İLERİ PROGRAMLAMA

**Öğrenci Numarası**: 20181129013

**İsim Soyisim**: Gaye Şahin

**Js İle Yapay Zeka Uygulamaları Geliştirilebilir mi?**

**TensorFlow Nedir?**

Açık kaynak kodlu bir deep learning(derin öğrenme) kütüphanesidir. Esnek yapısı sayesinde, tek bir API ile platform farketmeksizin hesaplamaları, bir veya birden fazla CPU, GPU kullanarak deploy etmenize olanak sağlar. Temelinde Python kullanılarak geliştirilen bu framework, günümüzde Python‘ın yanısıra C++, Java, C#, Javascript ve R gibi birçok dili desteklemektedir.  
 TensorFlow çeşitli alanlarda kullanıma sahip bir teknolojidir. Yaygın olarak kullanılmasındaki en önemli etkenlerden birisi farklı platformlara göre hazırlanmış TensorFlow kütüphanelerinin olmasıdır. Yani mobil uygulama, web uygulaması veya IoT cihazları üzerinde geliştireceğiniz projeleriniz için TensorFlow platforma uygun kütüphane sağlıyor. Python, R, C++, Java gibi dillerdeki TensorFlow uygulamaları geliştirmek için dile ait olan kütüphaneyi kullanabilirsiniz. Ayrıca web ve mobil uygulamalara özel geliştirilen TensowFlow paketleri de bulunmaktadır.

**TensorFlow.js**

TensorFlow’un istemci taraflı çalışan JavaScript kütüphanesidir. İstemci taraflı web uygulaması geliştirmek isterseniz TensorFlow.js kullanmalısınız. Bunun için Node.js paketini yükleyebilir veya aşağıdaki script tag’lerini sayfanıza ekleyerek TensorFlow’u kullanabilirsiniz:

<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0/dist/tf.min.js"></script>

**TensorFlow.js** sayesinde internet tarayınız üzerinden yapay zeka ile ilgili bir çok işlemi yapabilirsiniz. TensorFlow.js sayesinde makine öğrenmesi modellerini tarayıcınız üzerinde geliştirebilir veya eğitebilirsiniz.

PYTHON ve R DİLİNE GÖRE JS/NODE.JS KARŞILAŞTIRMASI

# **Node vs Python'ın Karşılaştırması**

**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.

**JS/NODE.JS ÖRNEK UYGULAMA**

Bu yazımdaki projemizin konusu; internet sitemizin tarayıcı üzerinden kamera iznini kullanarak kamerada görünen nesnelerin ne olabileceğini tahmin edip oranlarıyla birlikte ekrana yazdıran bir uygulama olacak.

İlgili kütüphanelerin anlaşılması için temel özelliklerini kullanarak bu uygulamayı geliştireceğiz. Nesne tanımlamaların olasılıkları İngilizce olarak çıktı verecektir. Projemizin yalnızca iki dosyası olacaktır; birincisi index.html ikincisi ise init.js dosyalarıdır.

|  |
| --- |
| <html> |
|  | <head> |
|  | <title>Bu nedir?</title> |
|  | <meta charset="UTF-8" /> |
|  | <script src="https://unpkg.com/@tensorflow/tfjs@0.10.3"></script> |
|  | <script src="https://unpkg.com/@tensorflow-models/mobilenet@0.0.1"></script> |
|  | <script src="init.js"></script> |
|  | </head> |
|  |  |
|  | <body onload="setup()"> |
|  | <div style="text-align:center"> |
|  | <h1> |
|  | Bu nedir? |
|  | </h1> |
|  | <video id="video">Video bulunamadı.</video> |
|  | <pre id="tahminler"></pre> |
|  | <canvas id="canvas" style="display:none"></canvas> |
|  | </div> |
|  | </body> |
|  | </html> |

Kodlardan anlaşılacağı üzere <video> tag’i içerisinde kamera görüntümüz belirecektir. <canvas> tag’inde ise kameramızdan bir saniyede bir çektiğimiz görselleri arka planda burada tutacağız. <pre> tag’i içerisinde ise canvas da tutulan görselin anlık olarak tahminini yayınlayacağız.

Şimdi Javascript dosyamızı oluşturalım. Öncelikle ana fonksiyonumuz olan setup() fonksiyonunu oluşturup içerisinde de bazı sabitlerimizi tanımlayalım:

function setup()

{

let video = document.getElementById("video");

let canvas = document.getElementById("canvas");

let pre = document.getElementById("tahminler");

let model = null;

}

Kamera izinlerimizi isteyip izin verilmesi halinde <video> tag’i içerisinde kamera görüntülerimizi gösterelim:

async function startCamera() {

let stream = await navigator.mediaDevices.getUserMedia({ video: true }); //kamera izni isteyelim

video.srcObject = stream; //<video> tag'i içerisine kaynağımızı kamera olarak tanımladık

await video.play(); //asenkron olarak kameramızın görüntüsünü bu tag içerisinde görüntüleyelim

setInterval(() => takeSnapshot(), 1000); //her 1 saniyede bir görüntü çekip canvas'a göndereceğiz}

Fonksiyon içerisinden de anlaşılacağı üzere takeSnapshot() isimli fonksiyon yardımıyla kameramızdan saniyede bir kare çekilip canvas içerisine aktarılacak:

function takeSnapshot() {

let context = canvas.getContext("2d"),

width = video.videoWidth,

height = video.videoHeight; //canvas'ı videomuzdan çekilen genişlik ve yüksekliğe uygun olarak oluşturuyoruz.

if (width && height) {

canvas.width = width;

canvas.height = height;

context.drawImage(video, 0, 0, width, height); //canvas içerisine kameradan çekilen görüntüyü işliyoruz.

classifyImage(); //MobileNet yardımıyla resim sınıflandırması yapılacak}}

Şimdi de sıra geldi **MobileNet** kütüphanesi kullanarak canvas içerisindeki resimlerin sınıflandırmasına:

async function classifyImage() {

predictions = await model.classify(canvas); //sınıflandırmalar için kullanılan method

displayPredictions(predictions); //elde edilen tahminleri gösterecek fonksiyon}

Sınıflandırma fonksiyonumuzda görselden elde ettiğimiz tahminlerin dizisini göstermek üzere **displayPredictions()** fonksiyonuna aktarıyoruz. Ardından bu fonksiyon içerisinde de ilgili tahminlerde gezinerek olası en yüksek üç tahmini sitemizde göstereceğiz:

function displayPredictions(predictions) {

let val = ""; //boş tahmin tanımalama

for (prediction of predictions) {

let perc = (prediction.probability \* 100).toFixed(2); // tanımlanan nesne olasılığını hesaplama

val += `${perc}% | ${prediction.className}\n`; //olasılığı hesaplanan nesneyi ve olasılık değerini ekleme

}

pre.innerHTML = val; //elde edilen değerleri tahminler içerisine yazdırma

}

Tüm bunların ardından MobileNet kütüphanemizi model değişkenine atayan ve kameramızı çalıştıran fonksiyonu tetikleyen ana fonksiyonumuz olan main() fonksiyonunu tanımlayalım:

async function main() {

model = await mobilenet.load();

await startCamera();

}

Son olarak init.js dosyamız aşağıdaki gibi olacaktır:

|  |
| --- |
| function setup() { |
|  | let video = document.getElementById("video"); |
|  | let canvas = document.getElementById("canvas"); |
|  | let pre = document.getElementById("tahminler"); |
|  | let model = null; |
|  |  |
|  | async function startCamera() { |
|  | let stream = await navigator.mediaDevices.getUserMedia({ video: true }); //kamera izni isteyelim |
|  | video.srcObject = stream; //<video> tag'i içerisine kaynağımızı kamera olarak tanımladık |
|  | await video.play(); //asenkron olarak kameramızın görüntüsünü bu tag içerisinde görüntüleyelim |
|  |  |
|  | setInterval(() => takeSnapshot(), 1000); //her 1 saniyede bir görüntü çekip canvas'a göndereceğiz |
|  | } |
|  |  |
|  | function takeSnapshot() { |
|  | let context = canvas.getContext("2d"), |
|  | width = video.videoWidth, |
|  | height = video.videoHeight; //canvas'ı videomuzdan çekilen genişlik ve yüksekliğe uygun olarak oluşturuyoruz. |
|  |  |
|  | if (width && height) { |
|  | canvas.width = width; |
|  | canvas.height = height; |
|  |  |
|  | context.drawImage(video, 0, 0, width, height); //canvas içerisine kameradan çekilen görüntüyü işliyoruz. |
|  |  |
|  | classifyImage(); //MobileNet yardımıyla resim sınıflandırması yapılacak |
|  | } |
|  | } |
|  |  |
|  | async function classifyImage() { |
|  | predictions = await model. (canvas); //sınıflandırmalar için kullanılan method |
|  | displayPredictions(predictions); //elde edilen tahminleri gösterecek fonksiyon |
|  | } |
|  |  |
|  | function displayPredictions(predictions) { |
|  | let val = ""; //boş tahmin tanımalaöa |
|  |  |
|  | for (prediction of predictions) { |
|  | let perc = (prediction.probability \* 100).toFixed(2); // tanımlanan nesne olasılığını hesaplama |
|  | val += `${perc}% | ${prediction.className}\n`; //olasılığı hesaplanan nesneyi ve olasılık değerini ekleme |
|  | } |
|  | pre.innerHTML = val; //elde edilen değerleri tahminler içerisine yazdırma |
|  | } |
|  |  |
|  | async function main() { |
|  | model = await mobilenet.load(); |
|  | await startCamera(); |
|  | } |
|  | main(); |
|  | } |

**BÖLÜM 2: Project Euler**

[**Problem 13 - Large Sum**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html)

BigInteger sınıfı ile bu problemi çözmek çok kolay. Satırları tek tek bir BigInt dizisine aktarıp topladıktan sonra ilk 10 basamağı yazdırabiliriz. Fakat buna gerek yok. Biraz mantıkla, çok büyük sayılar kullanmamıza gerek kalmadan bu işlemi yapabiliriz.

Her satırdaki ilk 10 harfi alıp, 10 basamaklı bir sayıya çevirip bunları topladığımızda çıkan sonucun ilk 10 basamağı muhtemelen aradığımız sonuçtur. Muhtemelen diyorum, çünkü diğer basamaklardan gelen elde olabilir. Bu yüzden ilk 13 basamağı toplamayı tercih ettim.

long Toplam = 0;  
string gecici = "";  
  
for (int i = 0; i < 100; i++)  
{  
   gecici = "";  
   for (int j = 0; j < 13; j++)  
   {  
      gecici += list[i][j].ToString();  
   }  
   Toplam += Convert.ToInt64(gecici);  
}

Cevap: 5537376230   
 Eğer ilk 10 basamakla denerseniz son iki hanenin 30 yerine 29 olduğunu, eldenin eklenememiş olduğunu göreceksiniz.

[**Problem 3 -**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html) [**Largest Prime Factor**](https://projecteuler.net/problem=3)

**function** largestPrimeFactor(n) {

**var** d = **Math**.ceil(**Math**.sqrt(n));

**function** isPrime(n) {

**var** i, limit = **Math**.ceil(**Math**.sqrt(n));

**for** (i = 3; i <= limit; i += 2) {

**if** (n % i === 0) {

**return** **false**;

}

}

**return** **true**;

}

d = (d & 1) === 0 ? d - 1 : d;

**while** (!(n % d === 0 && isPrime(d))) {

d -= 2; *// odd numbers only*

}

**return** d;

console.log(largestPrimeFactor(600851475143));

Cevap : 6857

[**Problem 23 - Non-abundant sums**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html)

var divisors = function(number){

sqrtNumber = Math.sqrt(number);

var sum = 1;

for(var i = 2; i<= sqrtNumber; i++)

{

if (number == sqrtNumber \* sqrtNumber)

{

sum += sqrtNumber;

sqrtNumber--;

}

if( number % i == 0 )

{

sum += i + (number/i);

}

}

if (sum > number) {return true;}

else {return false;}

};

var abundent = [], k = 0;

var upperLimit = 28123;

for (var i = 1; i <= upperLimit; i++)

{

if (divisors(i))

{abundent[k] = i; k++};

}

var abundentCount = abundent.length;

var canBeWrittenAsAbundant = [];

for (var i = 0; i < abundentCount; i++){

for (var j = i; j < abundentCount; j++){

if (abundent[i] + abundent[j] <= upperLimit){canBeWrittenAsAbundant[abundent[i]+abundent[j]] = true;}

else {

break;

}

}

}

for (i=1; i <= upperLimit; i++){

if (canBeWrittenAsAbundant[i] == true){continue;}

else {canBeWrittenAsAbundant[i] = false;}

}

var sum = 0;

for (i=1; i <= upperLimit; i++)

{

if (!canBeWrittenAsAbundant[i]){

sum += i;

}

}

console.log(sum);

Cevap : 4179871

[**Problem 33 - Digit Cancelling Fractions**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html)

var gcd = function(a, b) {

var temp;

while (b !== 0) {

temp = b;

b = a % temp;

a = temp;

}

return a;

};

var cancelFractionDigits = function(fraction) {

var nDigits = String(fraction.n).split("");

var dDigits = String(fraction.d).split("");

var nI = 0;

var cancelled = false;

while (nI < nDigits.length) {

var dI = dDigits.indexOf(nDigits[nI]);

if (dI >= 0) {

nDigits.splice(nI, 1);

dDigits.splice(dI, 1);

cancelled = true;

} else {

nI++;

}

}

if (nDigits.length > 0 && dDigits.length > 0) {

var dNum = Number(dDigits.join(""));

if (dNum) {

return {n: Number(nDigits.join("")),

d: dNum,

cancelled: cancelled};

}

}

return null;

};

var simplify = function(fraction) {

if (fraction.d == 0) return null;

var gcdValue = gcd(fraction.n, fraction.d);

return {n: fraction.n / gcdValue,

d: fraction.d / gcdValue,

simplified: true};

};

var arrayProduct = function(prev, current) {

return {n: prev.n \* current.n,

d: prev.d \* current.d};

};

var p33 = function() {

var foundFractions = [];

for (var n = 10; n <= 99; n++) {

for (var d = n+1; d <= 99; d++) {

if (d % 10 == 0) continue;

var ratio = n / d;

var fraction = {n: n, d: d};

cancelled = cancelFractionDigits(fraction);

if (cancelled != null && cancelled.cancelled) {

if (ratio == (cancelled.n / cancelled.d)) {

foundFractions.push(fraction);

}

}

}

}

return simplify(foundFractions.reduce(arrayProduct)).d;

};

console.log(p33());

[**Problem 53 - Combinatoric Selections**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html)

Kombinasyon hesaplama işi bilgisayarı uzun süre meşgul edebiliyor, çünkü kombinasyon belli sayıların faktöriyellerine bakılarak bulunuyor:

C(n,r) = n! / r!(n-r)!

O nedenle gerektiği yerde bu faktöriyel sonuçlarını cachelemeye karar verdim. Sonuç olarak, aşağıdaki kodla çift çekirdekli 2.4 GHz makinede 0.15 saniyelik bir runtime ile cevaba ulaştım. Project Euler’de diğer cevaplara baktığımda daha verimli algoritmalara da ulaştım, bazıları Pascal Triangle kullanmış – ki bence mükemmel bir fikir. Ama benim algoritmam da hızlı bir şekilde sonuca ulaştırdı.

import sys

import math

import time

fact\_results = []

def fact(i):

if i == 1:

return i

elif i < 1:

return 1

else: return i \* fact(i-1)

def comb( i, j):

global fact\_results

if fact\_results[i] == 0:

fact\_results[i] = fact(i)

if fact\_results[j] == 0:

fact\_results[j] = fact(j)

if fact\_results[i-j] == 0:

fact\_results[i-j] = fact(i-j) return fact\_results[i] / (fact\_results[j] \* fact\_results[i-j])

def main():

global fact\_results

count = 0

for i in range(101):

fact\_results.append(0)

for i in range(23,101):

for j in range(1,i+1):

if comb( i, j) > 1000000:

count += 1

print 'found one on %d.C.%d' % (i,j)

print 'finished. total: %d' % (count)

if \_\_name\_\_ == '\_\_main\_\_':

start\_time = time.time()

main()

print time.time() - start\_time, "seconds"

**BÖLÜM 3: Leetcode**

[**Problem 13 - Remove Duplicates from Sorted List**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html)

Bu, yalnızca liste düğümü işaretleyicilerini değiştirme yeteneğinizi sınayan basit bir sorundur. Giriş listesi sıralandığından, değerini listedeki sonraki düğümle karşılaştırarak bir düğümün kopya olup olmadığını belirleyebiliriz. Yineleniyorsa, next geçerli düğümün işaretçisini, bir sonraki düğümü atlayacak ve bir sonraki düğümü izleyen doğrudan işaret edecek şekilde değiştiririz.

public ListNode deleteDuplicates(ListNode head) {

ListNode current = head;

while (current != null && current.next != null) {

if (current.next.val == current.val) {

current.next = current.next.next;

} else {

current = current.next;

}

}

return head;

}

[**Problem 3 - Longest Substring Without Repeating Characters**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html)

Alt dizideki boolean allUnique(String substring)karakterlerin tümü benzersizse true döndürecek bir işleve sahip olduğumuzu varsayalım . Verilen dizenin tüm olası alt dizeleri üzerinden yineleme yapabilir sve işlevi çağırabiliriz allUnique. Doğru olduğu ortaya çıkarsa, yinelenen karakterler olmadan maksimum alt dize uzunluğu cevabımızı güncelliyoruz.

public class Solution {

public int lengthOfLongestSubstring(String s) {

int n = s.length();

int ans = 0;

for (int i = 0; i < n; i++)

for (int j = i + 1; j <= n; j++)

if (allUnique(s, i, j)) ans = Math.max(ans, j - i);

return ans;

}

public boolean allUnique(String s, int start, int end) {

Set<Character> set = new HashSet<>();

for (int i = start; i < end; i++) {

Character ch = s.charAt(i);

if (set.contains(ch)) return false;

set.add(ch);

}

return true;

}

}

[**Problem 73 - Set Matrix Zeroes**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html)

Çözüm için sırasıyla şu yolu izliyoruz.

1. Orijinal dizimiz üzerinden bir geçiş yaparız ve sıfır giriş ararız.
2. Adresindeki bir girişin [i, j]0 olduğunu tespit edersek, satır ive sütunu bir yere kaydetmemiz gerekir j.
3. Yani, setssatırlar için iki , sütunlar için bir tane kullanıyoruz.
4. Son olarak, orijinal matrisi tekrarlıyoruz. Her hücre için satır rveya sütunun cönceden işaretlenip işaretlenmediğini kontrol ederiz . Bunlardan herhangi biri işaretliyse, hücredeki değeri 0 olarak ayarladık.

class Solution {

public void setZeroes(int[][] matrix) {

int R = matrix.length;

int C = matrix[0].length;

Set<Integer> rows = new HashSet<Integer>();

Set<Integer> cols = new HashSet<Integer>();

for (int i = 0; i < R; i++) {

for (int j = 0; j < C; j++) {

if (matrix[i][j] == 0) {

rows.add(i);

cols.add(j);

}

}

}

for (int i = 0; i < R; i++) {

for (int j = 0; j < C; j++) {

if (rows.contains(i) || cols.contains(j)) {

matrix[i][j] = 0;

}

}

}

}

}

[**Problem 63 - Unique Paths II**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html)

1. Birinci hücre yani obstacleGrid[0,0]içeriyorsa 1, bu birinci hücrede bir engel olduğu anlamına gelir. Bu nedenle robot herhangi bir hareket yapamayacak ve yol sayısını döndüreceğiz 0.
2. Aksi takdirde, obstacleGrid[0,0]sahip 0başlangıçta biz ayarlayın 1ve ilerlemeye.
3. İlk satırı yineleyin. Bir hücre başlangıçta a içeriyorsa 1, mevcut hücrenin bir engeli olduğu ve herhangi bir yola katkıda bulunmaması gerektiği anlamına gelir. Bu nedenle, bu hücrenin değerini olarak ayarlayın 0. Aksi takdirde, önceki hücrenin değerine ayarlayın;obstacleGrid[i,j] = obstacleGrid[i,j-1]
4. İlk sütunu yineleyin. Bir hücre başlangıçta a içeriyorsa 1, mevcut hücrenin bir engeli olduğu ve herhangi bir yola katkıda bulunmaması gerektiği anlamına gelir. Bu nedenle, bu hücrenin değerini olarak ayarlayın 0. Aksi takdirde, önceki hücrenin değerine ayarlayın;obstacleGrid[i,j] = obstacleGrid[i-1,j]
5. Şimdi, hücreden başlayarak dizi boyunca yineleme yapın obstacleGrid[1,1]. Bir hücre başlangıçta herhangi bir engel içermiyorsa, o hücreye ulaşma yollarının sayısı, üzerindeki hücreye ulaşma yollarının ve soldaki hücreye ulaşma yollarının toplamı olacaktır.
6. Bir hücre bir engel içeriyorsa, onu ayarlayın 0ve devam edin. Bu, başka bir yola katkıda bulunmadığından emin olmak için yapılır.

class Solution {

public int uniquePathsWithObstacles(int[][] obstacleGrid) {

int R = obstacleGrid.length;

int C = obstacleGrid[0].length;

if (obstacleGrid[0][0] == 1) {

return 0;

}

obstacleGrid[0][0] = 1;

for (int i = 1; i < R; i++) {

obstacleGrid[i][0] = (obstacleGrid[i][0] == 0 && obstacleGrid[i - 1][0] == 1) ? 1 : 0;

}

for (int i = 1; i < C; i++) {

obstacleGrid[0][i] = (obstacleGrid[0][i] == 0 && obstacleGrid[0][i - 1] == 1) ? 1 : 0;

}

for (int i = 1; i < R; i++) {

for (int j = 1; j < C; j++) {

if (obstacleGrid[i][j] == 0) {

obstacleGrid[i][j] = obstacleGrid[i - 1][j] + obstacleGrid[i][j - 1];

} else {

obstacleGrid[i][j] = 0;

}

}

}

return obstacleGrid[R - 1][C - 1];

}

}

[**Problem 83 -**](http://projecteulercozum.blogspot.com/2014/02/project-euler-13-large-sum.html) **Remove Duplicates from Sorted List**

Bu, yalnızca liste düğümü işaretleyicilerini değiştirme yeteneğinizi sınayan basit bir sorundur. Giriş listesi sıralandığından, değerini listedeki sonraki düğümle karşılaştırarak bir düğümün yinelenip yinelenmediğini belirleyebiliriz. Yineleniyorsa, next geçerli düğümün işaretçisini, bir sonraki düğümü atlayacak ve bir sonraki düğümden sonra doğrudan işaret edecek şekilde değiştiririz.

**public ListNode deleteDuplicates(ListNode head) {**

**ListNode current = head;**

**while (current != null && current.next != null) {**

**if (current.next.val == current.val) {**

**current.next = current.next.next;**

**} else {**

**current = current.next;**

**}**

**}**

**return head;**

**}**

**KAYNAKÇA**

<https://tr.wikipedia.org/wiki/Algoritma_analizi>

<http://www.bilgisayarkavramlari.com/2010/06/17/karmasiklik-siniflari-complexity-classes/>

<http://bilgisayarkavramlari.sadievrenseker.com/2010/09/24/algoritma-analizi-analysis-of-algorithms/>

<http://www.kodyazan.com/MakaleDetay/1225/Algoritma-Nedir--Analizi-Nasil-Yapilir->

[*"The Definitive Glossary of Higher Mathematical Jargon — Algorithm"*](https://mathvault.ca/math-glossary/#algo). Math Vault (İngilizce). 1 Ağustos 2019. 7 Şubat 2020 tarihinde kaynağından [*Arşivlendi*](https://web.archive.org/web/20200207194438/https:/mathvault.ca/math-glossary/#algo)*. Erişim tarihi: 14 Kasım 2019*.

[*"Definition of ALGORITHM"*](https://www.merriam-webster.com/dictionary/algorithm). Merriam-Webster Online Dictionary (İngilizce). 14 Şubat 2020 tarihinde kaynağından [*Arşivlendi*](https://web.archive.org/web/20200214074446/https:/www.merriam-webster.com/dictionary/algorithm)*. Erişim tarihi: 14 Kasım 2019*.

[*"Al-Khwarizmi - Islamic Mathematics"*](http://www.storyofmathematics.com/islamic_alkhwarizmi.html). The Story of Mathematics. 25 Temmuz 2019 tarihinde kaynağından [*Arşivlendi*](https://web.archive.org/web/20190725202237/http:/www.storyofmathematics.com/islamic_alkhwarizmi.html)*. Erişim tarihi: 14 Kasım 2019*.

[*"Al-Khwarizmi biography"*](http://www-history.mcs.st-andrews.ac.uk/Biographies/Al-Khwarizmi.html). www-history.mcs.st-andrews.ac.uk. 2 Ağustos 2019 tarihinde kaynağından [*Arşivlendi*](https://web.archive.org/web/20190802091553/http:/www-history.mcs.st-andrews.ac.uk/Biographies/Al-Khwarizmi.html)*. Erişim tarihi: 27 Şubat 2020*.

[*"Etymology of algorithm"*](http://chambers.co.uk/search/?query=algorithm&title=21st). Chambers Dictionary. 31 Mart 2019 tarihinde kaynağından [*Arşivlendi*](https://web.archive.org/web/20190331204600/https:/chambers.co.uk/search/?query=algorithm&title=21st)*. Erişim tarihi: 13 Aralık 2016*.