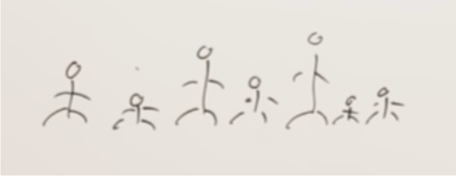
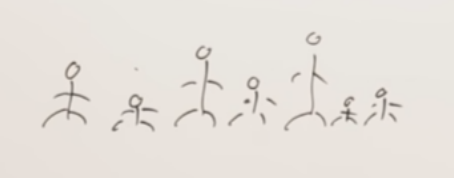


What is the idea behind that ? Idea is suppose there are group of students in class. And a teacher ask them to arrange themselves in increasing order of their height. There are two options. Teacher can show their place u go there u go there.so first option is Teacher can show students places. Second option teacher can ask to students to arrange themselves so every student find his place in the sorted order. So quick method is STUDENTS FINDS THEIR PLACES. THIS IS THE IDEA OF QUICK SORT.



En kisa boylu boyunu biliyor ve en basa geciyor.

Shortes person know his place and come and takes his place on first.

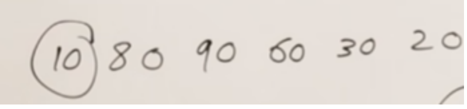
He can know that he is beginning because he is the shortest person in this group.

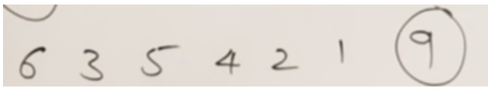
This is same for tallest person. He knows his place.

Ve diger geri kalanlarda kendi aralarinda arrangement yapiyorlar. Yani sen kisasin ben uzunum gibi.

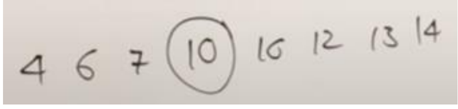
So others find their places by arranging each other.

Bu anlatttiklarimiz Quick sortun arkasinda ki temel fikir.





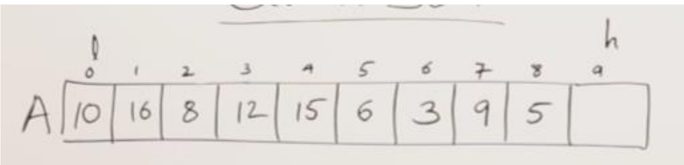
Which elements seems to be sorted? Ilk resimde 10 yani en kucuk.Ikincide 9 cunku en buyuk.



10 burada sorted cunku oncekiler kendinden kucuk sonrakiler ise buyuk.

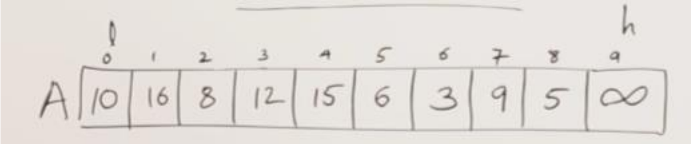
Quick sort based idea that; an element in sorted position If all the element left side smaller than that element all of element right side is greater than that element (Yukari'da olan resime bak anlarsin) and that element in sorted position.rest of the elements may or may not be sorted.

Let's look at procedure.Quick sort is divide and conquer algorithm it follows divide and conquer strategy. Means that it will split problem to subproblem. And solve those subproblems.



0	1	2	3	4	5	6	7	8	9
10	16	8	12	15	6	3	9	5	

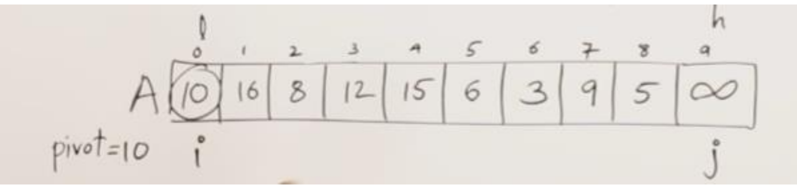
I add infinity to 9'th place



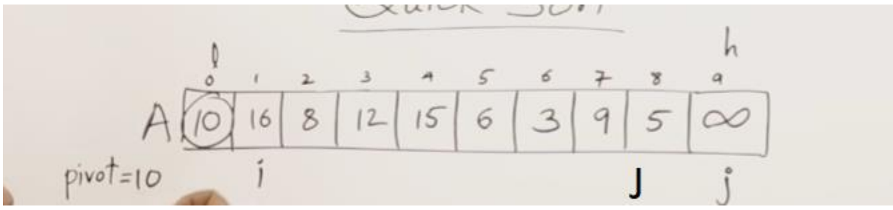
We have taken infinity to show end of a list. Otherwisse for ex there are 10 elements then we can work with length.

Our pivot element is 10.Temel mantik 10’dan kucuk olanlari yanina buyuk olanlari ileriye gondermek.

I will search the elements which are greater than 10 J is searching an element which are smaller than 10 so that they can exchange the numbers. 10 sayisi burada pivot sayi.



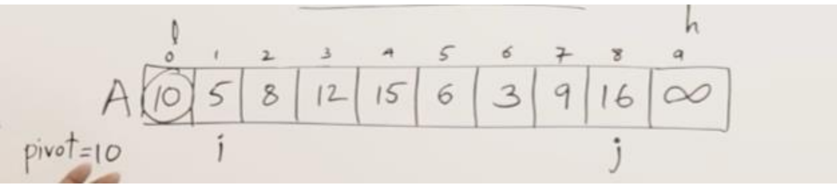
The procedure we are doing is partitioning procedure:



1- ADIM I sayisini 10’dan buyuk bir sayi veya esit buluncaya kadar arttir. Bunu dersek hemen yaninda olan 16’yi bulur. J harfi icin ise 10’dan kucuk veya esit bir sayi buluncaya kadar azalt hemen yanina 5 sayisina gidiyor.

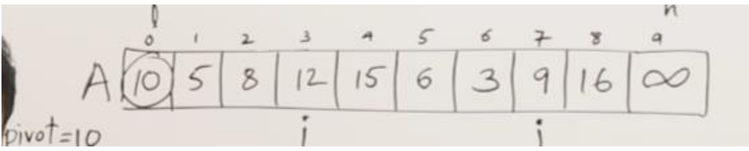
Simdi I ve J indexleri yer degistirecek.5 16’nin yerine 16 ise 5’in index yerine gece

cek.

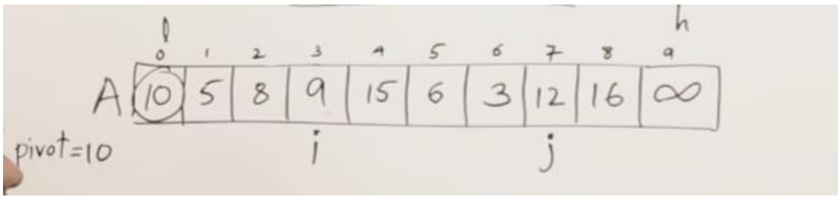


2-ADIMDevam edelim I harfi icin 8 e geldigimizde 10’dan buyuk degil. Fakat 12 10’dan buyuk ve duruyor.

J harfi icin 10’dan kucuk olan bir sayi buluncaya kadar azalarak gel. Zaten yanina 9 oldugu icin direk orada duracak. z

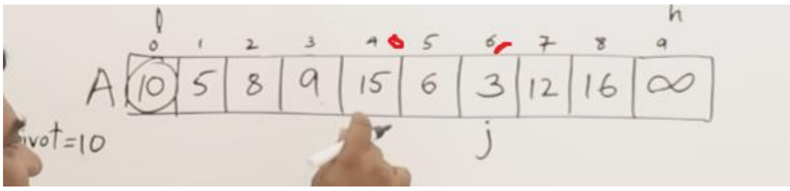


Simdi 12 sayisiyla 9 sayisini yer degistiriyoruz.

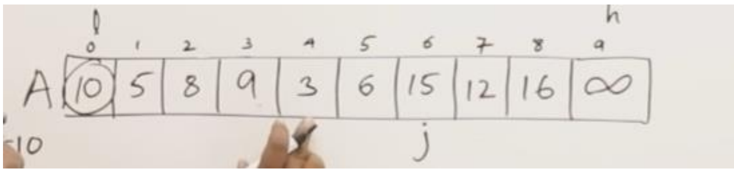


3 ADIM- I sayisi icin= 10'dan buyuk olan sayiya kadar ilerle hemen yaninda 15 var orada durur

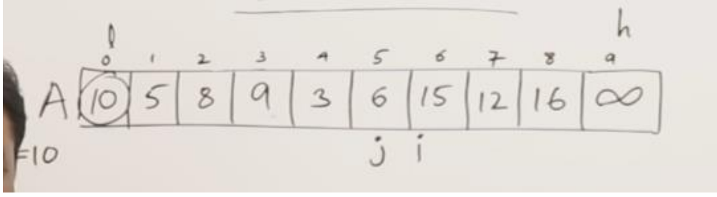
J sayisi icin=10'dan kucuk sayi bulana kadar ilerle. Hemen yaninda 3 var ona gider ve durur



Simdi 3 ve 15'in yerlerini degistirelim

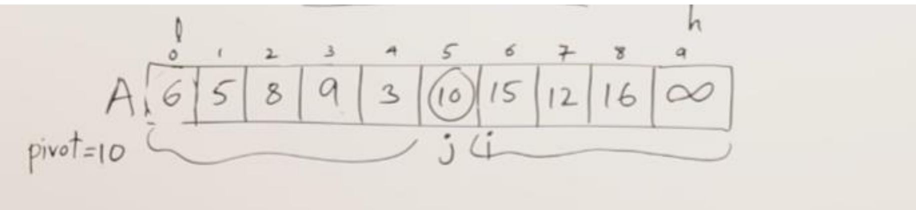


Sonrasinda I 10'dan buyuk deger olan 15' e gelir. J ise 10'dan kucuk olan 6'ya. Sonrasinda bu ikisi yine yer degistirir.



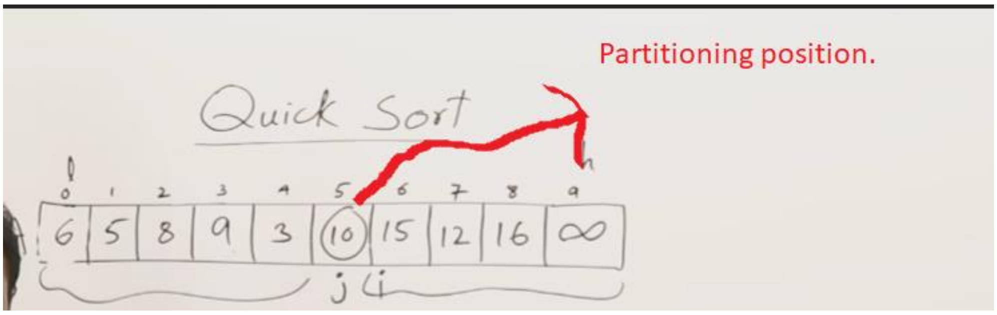
Burada artik I J'den buyuk hale geldi

Anlamamiz gereken sey 10 pivot degerinin yerini bulduk demektir.10 pivot degeri 5'inci indexe ve 5'inci indexte olan 6 ise 0'inci indexe gelir



10 sayisinin sol tarafindaki rakamlar kucuk oldu.Sag tarafindakiler ise buyuk rakamlar oldu.

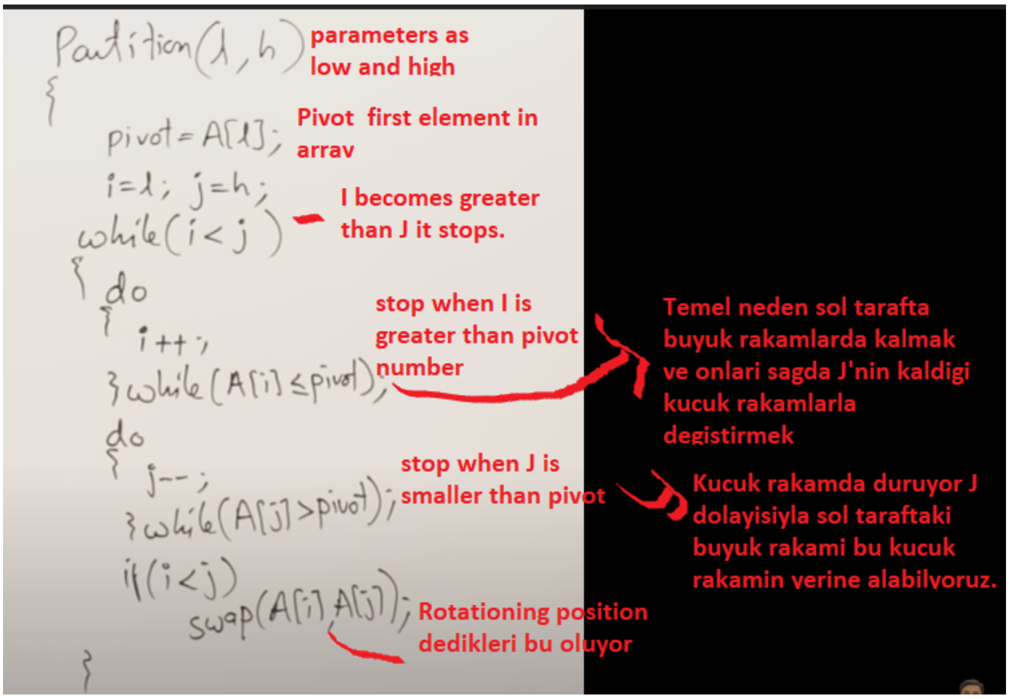
Burada 10 sayisi sorted digerleri henuz sorted degildir.


Partitioning position.

Low and high as parameter

Select pivot as low

I starts from left beginning J starts right end



```
Partition(l,h)
{
   pivot=A[l];
   i=l; j=h;
  while(i<j)
  { do
    {
      i++;
    }while(A[i]≤pivot);
    do
    {
      j--;
    }while(A[j]>pivot);
    if(i<j)
        swap(A[i],A[j]);
  }
```

parameters as low and high

Pivot first element in array

I becomes greater than J it stops.

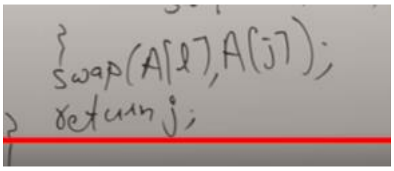
stop when I is greater than pivot number

Temel neden sol tarafta buyuk rakamlarda kalmak ve onlari sagda J'nin kaldigi kucuk rakamlarla degistirmek

stop when J is smaller than pivot

Kucuk rakamda duruyor J dolayisiyla sol taraftaki buyuk rakami bu kucuk rakamin yerine alabilyoruz.

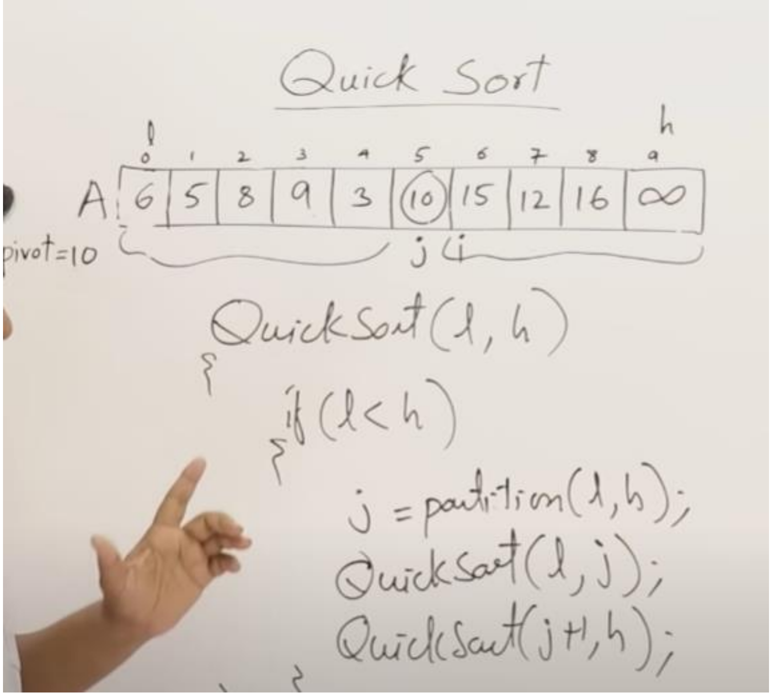
Rotationing position dedikleri bu oluyor



```
  }
  swap(A[l],A[j]);
  return j;
}
```

Bu son 2 kod ise Ustteki swap pivot sayisi ile J’nin bulundugu indexte olan sayiyi degistiriyor.Sonrasinda return J ise Sorted sayimizi bize donduruyor. BU YUKARIDAKILER PARTITIONING

HOW QUICK SORT WORKS?



	0	1	2	3	4	5	6	7	8	9
A	6	5	8	9	3	10	15	12	16	∞

```
QuickSort(l, h)
{
    if(l<h)
    {
        j = partition(l,h);
        QuickSort(l, j);
        QuickSort(j+1, h);
    }
}
```

Low ve high parametreleri aliniyor.

Low High’dan azsa (yani en az 2 eleman varsa)

Partition cagriliyor J=partition

Birinci quick sort Low to J yani sol taraf icin

Ikinci sag taraf icin

Sag tarafta infinity var ama sol tarafta yok bu 10 degeri sorted element olarak inifity olarak gorev aliyor.

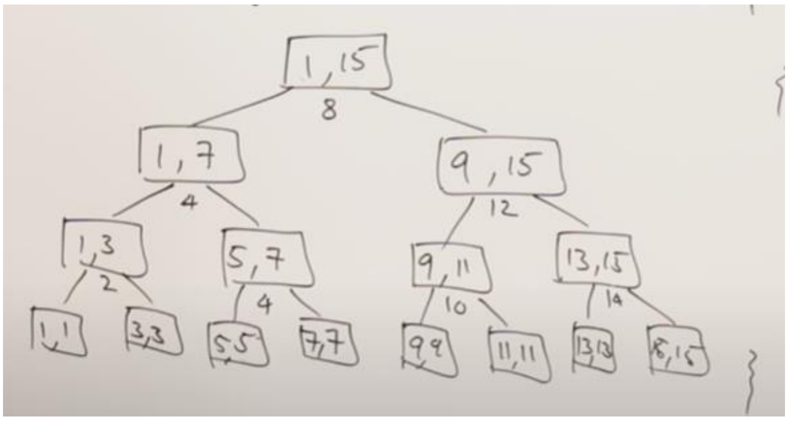
2.8.2 QuickSort Analysis

Quick sort is recursive and follows divide and conquer strategy.

I have a list of 15 elements.If I call quick sort as calling first index

Suppose its partitioning at the middle.

Our middle is 8

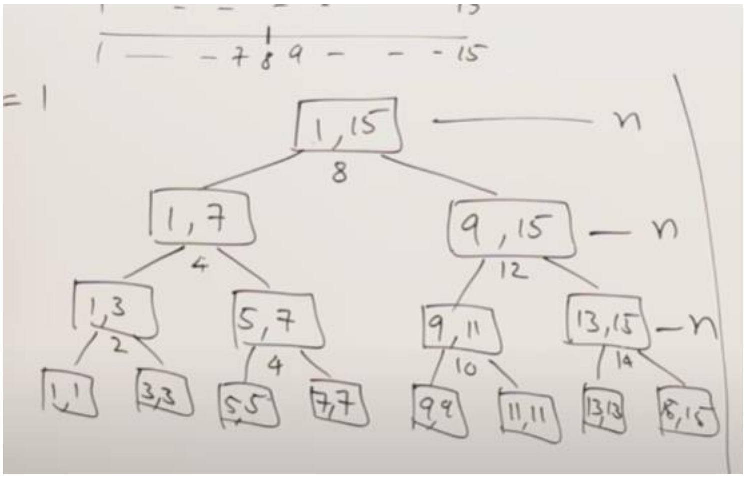


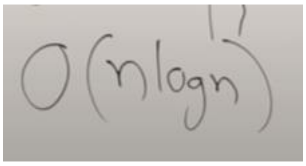
Bu sekilde calisiyur.

Time complexity: Her levelde N Her defasinda n /2’ye bolunuyor.

Eger partitioning her zamanda ortadaysa best case.

Nlogn ama bu sadece ortadaysa gecerli yani 8 olarak ortayi aldigimizda.

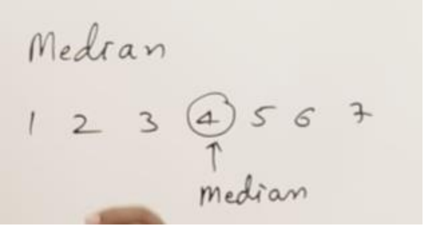




$O(n\log n)$

Lets see best case is possible or not. Median nedir?

The numbers are sorted here and middle of the is median



Median

1 2 3 4 5 6 7

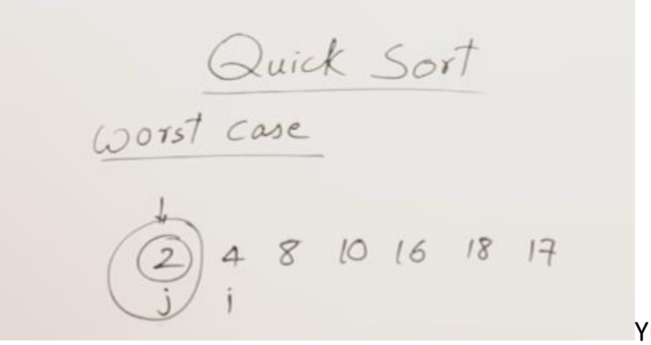
↑ median

Best case of quick sort is always the portioning should be in the middle. So if partitioning is in the middle means the element is selected as pivote should be a median.

How can you know median unless the list is sorted? We cannot know.

Achieving best case is not possible in quicksort you cannot achieve it it may randomly happen.

WORSTCASE OF QUICK SORT; suppose I have a list already sorted



Quick Sort

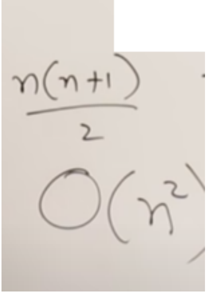
worst case

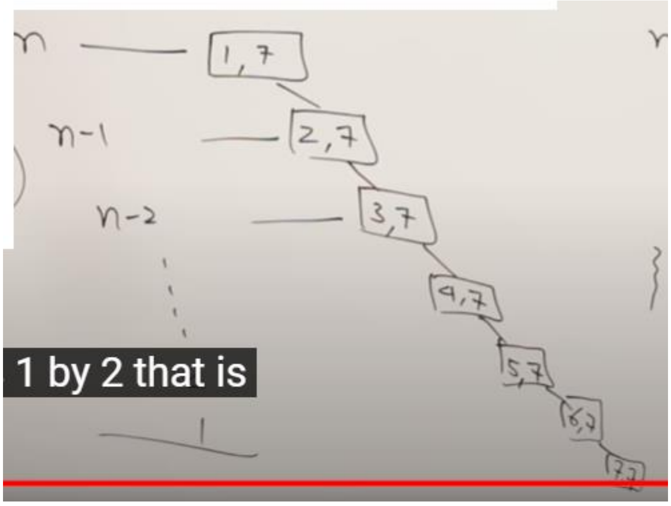
2 4 8 10 16 18 17

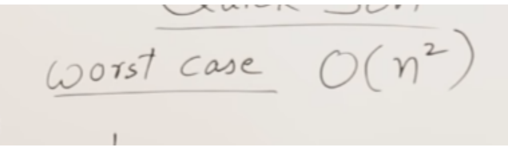
j i

Yukarida bahsettigimiz islemi yapiyoruz.Simdi J harfi sola dogru 2’den kucuk sayi aradi ama yok dolayisiyla partitioning 2 sayisinda olur ve diger sayilari sort yapmak zorunda kaliriz.

The partitioning will happen in the begin of the list.

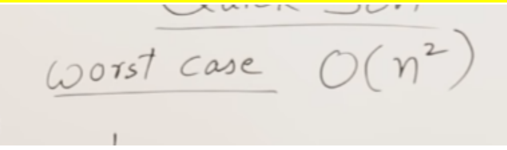






Bu aslinda liste coktan sorted olmussa gerceklesiyor. This is the problem of QUICK SORT

LISTE SORTED OLMUSKEN BUNU KULLANIRSAN WORST CASE

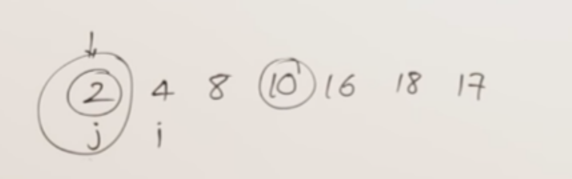


ALIYORSUN : BU ALGORITMANIN SORUNU

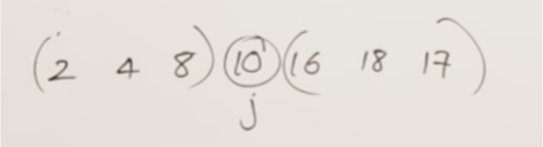
If you compare quick sort with merge sort merge sort does not having any best or worst cases. It was just average time. But here avarege time is log n the worst case is n squere.

HOW CAN WE REDUCE THISI PROBLEM?

Quick sortta sorted edilmis liste sorted edilirse tekrar n square zaman alirsin. Bunun cozumlerinden biri ilk elementi pivot olarak secmemek. We can select middle element of the pivote. Ortadaki elemani pivot olarak secebiliriz.



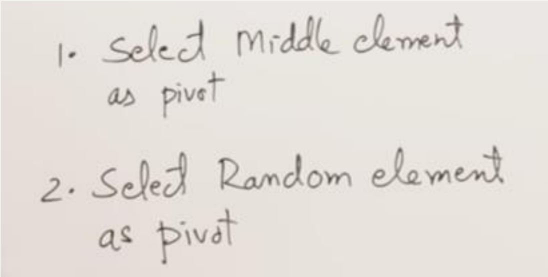
If we select middle element then what happens ?



Partitioning 10 sayisinin indexinde yapiliyor. We liste Ikiye bolunuyor

Ortadakini sectiginda N squared n log n’e donusuyor. The worst case become the best case.

THERE ARE TWO SUGGESTIONS FOR IMPROVING THE WORST CASE OF QUICK SORT.



1. Select Middle element as pivot

2. Select Random element as pivot

1-ortadaki elemani secmek eger liste coktan sorted edilmisse ortadaki listeyi secebiliriz. Log n aliriz.

2-Ya da random olarak bir sayi secmek bu seni log n’e veya n squared’e goturebilir. The worst time taken by quick sort is $O(n^2)$ ALWAYS

About space complexity. It does not need extra space. But it will use stack. What could be the maximum size of stack. In worst case it may be $O(n^2)$ what is the minimum base is logn

Quick sort takes stock sizes that is log n to n

You can know this just now have shown trees in best case. The heigtt of a tree is log n so the size of the stack depends on the height of a tree. And in worst case it was n so maximum size of the stack maybe n