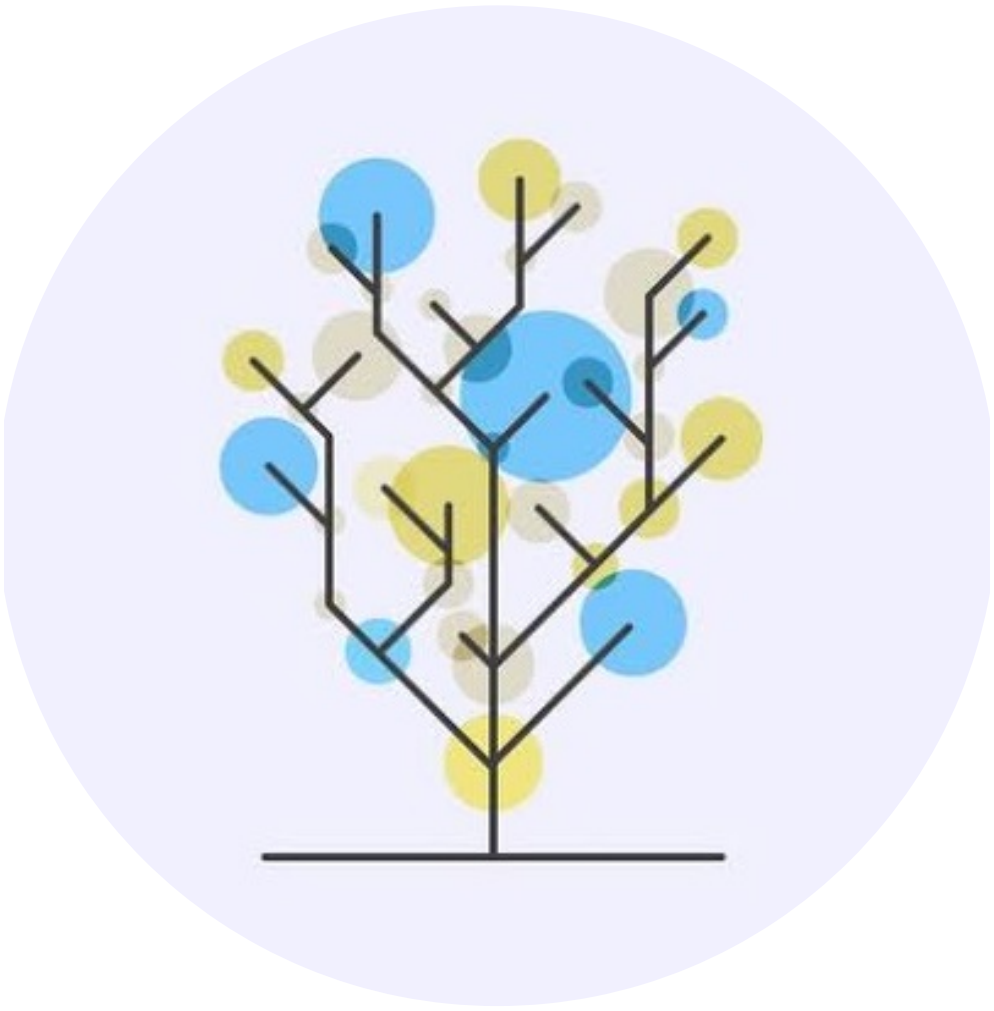

Arrays

03.01.2020

Data Structures & Algorithms

Intro

- Pandas is one of the most popular Python libraries for Data Science and Analytics.
- Arrays or sometimes called Lists, organizes items sequentially.
- One item after another in memory.
- Since data is stored in contiguous memory. Arrays have the smallest footprint of any data structure.
- If all we need is store some data and iterate over it, one by one, step by step; arrays are the best choice

Different Actions and Big Os

ACCESS - O(1)

- Bilgisayar array içindeki elemanların hepsinin adresini biliyor olarak düşünülebilir.
- Bu yüzden eğer bir elemana ulaşmak istersek (indexing) bu operation için constant time yani O(1) gerekir.

APPEND - O(1)

- Append ile arrayin en sonuna elaman ekliyoruz, bunu da yapmak aynı access gibi 1 operation alıyor çünkü arrayin en son adresini biliyoruz bir sonraki içine bir değişken yerleştirmiş oluyoruz o kadar.

POP - O(1)

- Append'in tersi gibi düşünülebilir. En sona eleman eklemek yerine en sondaki elemanı siler.
- Bu operasyonun da time complexity'si O(1) olur. Çünkü ilgili adresi biliyoruz tek yaptığımız bu adrese git şu item'i remove et demek!

`INSERT - O(n)`

- Arrayin en başına bir eleman eklemek O(n)
- Çünkü arrayin her elemanı artık yerinden ediliyor, 0. index'e yeni eleman oturuyor, eski sahibi 1. indexe geçiyor, 1'in eski sahibi 2'ye geçiyor şeklinde.
- Kısaca her array element ile işlem yapılıyor bu yüzden de O(n).
- Ayrıca Insert ile başka bir index'e de eleman ekleyebiliriz ancak farketmez bu da O(n) çünkü biz worst case scenario düşünürüz.

`REMOVE - O(n)`

- En baştan veya sonraki indexlerden eleman silinirse, diğer elemanların kaydırılması gerekecek ve en kötü durumu düşünürek buna O(n) deriz.

Low Level Arrays & Dynamic Arrays in Python

- At this point, it might help to check this link out.

Dynamic vs Static Arrays

- Static arrays are fixed in size.Arrayin tutacağı eleman sayısının baştan belirtilmesi gerek.
- Array elemanları ardarda dizildiği için ben en başta 7 elemanlık yer açtıysam 8. elemanın boş olduğunun garantisi yok.
- Dynamic arrays expands as we add more elements. Bu yüzden boyutunu baştan belirtmemize gerek yoktur.
- Hafıza dolunca elemanlar, RAM içinde daha geniş bir alana taşınır.
- Javascript, Python vb. gibi high level dillerde memory allocation olayı otomatik olarak yapılır array dynamic array gibi çalışır.

- **`WARNING:`** Burada önemli bir detay şu, push methodu bazen **O(n)** olabilir. Çünkü mesela 4 elemanlık bir array yarattık sonra 5. yi ekliyoruz ama 5. için yer yok o zaman dynamic array olduğu için bilgisayar kendi kendine tüm elemanları kopyalıyor 4*2=8 elemanlık bir boşluğa götürüyor ve yapıştırıyor sonra yeni elemanı ekliyor. Yani eleman sayısı kadar boşluk açılmış oluyor. Ama bu yaşanırsa tüm elemanları gezdiği için time complexity O(n) oluyor!

Implementing an Array

JAVASCRIPT

```
// Let's understand how to Build an Array and How to use it
// Array class'ı oluşturacağız.
// Data structures 0'dan oluşturulabilen şeylerdir.
// Most data structures are built on top of other data structures.
//-------------------------------------------------------------

class MyArray {
  constructor() {
    this.length = 0;
    this.data = {};
  }

  get (index){
    return this.data[index];
  }

  push (item){
    this.data[this.length] = item;
    this.length++;
    return this.length;
  }

  pop(){
    const lastItem=this.data[this.length-1];
    delete this.data[this.length-1];
    this.length--;
    return lastItem;
  }

  delete(index){
    const item = this.data[index];
    this.shiftItems(index);
  }

  shiftItems(index){
    for(let i=index; i
```

PYTHON

```
class MyArray:
    def __init__(self):
        self.length = 0
        self.data = {}

    def get(self,index):
        return self.data[index]

    def push(self,item):
        self.data[self.length]  = item
        self.length = self.length+1

    def pop(self):
        lastitem = self.data[self.length-1]
        del self.data[self.length-1]
        self.length = self.length -1
        return lastitem

    def delete(self,index):
        deletitem = self.data[index]
        self.shiftitems(index)
        return  deletitem

    def shiftitems(self,index):
        for i in range(index,self.length-1):
            self.data[i] = self.data[i+1]
            i= i+1
        del self.data[self.length-1]
        self.length = self.length -1


newarray = MyArray()
newarray.push('hi')
newarray.push('hey')
newarray.push('wow')
newarray.delete(1)
print(newarray)
```

Arrays Pros and Cons

```
+Fast lookups
+Fast append/pop
+Ordered

-Slow inserts
-Slow deletes
-Fixed size for static arrays(avoidable by dynamic arrays)
```

More blogs