

***Öğrenci No:*** 20181121011

***Öğrenci Adı Soyadı:*** Eyyüp Altun

***Ders:*** İleri Programlama



Python ve R diline göre JS/Node.JS

artı ve eksileri



**Performans**

Bir programlama aracından istediğimiz şey performanstır. Performans, uygulamanızın yanıt hızını tanımlar ve bu da kodunuzun ne kadar hızlı yürütüleceğine bağlıdır. Python ile Node.js performansını karşılaştırırsak Node.js'in daha hızlı olduğunu göreceğiz. Bu performans, Node.js'nin çok hızlı ve güçlü bir motor olan Chrome'un V8'i üzerine kurulu olmasından kaynaklanıyor. Bu nedenle, Node.js'in hızı Python'a oranla çok daha fazladır; bu, gerçek zamanlı uygulamalar için, örneğin sohbet işlevselliği içeren uygulamalar için daha iyi bir seçim olmasını sağlar.

Ayrıca, Python, yoğun bellek kullanan uygulamalarda optimum performans göstermez. Bu nedenle, ağır 3D grafikler kullanan uygulamalar geliştirmek için tercih edilen bir çözüm olmamalıdır.

**Ölçeklenebilirlik ve Eşzamanlılık**

Backend planlamasında dikkate alınması gereken bir diğer önemli husus ölçeklenebilirliktir. Ölçeklenebilirlik, bir uygulamanın, performans düşürmeden artan sayıda isteklere cevap verebilme yeteneğidir. Bu yetenek içerik ağırlıklı uygulamalarda olduğu kadar hem masaüstü hem de mobil arayüzlerle birden fazla yeni kullanıcıya hizmet edenlerde de gereklidir.

Programlama dili, uygulama ölçeklenebilirliği üzerinde de etkiye sahiptir. Node.js, iş parçacığı dışında tamamlanan I/O işlemleriyle tek iş parçacıklı eşzamansız mimariyi oluşturur ve bu nedenle engellemez. Bu özellik basit web uygulamalarında sorunsuz Node.js ölçeklenebilirliğini garanti eder, ancak aynı anda birçok işlemi içeren karmaşık uygulamaların geliştirilmesi derinlemesine bilgi, dikkat ve titiz mühendislik araştırmaları gerektirir.

Python, sırasıyla asenkron programlamayı desteklemez, ancak eşzamansız işlemeye kolayca erişilebilen koridorları desteklemektedir. Dolayısıyla, mimarisi Nodejs kadar ölçeklenebilir gibi görünmese bile, Python gerekli ölçeklenebilirliğe ulaşılabilecek araçlara sahiptir.

**Hata işleme**

Bir programlama dilini seçerken hata yönetiminin kolaylığı ve şeffaflığı kritik bir konudur. Node.js ve Python, istisnaları atma ve yakalama ile, yani kod yürütülürken oluşan hataları iyi ele alır. Bununla birlikte genel fikir, Python'da istisnaların işlenmesinin daha kolay olduğunu ve hataları ayıklamak ve düzeltmek için daha az zaman almaktadır.

**Öğrenme eğrisi**

Daha önce de belirttiğimiz gibi, Node.js JavaScript kodunu çalıştırmak için bir ortamdır. JavaScript'i biliyorsanız, aynı dili konuştuğu için Node.js ile ilgili çok fazla sorun yaşamayacaksınız

İkisinide bilmiyorsanız ve hangisini öğreneceğinize karar veremiyorsanız, Python öğrenmek daha temiz sözdizimi ve daha kompakt bir kodlamaya sahip olduğu için daha kolay olabilir. Python, aynı sonuca ulaşmak için Node.js'den daha az kod satırı gerektirir. Ayrıca, Python 25 yılı aşkın bir süredir bu piyasanın içinde ve geniş bilgi birikimi içerdiğinden kaynak sıkıntısı çekmezsiniz.

**Sonuç**

Bu küçük araştırmada, Node.js'nin Python'dan daha iyi olduğunu ispatlamayacağımızı ya da tam tersinin olamayacağını söyleyerek başladığımızı belirttik. Nitekim, her ikisinin de uygulama alanlarında belli başlı avantajları vardır. Nodejs ve Python, yazılım geliştirme dünyasındaki yerlerini haklı olarak elde etti. Bir projenin ana teknolojisinin seçimi, uygulama türüne, endüstri sektörüne, performans gereksinimlerine, geliştiricilerin deneyimine vb. bağlı birden çok faktöre bağlıdır. Node.js ve Python, doğru yaklaşımla güvenilir, tutarlı ve etkili çözümler sağlayacaktır.



### **Hangisi daha basit?**

### Python ve JS birbirine oldukça benzer olsa da karmaşıklık (veya daha doğrusu basitlik) açısından ele alındığında Python çok daha basit. Boşuna en basit programlama dillerinden biri olarak görülmüyor. JS öğrenilmesi “zor” bir dil olarak görülmese de Python öğrenmek JS öğrenmeye göre çok daha az çaba gerektiriyor.

### **Hangisi daha uyarlanabilir?**

Sorunun cevabı aslında daha da basit bir soruda yatıyor; hangisi OOP temelli?

Gerçeği söylemek gerekirse bu konuda iki programlama dili de birbirine çok benzer. Yani anlayacağınız ikisi de OOP temelli. Aralarındaki dikkate değer tek fark Python OOP diliyken JS’nin OOP scripting dili olması. Ama bu bir sorun yaratmıyor.

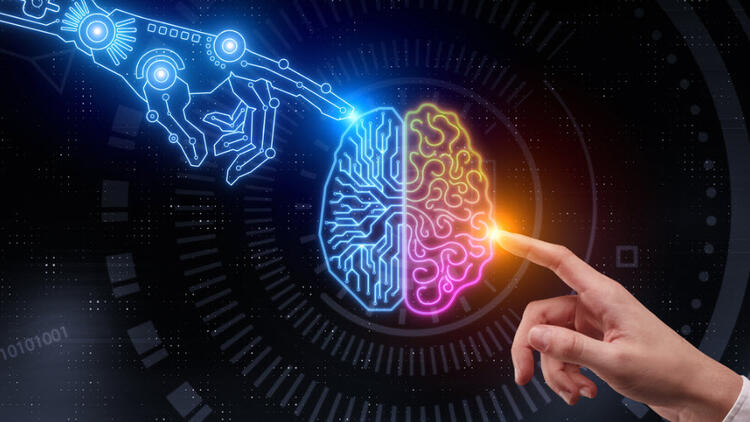
### **Hangisi daha popüler?**

### Ancak, Python zaman geçtikçe daha da popüler bir hale gelmeye başladı. Basit olması ve sahip olduğu topluluk desteği sayesinde ana akımın duvarlarını yıktı. Şu anda dünyada en çok konuşulan programlama dillerinden biri!

## **Son**

Gördüğünüz gibi JavaScript vs Python tartışmasında kesin bir “kazanan” seçmek zor. Python daha popüler ve öğrenmesi daha kolay ama JS’nin de belli alanlarda kendine göre üstünlüğü var. Tamamen kendi kişisel tercihlerinize bakıyor. Kodlama eğitimi alma nedenlerinize göre ikisinden birini tercih edebilirsiniz.

Umarım JavaScript ve Python karşılaştırması yazımız sizin için faydalı olmuş ve size yeni şeyler öğretmiştir! Görüşmek üzere!

Javascript ve Node.JS ile yapay zeka uygulamaları geliştirilebilir mi?

## Yapay Zeka Nedir?

En basit ifadeyle yapay zeka (AI), görevleri yerine getirmek için insan zekasını taklit eden ve topladıkları bilgilere göre yinelemeli olarak kendilerini iyileştirebilen sistemler veya makineler anlamına gelir.Örneğin:

* Sohbet robotları, müşterilerin sorunlarını daha hızlı bir şekilde anlamak ve daha verimli cevaplar vermek için yapay zekadan yararlanır
* Akıllı asistanlar, zamanlamayı iyileştirmek için büyük kullanıcı tanımlı veri kümelerinden kritik bilgileri çekmek için yapay zekadan yararlanır
* Öneri motorları, kullanıcıların izleme alışkanlıklarına göre TV programları için otomatik öneriler sunabilir

Yapay Zeka, herhangi bir özel biçim veya işlevden ziyade süper güçlendirilmiş düşünce ve veri analizi yeteneği ve süreci ile ilgilidir. Yapay Zeka dendiğinde zihinlerde dünyayı ele geçiren çok fonksiyonel, insan benzeri robotlar canlansa da yapay zeka insanların yerine geçmek üzere tasarlanmamıştır. İnsan yeteneklerini ve katkılarını önemli ölçüde geliştirmek üzere tasarlanmıştır. Bu nedenle oldukça değerli bir ticari varlıktır.



## 4. Yapay zeka için programlama dilleri

* Python (%57),
* C/C++ (%44),
* Java (%41),
* R (%37), ve
* JavaScript (%28) dir.
* **Python**
  + ML çalışmalarında kullanılan en yaygın dildir. Başlangıç, hızlı proje üretme amacıyla ve start-up projeleri için önerilir.
* **C/C++**
  + İleri seviye, yapay zeka altyapı/library gibi yüksek performans gereken projelerde önerilir.
* **Java**
  + Eğer ‘*Biz Java biliyorum, hızlı bir şekilde nasıl öğrenebilirim*’ diyorsan sınırlı da olsa kaynak var. –Ancak çok tecrübeli Java ekibi yoksa önerilmez.
* **R**
  + ‘*Ben uygulama geliştirici değilim, istatistik temelim var ve bu tür uygulamalarda çalışıyorum*’ diyorsan önerilir ve çok yaygındır. Yapay zeka uygulama geliştirici girişim ya da bir yazılım odaklı girişim için faydası sınırlıdır.
* **JavaScript**
  + Machine Learning ve Deep Learning odaklı web projelerinde bazı hesaplama yüklerini client’a yıkmak için önerilir. Ayrıca cross-platform mobil uygulamalar için de entegre edilebilen çözümlere sahiptir.
* **C#**
  + Performans ve dil gücü olarak yapay zeka da etkili olabilecekse de açık kaynakta amiral gemisi olmaktan şuan için uzak. Ancak bir C# geliştiricisi olarak herhangi başka bir dil öğrenmeden machine learning, deep learning, computer vision projeleri geliştirilebilecek kadar kaynak ve proje mevcuttur.
* **Go**
  + Güç ve performans olarak çok iyidir. Şahsen yapay zeka uygulamalarında gelecek vadettiğini düşünsem de bu tür projelerde kod yazma maliyeti yüksektir. Sınırlı open-source proje nedeniyle iyi bir Go geliştirici olmak ve ciddi bir proje geliştirme süresine ihtiyaç vardır.
* **Rust**

Bu listedeki en performanslı dillerden biridir. Sistem ve işletim sistemi seviyesinde çalışmalarda kullanılabilecek kadar güçlüdür. Hesaplama gücü nedeniyle henüz ürün ortamında olmasa da AR-GE amacıyla araştırmalar yapılması tavsiye edilir.

* Yapay Zeka (Artificial Intelligence), uzun bir geçmişi olan, ancak hala sürekli ve aktif olarak büyüyen ve değişen bir alandır. Bu derste, modern AI’nın temellerini ve AI’nin bazı temsili programlarını öğreneceksiniz. Yol boyunca, hayal gücümüzün ötesinde insan gücünü genişletmeye devam eden AI alanında çok sayıda programlama ve muazzam olanaklar hakkında sizi heyecanlandırmayı umuyorum.
* Yapay Zeka (AI) teknolojisi günlük yaşamımızda giderek yaygınlaşmaktadır. Oyunculuk, gazetecilik / medya, finans ve aynı zamanda robotik, tıbbi teşhis ve kuantum bilimindeki son teknoloji araştırma alanlarındaki çeşitli endüstrilerde kullanılmaktadır. Bu derste, makine öğrenimi, olasılıksal akıl yürütme, robotik, bilgisayar vizyonu ve doğal dil işleme gibi AI’nın temellerini ve uygulamalarını anlatmamdan önce programlama dilleri hakkında bilgiyi öğreneceksiniz. Başka bir konuda size makine öğrenimi, olasılıksal akıl yürütme, robotik, bilgisayar vizyonu ve doğal dil işleme gibi AI’nın temellerini ve uygulamalarını gösteren bilgiler paylaşacağım.

JS/Node.JS örnek uygulama

**Fotoğrafı parçalara Bölme Uygulaması**

index

 extends layout

  block content

    h1= title

    p Hoşgeldiniz

    script.

      var f1 = function() { document.getElementById('Image').src='#{data.item1}' }

    script.

      var f2 = function() { document.getElementById('Image').src='#{data.item2}' }

    script.

      var f3 = function() { document.getElementById('Image').src='#{data.item3}' }

    button(onclick='f1()') Bir!

    button(onclick='f2()') İki!

    button(onclick='f3()') Üç!

    p

    a: img(id='myImage' height='200' width='400' src='')

router.get

var getData = function () {

    var data = {

        'item1': 'http://public-domain-photos.com/free-stock-photos-1/flowers/cactus-76.jpg',

        'item2': 'http://public-domain-photos.com/free-stock-photos-1/flowers/cactus-77.jpg',

        'item3': 'http://public-domain-photos.com/free-stock-photos-1/flowers/cactus-78.jpg'

    }

    return data;

}

router.get

router.get('/', function (req, res) {

    res.render('index', { title: 'Express', "data" });

});



Project Euler

### **Problem 1**

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Find the sum of all the multiples of 3 or 5 below 1000.

< script >

function sumOfMultiple(limit) {

    var  i, sum = 0;

    for (i = 3; i < limit; i += 1) {

        if (i % 3 === 0 || i % 5 === 0) {

**Sonuç : 233168**

            sum += i

        }

    }

    return sum;

}

console.log(sumOfMultiple(1000));

</script>

### **Problem 11**

In the 20×20 grid below, four numbers along a diagonal line have been marked in red.

08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08  
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00  
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65  
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91  
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80  
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50  
32 98 81 28 64 23 67 10 **26** 38 40 67 59 54 70 66 18 38 64 70  
67 26 20 68 02 62 12 20 95 **63** 94 39 63 08 40 91 66 49 94 21  
24 55 58 05 66 73 99 26 97 17 **78** 78 96 83 14 88 34 89 63 72  
21 36 23 09 75 00 76 44 20 45 35 **14** 00 61 33 97 34 31 33 95  
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92  
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57  
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58  
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40  
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66  
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69  
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36  
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16  
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54  
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48

The product of these numbers is 26 × 63 × 78 × 14 = 1788696.

What is the greatest product of four adjacent numbers in the same direction (up, down, left, right, or diagonally) in the 20×20 grid?

< script >

   var grid = [

        [08, 02, 22, 97, 38, 15, 00, 40, 00, 75, 04, 05, 07, 78, 52, 12, 50, 77, 91, 08],

        [49, 49, 99, 40, 17, 81, 18, 57, 60, 87, 17, 40, 98, 43, 69, 48, 04, 56, 62, 00],

        [81, 49, 31, 73, 55, 79, 14, 29, 93, 71, 40, 67, 53, 88, 30, 03, 49, 13, 36, 65],

        [52, 70, 95, 23, 04, 60, 11, 42, 69, 24, 68, 56, 01, 32, 56, 71, 37, 02, 36, 91],

        [22, 31, 16, 71, 51, 67, 63, 89, 41, 92, 36, 54, 22, 40, 40, 28, 66, 33, 13, 80],

        [24, 47, 32, 60, 99, 03, 45, 02, 44, 75, 33, 53, 78, 36, 84, 20, 35, 17, 12, 50],

        [32, 98, 81, 28, 64, 23, 67, 10, 26, 38, 40, 67, 59, 54, 70, 66, 18, 38, 64, 70],

        [67, 26, 20, 68, 02, 62, 12, 20, 95, 63, 94, 39, 63, 08, 40, 91, 66, 49, 94, 21],

        [24, 55, 58, 05, 66, 73, 99, 26, 97, 17, 78, 78, 96, 83, 14, 88, 34, 89, 63, 72],

        [21, 36, 23, 09, 75, 00, 76, 44, 20, 45, 35, 14, 00, 61, 33, 97, 34, 31, 33, 95],

        [78, 17, 53, 28, 22, 75, 31, 67, 15, 94, 03, 80, 04, 62, 16, 14, 09, 53, 56, 92],

        [16, 39, 05, 42, 96, 35, 31, 47, 55, 58, 88, 24, 00, 17, 54, 24, 36, 29, 85, 57],

        [86, 56, 00, 48, 35, 71, 89, 07, 05, 44, 44, 37, 44, 60, 21, 58, 51, 54, 17, 58],

        [19, 80, 81, 68, 05, 94, 47, 69, 28, 73, 92, 13, 86, 52, 17, 77, 04, 89, 55, 40],

        [04, 52, 08, 83, 97, 35, 99, 16, 07, 97, 57, 32, 16, 26, 26, 79, 33, 27, 98, 66],

        [88, 36, 68, 87, 57, 62, 20, 72, 03, 46, 33, 67, 46, 55, 12, 32, 63, 93, 53, 69],

        [04, 42, 16, 73, 38, 25, 39, 11, 24, 94, 72, 18, 08, 46, 29, 32, 40, 62, 76, 36],

        [20, 69, 36, 41, 72, 30, 23, 88, 34, 62, 99, 69, 82, 67, 59, 85, 74, 04, 36, 16],

        [20, 73, 35, 29, 78, 31, 90, 01, 74, 31, 49, 71, 48, 86, 81, 16, 23, 57, 05, 54],

        [01, 70, 54, 71, 83, 51, 54, 69, 16, 92, 33, 48, 61, 43, 52, 01, 89, 19, 67, 48]

    ];

var maxProduct = 0;

for (var i = 0; i < grid.length; i++) {

    var row = grid[i];

    for (var j = 0; j < row.length; j++) {

        if (j < row.length - 3) {

            var product = row[j] \* row[j + 1] \* row[j + 2] \* row[j + 3];

            if (product > maxProduct) {

                maxProduct = product;

            }

        }

        if (i < grid.length - 3) {

            var product = grid[i][j] \* grid[i + 1][j] \* grid[i + 2][j] \* grid[i + 3][j];

            if (product > maxProduct) {

                maxProduct = product;

            }

        }

        if ((j < row.length - 3) && (i < grid.length - 3)) {

            var product = grid[i][j] \* grid[i + 1][j + 1] \* grid[i + 2][j + 2] \* grid[i + 3][j + 3];

            if (product > maxProduct) {

                maxProduct = product;

            }

        }

        if ((j > 2) && (i < grid.length - 3)) {

            var product = grid[i][j] \* grid[i + 1][j - 1] \* grid[i + 2][j - 2] \* grid[i + 3][j - 3];

            if (product > maxProduct) {

                maxProduct = product;

            }

        }

    }

}

**Sonuç : 70600674**

console.log(maxProduct);

</script>

### **Problem 21**

Let d(*n*) be defined as the sum of proper divisors of *n* (numbers less than *n* which divide evenly into *n*).  
If d(*a*) = *b* and d(*b*) = *a*, where *a* ≠ *b*, then *a* and *b* are an amicable pair and each of *a* and *b* are called amicable numbers.

For example, the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 and 110; therefore d(220) = 284. The proper divisors of 284 are 1, 2, 4, 71 and 142; so d(284) = 220.

Evaluate the sum of all the amicable numbers under 10000.

< script >

var factors = {};

factors.getFactors = function (number){

    var factors = [];

    var highFactors = [];

    var possibleFactor = 1;

    var sqrt = Math.sqrt(number);

    while(possibleFactor <= sqrt){

        if(number % possibleFactor == 0){

            factors[factors.length] = possibleFactor;

            var otherPossibleFactor = number / possibleFactor;

            if(otherPossibleFactor > possibleFactor){

                highFactors[highFactors.length] = otherPossibleFactor;

            }

        }

        possibleFactor++;

    }

    for(var i=highFactors.length-1; i >= 0; i--){

        factors[factors.length] = highFactors[i];

    }

    return factors;

};

var getNaturalFactors = function(number){

    var naturalFactors = factors.getFactors(number);

    naturalFactors.splice(naturalFactors.length - 1, 1);

    return naturalFactors;

};

var getSumOfArray = function(array){

    var sum = 0;

    for(var i = 0; i < array.length; i++){

        sum += array[i];

**Sonuç : 31626**

    }

    return sum;

}

var amicableNumbersSum = 0;

for(var i = 1; i < 10000; i++){

    var naturalFactorsSum = getSumOfArray(getNaturalFactors(i));

    if(naturalFactorsSum != i){

        var testI = getSumOfArray(getNaturalFactors(naturalFactorsSum));

        if(i == testI){

            amicableNumbersSum += i;

            console.log('Amicable number found: ' + i);

        }

    }

}

document.write(amicableNumbersSum);

</script>

**Problem 31**

In the United Kingdom the currency is made up of pound (£) and pence (p). There are eight coins in general circulation:

1p, 2p, 5p, 10p, 20p, 50p, £1 (100p), and £2 (200p).

It is possible to make £2 in the following way:

1×£1 + 1×50p + 2×20p + 1×5p + 1×2p + 3×1p

How many different ways can £2 be made using any number of coins?

<script>

var coinValues = [1, 2, 5, 10, 20, 50, 100, 200];

var comboCounts = [];

for(var i = 0; i < coinValues.length; i++){

    comboCounts[i] = [];

}

var findDifferentCoinCombos = function(amount, maxIndex) {

    var combos = 0;

    maxIndex = (maxIndex || maxIndex == 0) ? maxIndex : coinValues.length - 1;

    if(maxIndex == 0 || amount == 0){

        return 1;

    }

    if(comboCounts[maxIndex][amount]){

        return comboCounts[maxIndex][amount];

    }

    var coinValue = coinValues[maxIndex];

    var coinAmount = 0;

    while(amount >= coinAmount){

        combos += findDifferentCoinCombos(amount - coinAmount, maxIndex - 1);

        coinAmount += coinValue;

    }

    comboCounts[maxIndex][amount] = combos;

    return combos;

**Sonuç : 73682**

};

console.log(findDifferentCoinCombos(200));

    </script>

**Problem 71**

Consider the fraction, *n/d*, where *n* and *d* are positive integers. If *n*<*d* and HCF(*n,d*)=1, it is called a reduced proper fraction.

If we list the set of reduced proper fractions for *d* ≤ 8 in ascending order of size, we get:

1/8, 1/7, 1/6, 1/5, 1/4, 2/7, 1/3, 3/8, **2/5**, 3/7, 1/2, 4/7, 3/5, 5/8, 2/3, 5/7, 3/4, 4/5, 5/6, 6/7, 7/8

It can be seen that 2/5 is the fraction immediately to the left of 3/7.

By listing the set of reduced proper fractions for *d* ≤ 1,000,000 in ascending order of size, find the numerator of the fraction immediately to the left of 3/7.

       var program = require('commander');

program

    .version('0.1.0')

    .description('Ordered fractions')

    .option('-l, --limit <int>', 'The maximum denominator', Number, 1000000)

    .parse(process.argv);

var RIGHT\_FRAC = [3, 7];

var left\_frac = [2, 7];

for (var d = 2; d <= program.limit; d++) {

    var start\_n = Math.floor(left\_frac[0] / left\_frac[1] \* d);

    var end\_n = Math.floor(RIGHT\_FRAC[0] / RIGHT\_FRAC[1] \* d) + 1;

    for (var n = start\_n; n <= end\_n; n++) {

        if (compare\_fractions(n, d, RIGHT\_FRAC[0], RIGHT\_FRAC[1]) >= 0) {

            continue;

        }

        if (compare\_fractions(n, d, left\_frac[0], left\_frac[1]) > 0) {

            left\_frac = [n, d];

        }

    }

**428570/999997 3/7**

}

console.log(left\_frac[0]);

function compare\_fractions(numer1, denom1, numer2, denom2) {

    return numer1 \* denom2 - numer2 \* denom1;

}

LeetCode

1. **Two Sum**

Given an array of integers, return **indices** of the two numbers such that they add up to a specific target.

You may assume that each input would have ***exactly*** one solution, and you may not use the *same* element twice.

**Example:**

Given nums = [2, 7, 11, 15], target = 9,

Because nums[**0**] + nums[**1**] = 2 + 7 = 9,

return [**0**, **1**].

public int[] twoSum(int[] nums, int target) {

for (int i = 0; i < nums.length; i++) {

for (int j = i + 1; j < nums.length; j++) {

if (nums[j] == target - nums[i]) {

return new int[] { i, j };

}

}

}

throw new IllegalArgumentException("No two sum solution");

}

< script >

dizi =[2, 7, 11, 15]

hedef=9

var toplam = function(dizi, hedef) {

    const length = dizi.length;

    const cacheNumbToIndex = {};

**Sonuç : [0,1]**

    let index;

    for (index = 0; index < length; index++) {

        const gotNum = dizi[index];

        const wantedNum = hedef - gotNum;

        if (cacheNumbToIndex[wantedNum] !== undefined) {

            return [index, cacheNumbToIndex[wantedNum]].sort();

        }

        cacheNumbToIndex[gotNum] = index;

    }

};

</script>

**121. Best Time to Buy and Sell Stock**

Say you have an array for which the *i*th element is the price of a given stock on day *i*.

If you were only permitted to complete at most one transaction (i.e., buy one and sell one share of the stock), design an algorithm to find the maximum profit.

Note that you cannot sell a stock before you buy one.

**Example 1:**

**Input:** [7,1,5,3,6,4]

**Output:** 5

**Explanation:** Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

  Not 7-1 = 6, as selling price needs to be larger than buying price.

**Example 2:**

**Input:** [7,6,4,3,1]

**Output:** 0

**Explanation:** In this case, no transaction is done, i.e. max profit = 0.

public class Solution {

public int maxProfit(int prices[]) {

int maxprofit = 0;

for (int i = 0; i < prices.length - 1; i++) {

for (int j = i + 1; j < prices.length; j++) {

int profit = prices[j] - prices[i];

if (profit > maxprofit)

maxprofit = profit;

}

}

return maxprofit;

}

}

< script >

sayilar: [3,8,2,5,4,7]

**Sonuç : 5**

var maxProfit = function(prices) {

    if(prices.length < 2) {

        return 0;

    }

    let maxProfit = 0;

    let maxStock = Math.max(prices[prices.length - 1], prices[prices.length - 2]);

    for(let i = prices.length - 2; i > -1; i--) {

        let profit = maxStock - prices[i];

        maxStock = Math.max(maxStock, prices[i]);

        maxProfit = Math.max(profit, maxProfit);

    }

    return maxProfit;

};

</script>

**371. Sum of Two Integers**

Calculate the sum of two integers *a* and *b*, but you are **not allowed** to use the operator + and -.

**Example 1:**

**Input:** a = 1, b = 2

**Output:** 3

**Example 2:**

**Input:** a = -2, b = 3

**Output:** 1

< script >

var s1=5

var s2=5

**Sonuç : 10**

var toplam=Number(s1)+Number(s2);

window.alert(toplam);

</script>

**Soru 41**

Given an unsorted integer array, find the smallest missing positive integer.

**Example 1:**

Input: [1,2,0]

Output: 3

**Example 2:**

Input: [3,4,-1,1]

Output: 2

**Example 3:**

Input: [7,8,9,11,12]

Output: 1

   <script>

    var firstMissingPositive = function(nums) {

    const n = nums.length

    for (let i = 1; i < n; i++) {

        while (nums[i] <= n && nums[i] !== nums[nums[i] - 1]) {

        const t = nums[i]

        nums[i] = nums[t - 1]

        nums[t - 1] = t

        }

    }

    for (let i = 0; i < n; i++) {

**Sonuç : 3642**

        if (nums[i] !== i + 1) {

        return i + 1

        }

    }

    return n + 1

    };

    <script>

**Soru 11**

Given n non-negative integers a1, a2, ..., an, where each represents a point at coordinate (i, ai). n vertical lines are drawn such that the two endpoints of line i is at (i, ai) and (i, 0). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Note: You may not slant the container and n is at least 2.

The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

**Example:**

**Input:** [1,8,6,2,5,4,8,3,7]

**Output:** 49

    <script>

    var maxArea = function (height) {

  let max = 0

  for (let l = 0, r = height.length - 1; l < r; l++, r--) {

    max = Math.max(max, (r - l) \* Math.min(height[l], height[r]))

    if (height[l] < height[r]) {

      r++

    } else {

      l--

**Sonuç : 49**

    }

  }

  return max

};

    <script>