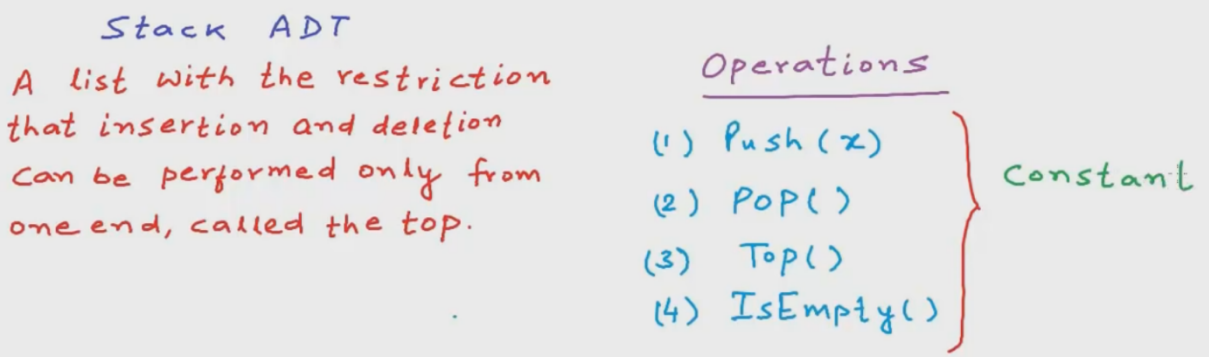
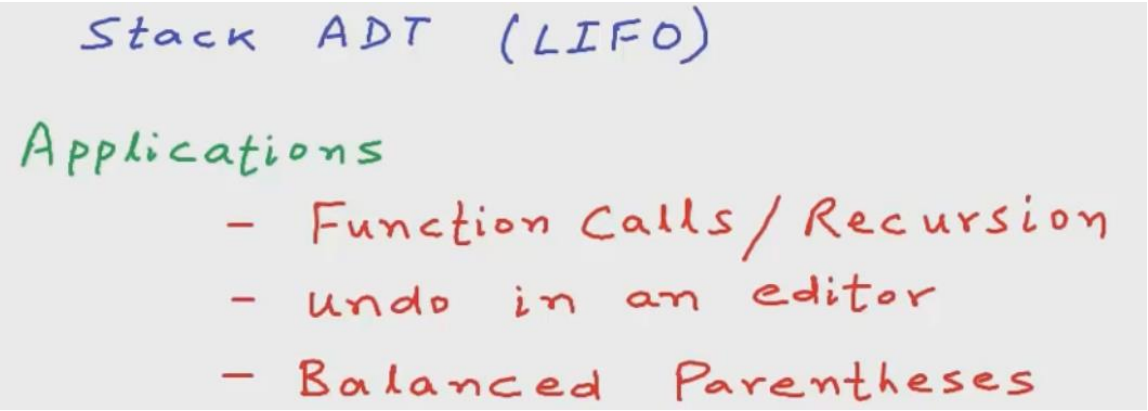
Stack ADT=Abstract Data type **LAST IN FIRST OUT**

Complexity	
Pushing	O(1)
Popping	O(1)
Peeking	O(1)
Searching	O(n)
Size	O(1)

Push ekler pop siler top ise listenint en basindaki elemani dondurur is empty stacking bos olup olmadigini sorgular. False olarak True olarak geri doner. Buradaki butun operasyonlar constant timedir.

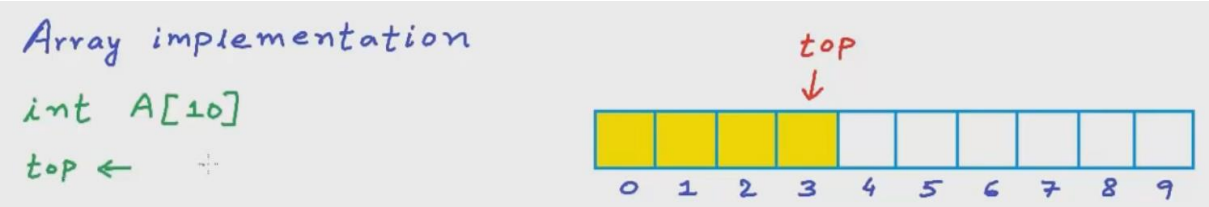




Array implementation of stacks

We can implement stacks using Arrays:

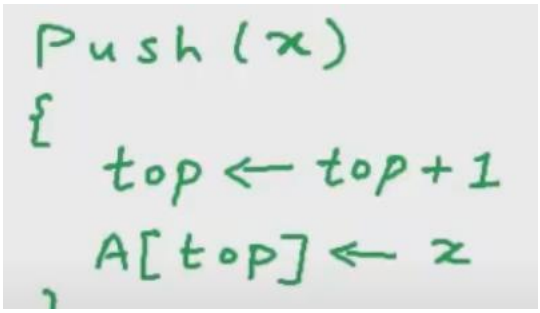
A=Arrays B=Linked lists

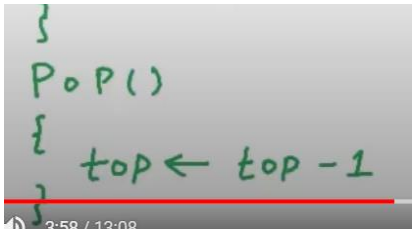


I am going to use this array to store stack. Any point some part of this array starting index zero till an index marked as 'top' will be my stack. We can create a variable named top to store to store index of top of stack.

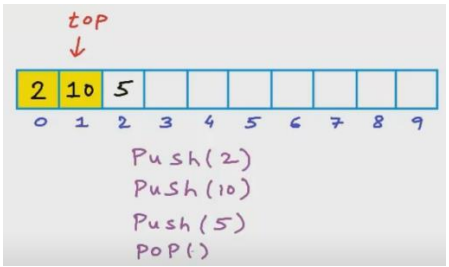
For an empty stack top is set as -1 Burada push function ile top yerine top +1 ekliyoruz.

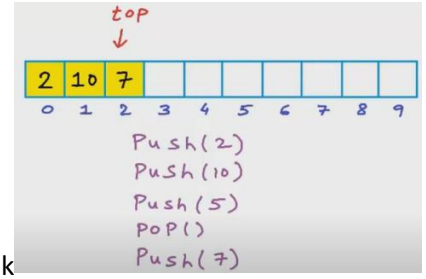
Push 2 yaptigimizi dusunelim; her push yaptigidinda top bir sonraki sayiya geciyor mantik bu

Push icin 

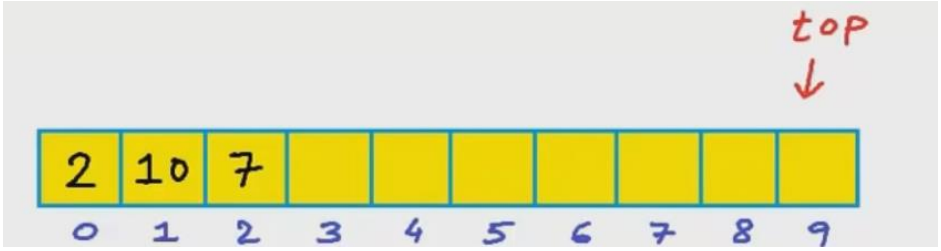
pop icin 

Pop kullandiktan sonra onceden 5 sayisinde olan topumuz bir onceki sayiya doner



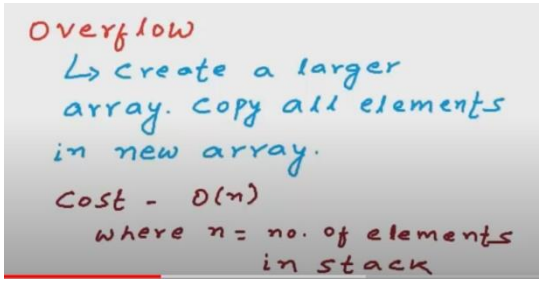
Simdi yeniden push yaparsak 

Pop ve push funcitionu constant time aliyur Time complexity O(1) Arraye yalnizca arrayda bos yer kaldiysa push yapabiliriz. Stacking tum arrayi tukettigi durumlarda top arrayin en sonunda olacadir.

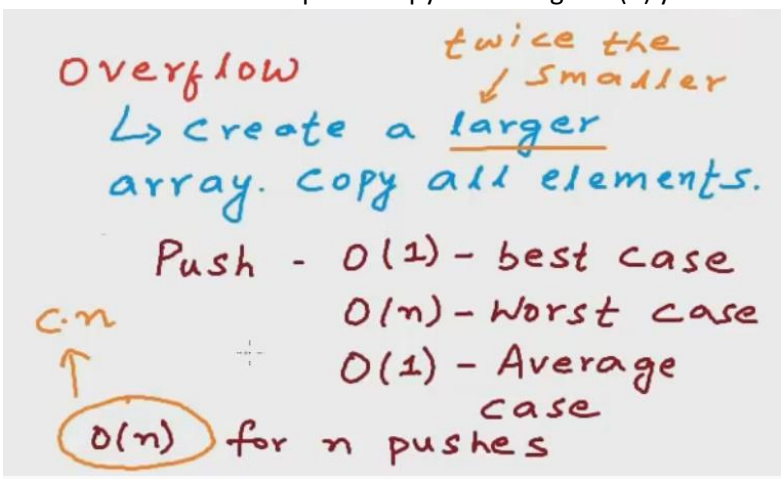


A further push is not possible because it will result an overlflow this is one limit based array to avoid an overflow we can always create an erray for that we will have to be reasonably sure that stack will not grow Bu durumu onlemek icinde ornegin kod yazarken push fonctionun arrrayin exhausted olup olmadigini ogrendini yazan bir kod yazabiliriz.

Bunun yerine dynamic arrayi kullanabiliriz. Overflow durumlarinda yeni daha buyuk bir array yaratabiliriz diger stackin icerigini bu yeni arraya aktararak bunu yapariz.

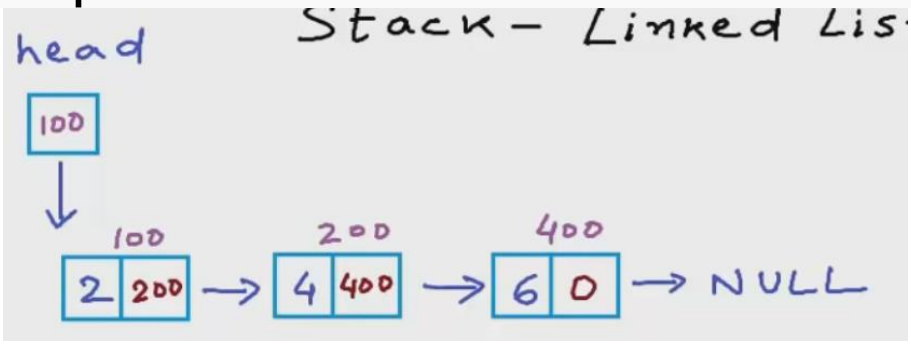


Burada onemli olan bize bunun timpe complexity maliyeti O(n) olacadir. Means time taken to copy element from last array to new array would be proportional to number of arrays in stack . or the size of the smaller array.because stack will occupy the stack of whole array. There must be strategy to decide size of array. Bunun oluru onceki arrayimizin 2 kati bir array yaratmaktir. Ornegin onceki array 4 ise yeni olan 8 o da yetmediyse 16 tane gibi. Belli bir senaryoda push constant time iken yeni bir array eklenmesi ve bunlarin hepsinin kopyalanmasi gibi O(n) yani lineer timedir.



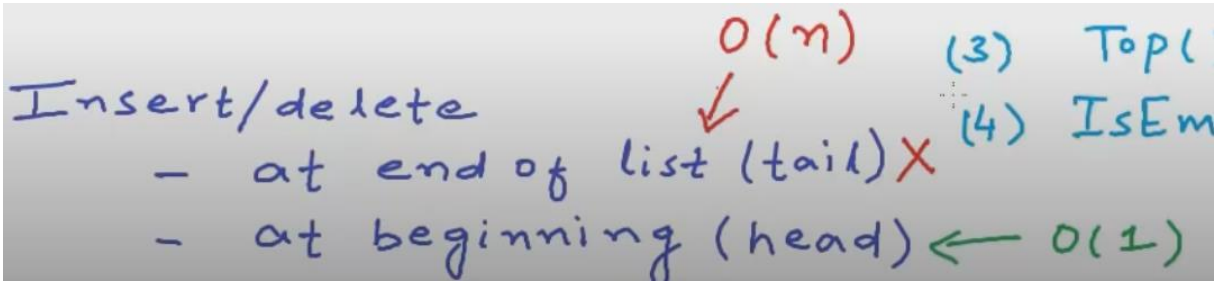
En iyi durumda constant time kotu durumda n time. Fakat yine de O(1) avarage case ile ile bulabilriiz If we will calculate time taken for n pushes then it will be proportional N remember n is the number of elements in the stack. Burada sari isaretli O(n) su anlama geliyor time taken will be very close to some constant times simple words time taken will be proportional to n

Data Structures: Linked List implementation of stacks



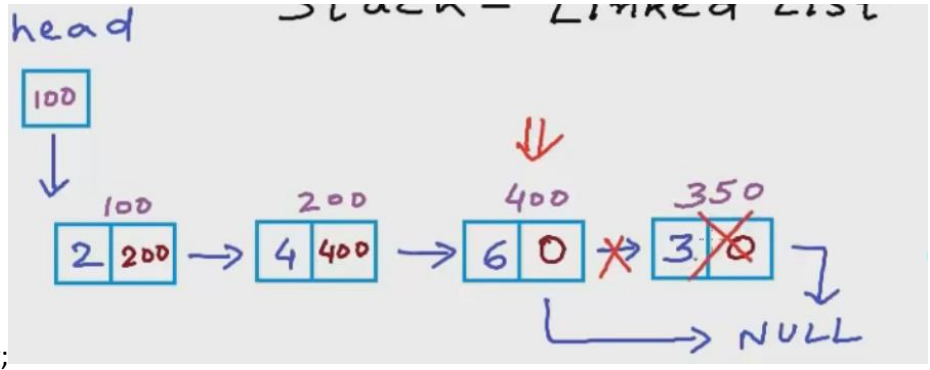
Linked listlerde 1 bolge data 1 bolge diger node'un addresini iceriyor.unlike arrays linked lists are not if fixed size and element in linked list are not stored in one contiguous block of memory

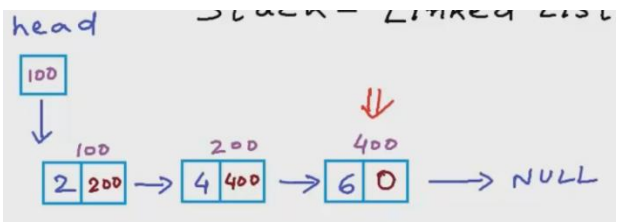
To insert element in linked list for a stack we want that insertion and deletion must always happen from the same end we can use our linked list as stack if we always insert and delete at same end.we have two options we can delete or add at the end of the list ya da basindan ekleyip cikarabiliruz.



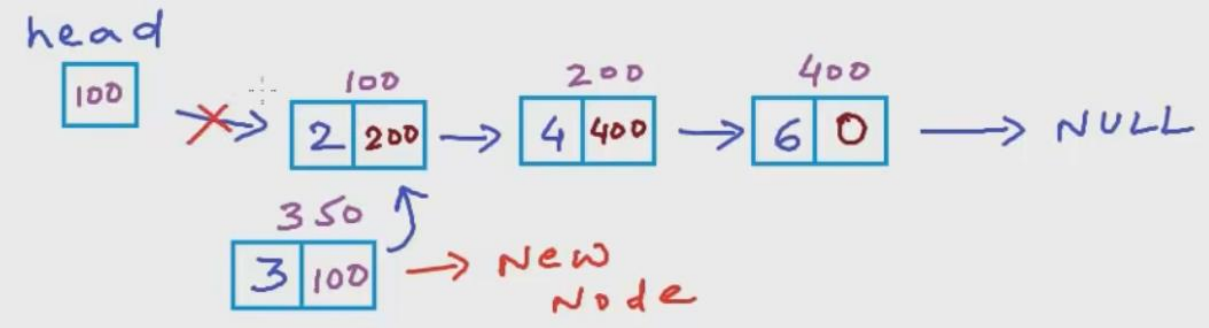
***Inserting a node at end of linked list is not constant time. Hem insertion hem deletion to at end of list is O(n) yani lineer time. Stack icin elbette constant time ama mevzu linked listlere geldiginde bunun lineer time O(n)

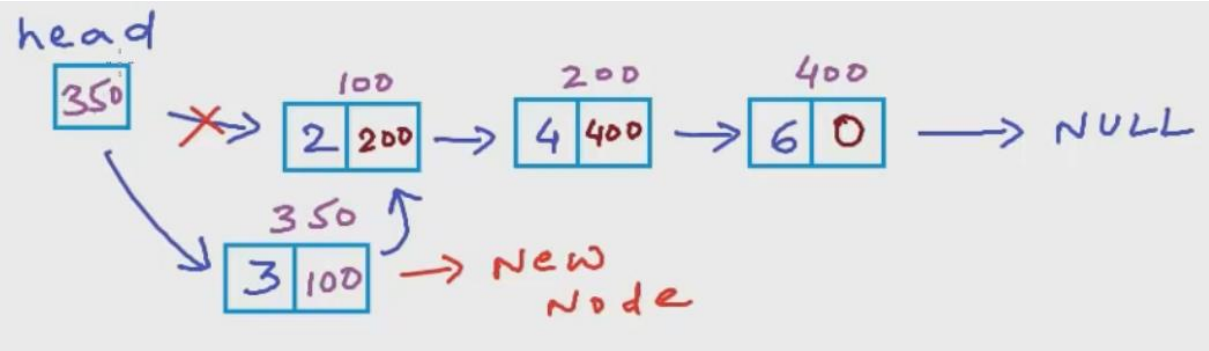
Deleting;





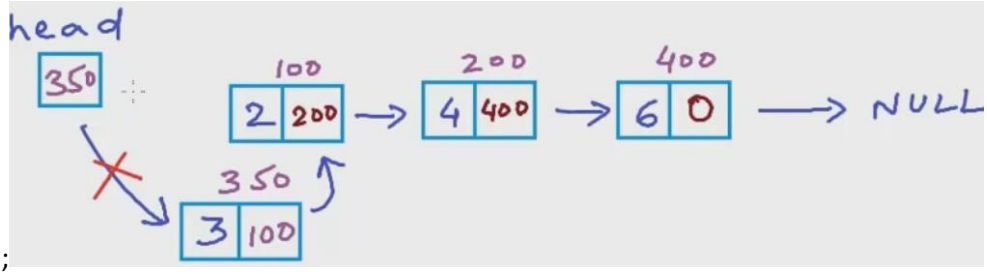
Bunun yaninda baslangicta insert ve delete O(1) yani constant time.

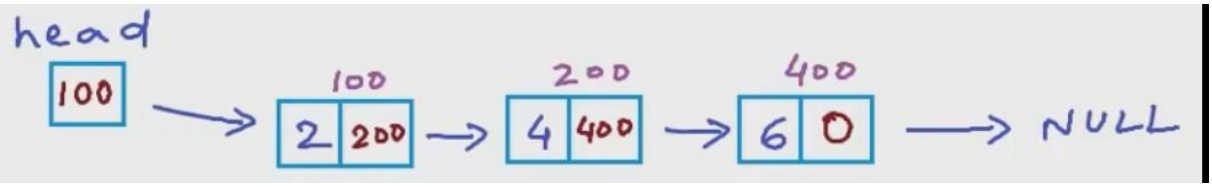




Yeni node ekliyoruz birinci resimde head ile ikinci node baglantisni kes alta ekledgimiz yeni node' a eski head sayimiz 100 yaz sonrasinda yen inode sayimiz 350'yi 100' un yeirne yaz.

Delete a node ;



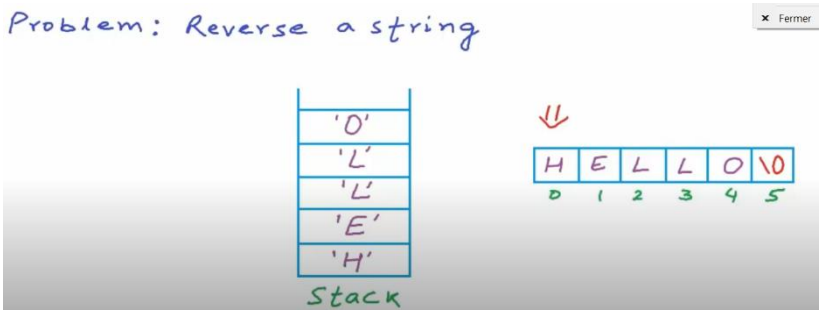


Linked list implementation of stack is pretty straightforward all we need is insert and delete beginning so head of linked list is top of stack.

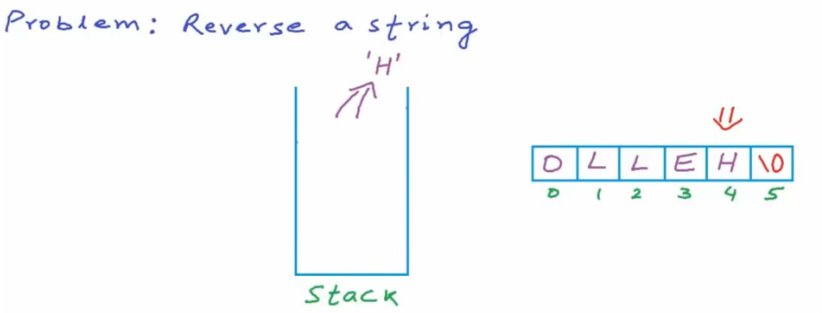
Reverse a string or linked list using stack.

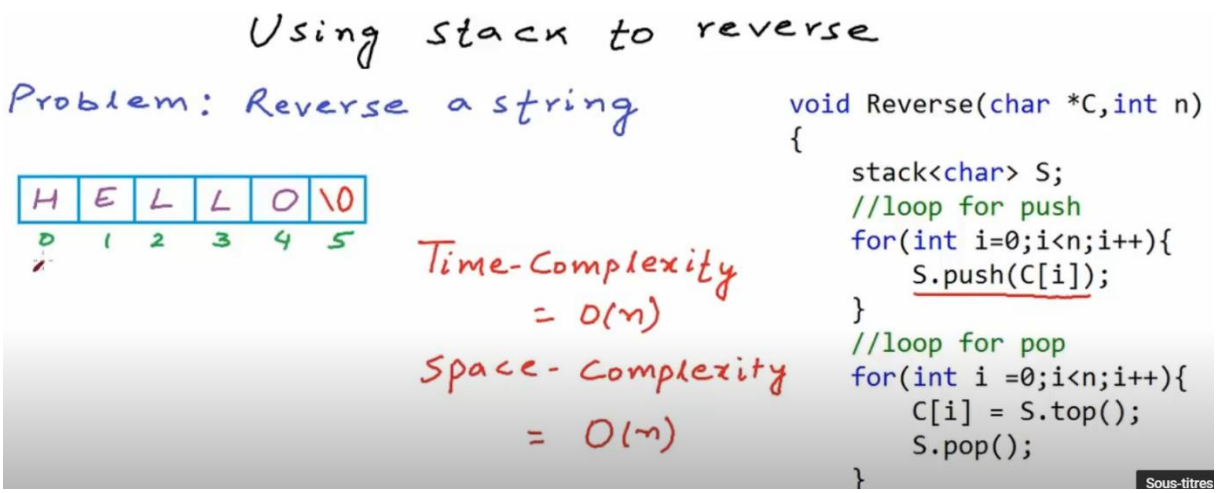
A stack can be used to reverse a list or collection. Or simply traverse a list or collection.

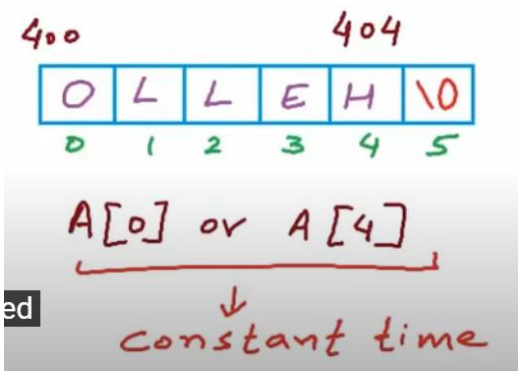
Problem; Reverse a string



Popu kullanarak simdi her harfi alalim

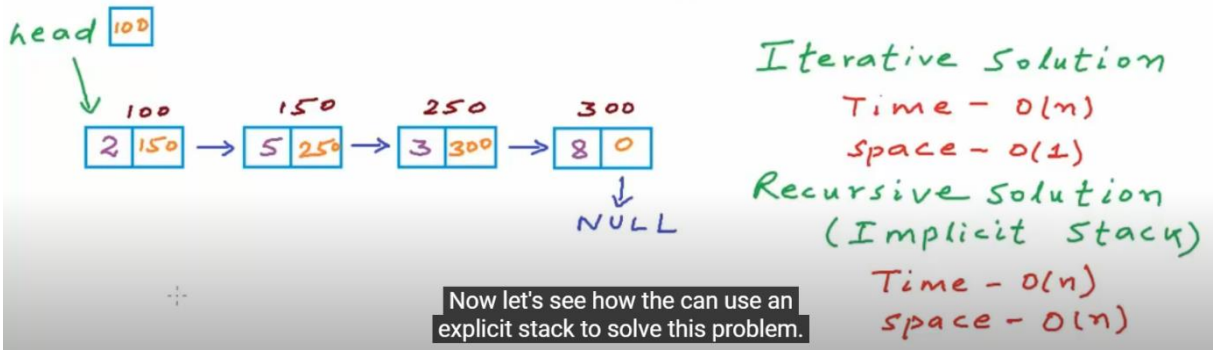


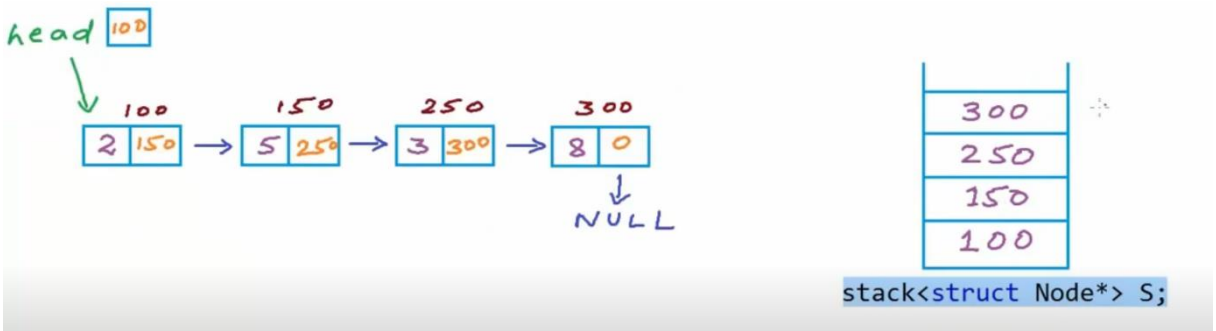




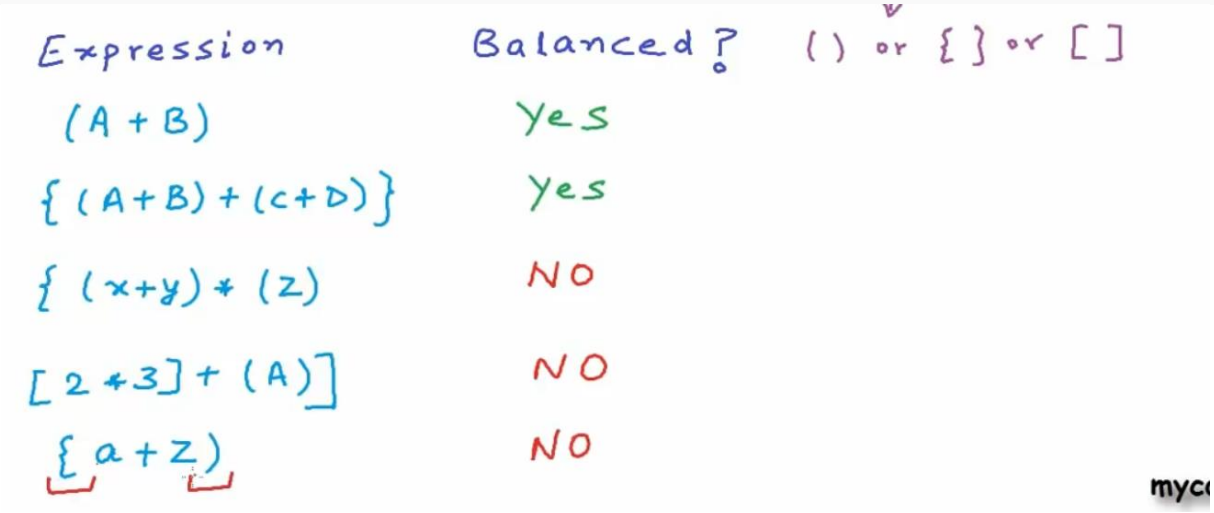
Bu yandaki Array ornegimiz.

Linked liste bakalim.



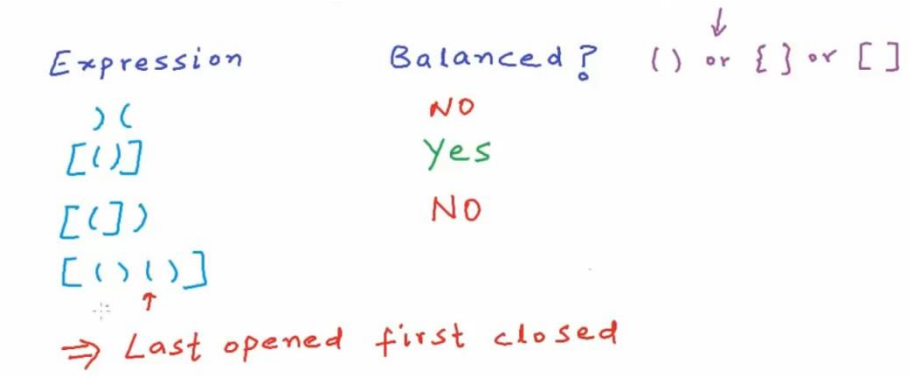


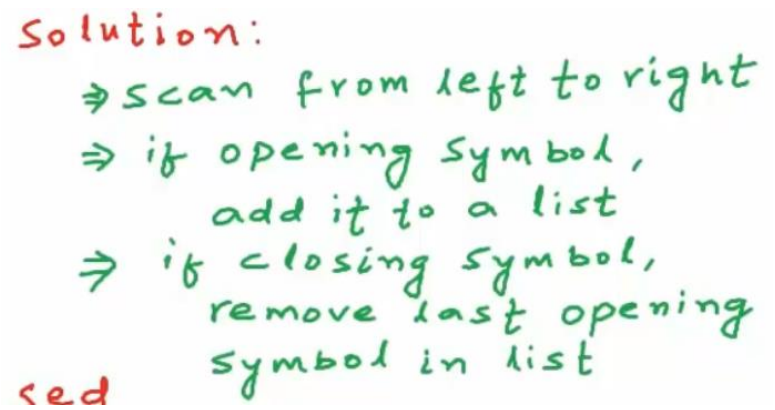
Check for balanced parentheses using stack



Bu expressionlar balanced mi evet cunku ornegin sozluk isaretiyle acilan sozluk isqretiyle kapatilmis yine tuple isaretiyle olan tuple ile kapatilmis Fakat ucuncu sirada sozluk kapatma isareti olmadigi icin balanced degildir.

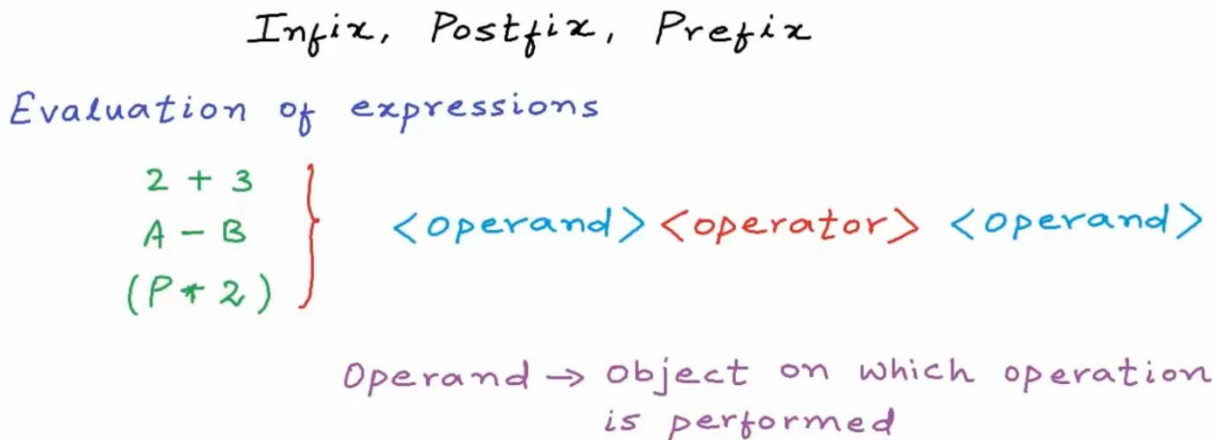
Compiler burada parentez isaretlerini vs kontrol ediyor.



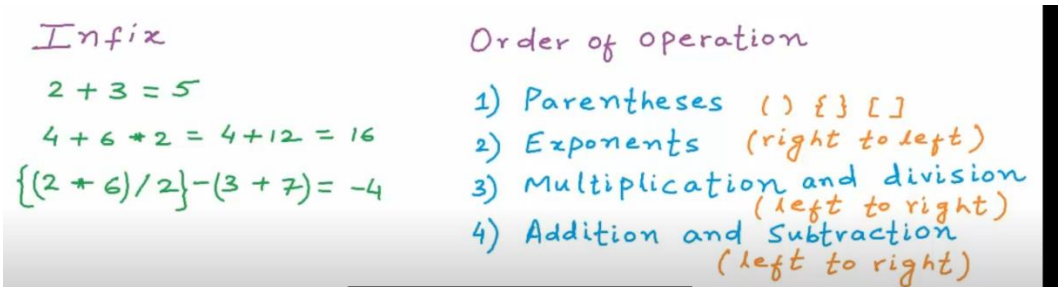


Infix, Prefix and Postfix

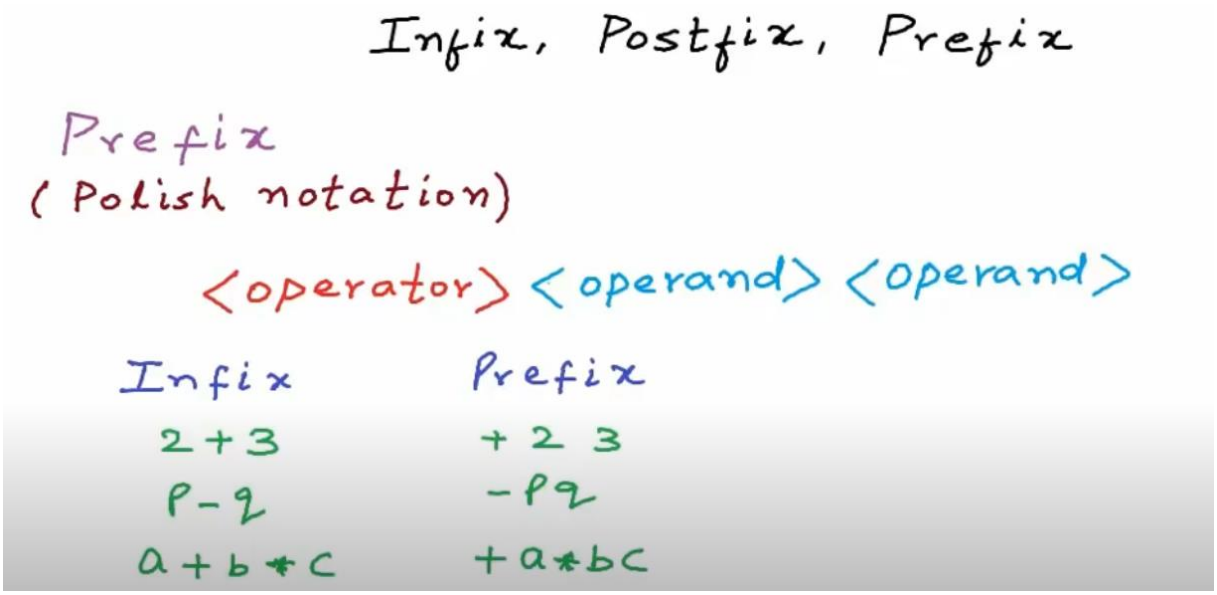
Infix notation



Burada operator yani toplama olur cikarma vs olur ortada oluyor sayilara veya variableslar ise saginda veya solunda.



Prefix notation(polish notation)



Bu da polisj notation operatorler toplama carpma gibi onunde yer aliyur.

Postfix notation: Bu 1950'li yillarda computer scientistler taraifndan onerildi. Zamandan ve yerden kazandigi icin icat edilmistir.