Difference between Subarray, Subsequence and Subset

In this post, we will discuss the difference between a subarray/substring, a subsequence and a subset.

1. Subarray

A subarray is a slice from the array which is contiguous (i.e. occupy consecutive positions) and inherently maintains the order of elements. For example, the subarrays of the array {1, 2, 3} are {1}, {1, 2}, {1, 2, 3}, {2}, {2, 3}, and {3}.

Below is a C and Java program to generate all subarrays of the specified array:

C

```c
#include <stdio.h>

// Function to print a subarray formed by [start, en
void printSubarray(int arr[], int start, int end)
{
    printf("{");

    for (int i = start; i < end; i++)
        printf("%d, ", arr[i]);

    printf("%d}, ", arr[end]);
}

// Function to print all subarrays of the specified
void printallSubarrays(int arr[], int n)
{
    // consider all subarrays starting from i
    for (int i = 0; i < n; i++)
    {
        // consider all subarrays ending at j
        for (int j = i; j < n; j++)
            printSubarray(arr, i, j);
    }
}

// Program to print all subarrays of the specified a
int main()
```

Browse

Adobe Algorithm Amazon BFS Binary Search Bit Hacks DFS FIFO Google Greedy Hashing Intro JSON LCS LIFO Maze Memoized Microsoft Must Know Priority Queue Probability Recursive Searching Sliding Window Tabulation Tricky Trie

```
{
    int arr[] = { 1, 2, 3, 4, 5 };
    int n = sizeof(arr)/sizeof(arr[0]);

    printallSubarrays(arr, n);

    return 0;
}
```

Download Code Run

Output:

```
{1}, {1, 2}, {1, 2, 3}, {1, 2, 3, 4}, {1, 2, 3, 4, 5},
{2}, {2, 3}, {2, 3, 4}, {2, 3, 4, 5}, {3}, {3, 4}, {3, 4,
5}, {4}, {4, 5}, {5}
```

Java

```
import java.util.Arrays;
import java.util.List;

class Main
{
    // Function to print all sub-arrays of the speci
    public static void printallSubarrays(List<Intege
    {
        // consider all sub-arrays starting from i
        for (int i = 0; i < arr.size(); i++)
```

```
11          {
12              // consider all sub-arrays ending at j
13              for (int j = i; j < arr.size(); j++)
14              {
15                  // Function to print a sub-array for
16                  System.out.println(arr.subList(i, j
17              }
18          }
19      }
20
21      // Program to print all sub-arrays of the specif
22      public static void main(String[] args)
23      {
24          List<Integer> arr = Arrays.asList(1, 2, 3, 4
25          printallSubarrays(arr);
26      }
27  }
```

Download Code Run

Output:

```
[1]
[1, 2]
[1, 2, 3]
[1, 2, 3, 4]
[1, 2, 3, 4, 5]
[2]
[2, 3]
```

```
[2, 3, 4]
[2, 3, 4, 5]
[3]
[3, 4]
[3, 4, 5]
[4]
[4, 5]
[5]
```

Please note that there are exactly n*(n + 1)/2 subarrays in an array of size n. Also, there is no such thing as contiguous subarray. The prefix contiguous is sometimes applied to make context more clear. So a contiguous subarray is just an another name for a subarray.

2. Substring

A substring of a string S is a string S' that occurs in S. A substring is almost similar to a subarray but it is in context of strings.

For example, the substrings of the string 'apple' are 'apple', 'appl', 'pple', 'app', 'ppl', 'ple', 'ap', 'pp', 'pl', 'le', 'a', 'p', 'l', 'e', and ''.

Below is C++ and Java program that generates all non-empty substrings of the specified string:

C++

```cpp
#include <iostream>
using namespace std;

// Function to print all non-empty substrings of the
void printallSubstrings(string str)
{
    int n = str.length();

    // consider all substrings starting from i
    for (int i = 0; i < n; i++)
    {
        // consider all substrings ending at j
        for (int j = i; j < n; j++)
        {
            cout << "'" << str.substr(i, j - i + 1)
        }
    }
}

// Program to print all non-empty substrings of the
int main()
{
    string str = "techie";
    printallSubstrings(str);

    return 0;
}
```

Download Code Run

Output:

‘t’, ‘te’, ‘tec’, ‘tech’, ‘techi’, ‘techie’, ‘e’, ‘ec’, ‘ech’, ‘echi’, ‘echie’, ‘c’, ‘ch’, ‘chi’, ‘chie’, ‘h’, ‘hi’, ‘hie’, ‘i’, ‘ie’, ‘e’

Java

```java
class Main
{
    // Function to print all non-empty sub-strings o
    public static void printallSubStrings(String str
    {
        int n = str.length();

        // consider all sub-strings starting from i
        for (int i = 0; i < n; i++) {
            // consider all sub-strings ending at j
            for (int j = i; j < n; j++) {
                System.out.print("'" + str.substring
            }
        }
    }

    // Program to print all non-empty sub-strings of
    public static void main(String[] args)
    {
```

```
20          String str = "techie";
21          printallSubStrings(str);
22      }
23  }
```

Download Code Run

Output:

‘t’, ‘te’, ‘tec’, ‘tech’, ‘techi’, ‘techie’, ‘e’, ‘ec’, ‘ech’, ‘echi’, ‘echie’, ‘c’, ‘ch’, ‘chi’, ‘chie’, ‘h’, ‘hi’, ‘hie’, ‘i’, ‘ie’, ‘e’

3. Subsequence

A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. For example, {A, B, D} is a subsequence of sequence {A, B, C, D, E} that is obtained after removing {C} and {E}.

People are often confused between a subarray and a subsequence. A subarray will always be contiguous but a subsequence need not be contiguous. That is, unlike subarrays, subsequences are not required to occupy consecutive positions within the original sequences. But we can say that both contiguous subsequence and subarray are same.

In other words, the subsequence is a generalization of substring or substring is a refinement of the subsequence. For example, {A, C, E} is a subsequence of {A, B, C, D, E}, but not a substring and {A, B, C} is both a subarray and a subsequence.

Please note that a Subsequence can be in context of both arrays and strings. Generating all subsequences of an array/string is equivalent to generating power set of the array/string. For a given set S, the power set can be found by generating all binary numbers between 0 to 2^n-1 where n is the size of the given set. This is demonstrated below:

C++

```cpp
#include <iostream>
#include <cmath>
using namespace std;

```

```
// Function to print all subsequences of the specifi
void findPowerSet(string str)
{
    int n = str.length();

    // N stores total number of subsets
    int N = pow(2, n);

    // generate each subset one by one
    for (int i = 0; i < N; i++)
    {
        cout << "'";

        // check every bit of i
        for (int j = 0; j < n; j++)
        {
            // if j'th bit of i is set, print S[j]
            if (i & (1 << j))
                cout << str[j];
        }
        cout << "', ";
    }
}

// Program to print all subsequences of the specifie
int main()
{
    string str = "apple";
    findPowerSet(str);
}
```

Download Code

Run

Output:

”, ‘a’, ‘p’, ‘ap’, ‘p’, ‘ap’, ‘pp’, ‘app’, ‘l’, ‘al’,
‘pl’, ‘apl’, ‘pl’, ‘apl’, ‘ppl’, ‘appl’, ‘e’, ‘ae’, ‘pe’,
‘ape’, ‘pe’, ‘ape’, ‘ppe’, ‘appe’, ‘le’, ‘ale’, ‘ple’,
‘aple’, ‘ple’, ‘aple’, ‘pple’, ‘apple’

Java

```java
import java.util.ArrayList;
import java.util.List;

class Main
{
    // Function to print all sub-sequences of the sp
    public static void findPowerSet(String str)
    {
        int n = str.length();

        // N stores total number of subsets
        int N = (int)Math.pow(2, n);
        List<String> result = new ArrayList<>();

        // generate each subset one by one
        for (int i = 0; i < N; i++)
        {
            StringBuilder sb = new StringBuilder();

            // check every bit of i
            for (int j = 0; j < n; j++)
```

```
22              {
23                  // if j'th bit of i is set, print S[
24                  if ((i & (1 << j)) != 0) {
25                      sb.append(str.charAt(j));
26                  }
27              }
28              result.add("'" + sb.toString() + "'");
29          }
30
31          System.out.println(result);
32      }
33
34      // Program to print all sub-sequences of the spe
35      public static void main(String[] args)
36      {
37          String str = "apple";
38          findPowerSet(str);
39      }
40  }
```

Download Run Code

Output:

[”, ‘a’, ‘p’, ‘ap’, ‘p’, ‘ap’, ‘pp’, ‘app’, ‘l’, ‘al’, ‘pl’, ‘apl’, ‘pl’, ‘apl’, ‘ppl’, ‘appl’, ‘e’, ‘ae’, ‘pe’, ‘ape’, ‘pe’, ‘ape’, ‘ppe’, ‘appe’, ‘le’, ‘ale’, ‘ple’, ‘aple’, ‘ple’, ‘aple’, ‘pple’, ‘apple’]

4. Subset

A subset is any possible combination of the original set. The term subset is often used for subsequence but this is wrong. A subsequence always maintain the relative order of elements of the array (i.e. increasing index) but there is no such restriction on a subset. For example, `{3, 1}` is a valid subset of `{1, 2, 3, 4, 5}` but it is neither a subsequence or a subarray.

It is worth noting that all subarrays are subsequences and all subsequences are subset but the reverse is not true. For instance, the subarray `{1, 2}` of the array `{1, 2, 3, 4, 5}` is also a subsequence and a subset.