

ECAP770

ADVANCE DATA STRUCTURES

Ashwani Kumar
Assistant Professor



Learning Outcomes



After this lecture, you will be able to

- understand Applications of stacks
 - Parenthesis checker
 - Quick sort

Parenthesis checker

- Parenthesis checker is used for balanced Brackets in an expression
- The balanced parenthesis means that when the opening parenthesis is equal to the closing parenthesis, then it is a balanced parenthesis.

Parenthesis

- $(a+b^*(c/d))$
- $[10+20^*(6+7)]$
- $(x+y)/(c-d)$

Balanced parenthesis

- $A = (10+20)$
- In the above expression there is one opening and one closing parenthesis means that both opening and closing brackets are equal; therefore, the above expression is a balanced parenthesis.

Unbalanced parenthesis

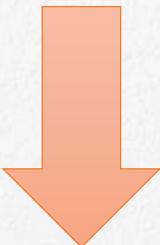
- $A = [(10+20)$
- The above expression has two opening brackets and one closing bracket, which means that both opening and closing brackets are not equal; therefore, the above expression is unbalanced.

Algorithm

- Initialize a character stack.
- Now traverse the expression string exp.
- If the current character is a starting bracket ('(' or '{' or '[') then push it to stack.
- If the current character is a closing bracket (')' or '}' or ']') then pop from stack and if the popped character is the matching starting bracket then balanced else brackets are not balanced.
- After complete traversal, if there is some starting bracket left in stack then not balanced

Sorting

A Sorting process is used to rearrange a given array or elements based upon selected algorithm/sort function.

25 30 15 12 16

12 15 16 25 30

Quick Sort

- QuickSort is used for sorting a list of data elements. It follows a divide and conquer principle.
- A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value.

Quick sort

There are many different versions of quick Sort that pick pivot in different ways.

- Always pick first element as pivot.
- Always pick last element as pivot
- Pick a random element as pivot.
- Pick median as pivot.

Algorithm

```
quickSort(arr, beg, end)           plIndex = beg - 1  
if (beg < end)                   for i = beg to end-1  
    pivotIndex = partition(arr,beg,  
end)                           if arr[i] < pivot  
                                swap arr[i] and  
                                arr[plIndex]  
                                plIndex++  
                                swap pivot and  
                                arr[plIndex+1]  
                                return plIndex + 1  
quickSort(arr, beg, pivotIndex)  
quickSort(arr, pivotIndex + 1,  
end)  
  
partition(arr, beg, end)  
set end as pivotIndex
```

That's all for now...