşılık geliyor) mapi ile işlem yapılacağından FindFirst adında bir method yazıldı, bu method tabloyu baştan sona dolaşarak aranan key'i ilk bulduğu gibi onun indexini return ediyor. Get methodunda bu findFirst methodunu çağırarak, ulaşılmak istenen key'in ilk value'si return ediliyor. Remove metodunda ise yine findFirst methodu çağırılarak, , ulaşılmak istenen key'in ilk indexi bulunuyor ardından tablodaki o indexe data field olarak tutulan DELETED koyuluyor, ve toString için tutmuş olduğumuz index listesinden o index çıkartılıyor.

HashTableChaining classında, HashtableOpen içeren bir arrayList tanımlandı data field olarak HashTableChaining classının içerisinde ve bu data field için constructurede yer alındı. Başlangıç kapasitesi olarak 101, load threshold olarak 3 seçildi.

Get methodunda, verilen key'in hastCode'sinin capasiteye göre mod'u alınarak, HashTableChaining table'sinde hangi index'te olduğu bulundu ardından eğer bu index null değilse HashTableChain'in o indexi içerisinde bulundurduğu HashtableOpen class'ının get methodu çağırıldı, böylece aradığımız key değerinin ilk value'si return edildi. (birden fazla aynı key'den olabilir çünkü)
Put methodunda önce load factor hesaplandı, eğer HashTableChaining'nin size'ının 3 katından büyük ise rehash yapıldı. Daha sonra mod işlemi ile girilen key'in HashTableChaining ' arrayinde hangi index'te olacağı hesaplandı ve eğer o index boş ise o HashtableOpen arrayi için yer alındı ve HashTableChaining arrayinde nereye ekleneceği bulunan elemanın indexindeki HashtableOpen classinın put methodu çağırıldı verilen key ve value ile.

Deelte methodunda ise put methodunun tam tersi yapıldı, HashTableChaining arrayinde keyin indexi bulunduktan sonra HashtableOpen 'ın remove methodu çağırıldı.

2. Test Cases

Q1)
NavigableMap oluşturularak, 11 adet il ilçe put edildi,
-son eklenen ilçe ekrana bastırıldı, put methodunun return'ünü test etmek için.
- NavigableMap bastırıldı.
- NavigableMap'in size'ı bastırıldı.
- NavigableMap'in lower entry'si - key'i bastırıldı.
- NavigableMap ceiling entry'si - key'i bastırıldı.
- NavigableMap higher entry'si - key'i bastırıldı.
- NavigableMap first entry'si - key'i bastırıldı.
- NavigableMap last entry'si - key'i bastırıldı.
- NavigableMap descendingMapi bastırıldı.
- NavigableMap navigableKeySet'i bastırıldı.
- NavigableMap descendingKeySet'i bastırıldı.
- NavigableMap headMap'i bastırıldı.
- NavigableMap tailMap'i bastırıldı.

-Exception gelmesi için ilk değeri büyük ikincisi küçük olan subMap yazıldı ve exception yakalandı.
-subMap'in geçerli şartlar sağlandığında ekleyip eklemediğini test etmek için gerçeli değerler verilerek subMap tekrar test edildi.

- NavigableMap pollFirstEntry'i ve ardından kendisi bastırıldı.
- NavigableMap pollLastEntry'i ve ardından kendisi bastırıldı.

-Önceden map'in içerdiği key değeri ile farklı bir value değeri verilerek değişip değişmediği test edildi value'nin.

Q2)
-16 adet ilçe il eklendi.
-son eklenen ilçe ekrana bastırıldı, put methodunun return'ünü test etmek için.
-HashTableChaining bastırıldı.
-HashTableChaining'in size'ı bastırıldı.
-4 farklı değer ile get metodu test edildi. (eklenen ilk value'i return etmeliler)
-2 farklı il ile eklenmiş ilçe (key = "edremit") 3 kez remove edildi onun return valuesi ve ardından HashTableChaining ve size'ı bastırıldı.

3. Running and Results

```
Q1--------------------------------------------------------------------------
biga eklendi: canakkale
The original set odds is {aksaray=istanbul, biga=canakkale, cekirge=bursa, foca=izmir, gebze=kocaeli, kadıkoy=istanbul, kahta=adıyaman, kecıoren=ankara, manavgat=antalya
The original size is 11
[aksaray=istanbul, biga=canakkale, cekirge=bursa, foca=izmir, gebze=kocaeli, kadıkoy=istanbul, kahta=adıyaman, kecıoren=ankara, manavgat=antalya, niksar=tokat, uskudar=
lowerEnrty of kadikoy: gebze=kocaeli
lowerKey of kadikoy: gebze
floorEntry of kadikoy: kadıkoy=istanbul
floorKey of kadikoy: kadıkoy
ceilingEntry of kadikoy: kadıkoy=istanbul
ceilingKey of kadikoy: kadıkoy
higherEntry of kadikoy: kahta=adıyaman
higherKey of kadikoy: kahta
The first entry is aksaray=istanbul
The first key is aksaray
The last entry is uskudar=istanbul
The last key is uskudar
The Descending Map is {aksaray=istanbul, biga=canakkale, cekirge=bursa, foca=izmir, gebze=kocaeli, kadıkoy=istanbul, kahta=adıyaman, kecıoren=ankara, manavgat=antalya, n
The Navigable Key Set is [aksaray, biga, cekirge, foca, gebze, kadıkoy, kahta, kecıoren, manavgat, niksar, uskudar]
The Descending Key Set is [aksaray, biga, cekirge, foca, gebze, kadıkoy, kahta, kecıoren, manavgat, niksar, uskudar]
The Head Map is {aksaray=istanbul, biga=canakkale, cekirge=bursa, foca=izmir }
The Tail Map is {gebze=kocaeli, kadıkoy=istanbul, kahta=adıyaman, kecıoren=ankara, manavgat=antalya, niksar=tokat, uskudar=istanbul }
IllegalArgumentException handled for subMap opearation.
Poll first entry aksaray=istanbul
After poll first entry {biga=canakkale, cekirge=bursa, foca=izmir, gebze=kocaeli, kadıkoy=istanbul, kahta=adıyaman, kecıoren=ankara, manavgat=antalya, niksar=tokat, usku
Poll last entry uskudar=istanbul
After last first entry {biga=canakkale, cekirge=bursa, foca=izmir, gebze=kocaeli, kadıkoy=istanbul, kahta=adıyaman, kecıoren=ankara, manavgat=antalya, niksar=tokat }
kecıoren'in value'sini değişirsek: newCity
kecıoren'in value'sini değişirdikten sonra: {biga=canakkale, cekirge=bursa, foca=izmir, gebze=kocaeli, kadıkoy=istanbul, kahta=adıyaman, kecıoren=newCity, manavgat=anta
```

```
Q2--------------------------------------------------------------------------
Biga eklendi, return value: canakkale
{{edremit=balikesir, edremit=van }{kemalpasa=bursa, kemalpasa=izmir }{ortakoy=istanbul, ortakoy=aksaray, ortakoy=corum, biga=canakkale }{kecıoren=ankara }{pinarbasi=kas
Size: 16
Biga icin girilen ilk value: canakkale
Eregli icin girilen ilk value: konya
Ortakoy icin girilen ilk value: istanbul
Edremit icin girilen ilk value: balikesir
Edremit' in key olduğu ilk ikili siliniyor: : balikesir
Edremit' in key olduğu ilk ikili silindikten sonra: : {{edremit=van }{kemalpasa=bursa, kemalpasa=izmir }{ortakoy=istanbul, ortakoy=aksaray, ortakoy=corum, biga=canakkale
After deleted size: 15
pinarbasi' in key olduğu ilk ikili siliniyor: : kastamonu
pinarbasi' in key olduğu ilk ikili silindikten sonra: : {{edremit=van }{kemalpasa=bursa, kemalpasa=izmir }{ortakoy=istanbul, ortakoy=aksaray, ortakoy=corum, biga=canakka
After deleted size: 14
Edremit' in key olduğu ilk ikili(ikinci kez) siliniyor: : van
Edremit' in key olduğu ilk ikili(ikinci kez) silindikten sonra: : {{kemalpasa=bursa, kemalpasa=izmir }{ortakoy=istanbul, ortakoy=aksaray, ortakoy=corum, biga=canakkale
After deleted size: 13
Edremit' in key olduğu ilk ikili(ucuncu kez (Mapta yer almıyor)) siliniyor: : null
Edremit' in key olduğu ilk ikili(ucuncu kez (Mapta yer almıyor)) silindikten sonra: : {{kemalpasa=bursa, kemalpasa=izmir }{ortakoy=istanbul, ortakoy=aksaray, ortakoy=co
After deleted size: 13
Your tests is done. Make sure that you test all methods of class!!

Process finished with exit code 0
```