

20 Apr 2023

<https://my.newtonschool.co/playground/code/mxg4viddct70/>

NS 18 Apr 2023

Main.java

```
11  return fact;
12  }
13
14  public static long calc_rank(String str)
15  {
16      if(str==null)
17          return 0;
18      long rank = 1;
19      for (int i=0; i<str.length(); i++)
20      {
21          // count all smaller characters than str[i] to the right of i
22          int count = 0;
23          for(int j=i+1; j<str.length(); j++)
24              if(str.charAt(i) > str.charAt(j))
25                  count++;
26
27          rank += count * factorial(str.length()-1-i);
28      }
29      return rank;
30  }
31
32  public static void main(String[] args)
33  {
34      String str = "BA";
35      System.out.println("Rank of "+str+" is "+calc_rank(str));
36  }
```

BA

str

0	1
B	A

rank = 1 + (1 * fact(2-1-0)) = 1 + (1 * fact(1)) = 2

count = 0

2 + (0 * fact(2-1-1)) = 2 + 0 = 2

i = 0 1

j = 1 2

Run | Debug

Ln 34, Col 22 Spaces: 4 UTF-8 LF {} Java

NS 18 Apr 2023

Main.java

```
1  // https://my.newtonschool.co/playground/code/mxg4viddct70/
2  public class Main
3  {
4      // we use the factorial funtion to calculate the number of permutations of
5      // the remaining characters in the string, from the current index to the end
6      public static long factorial(int n)
7      {
8          long fact = 1;
9          for(int i=1; i<=n; i++)
10             fact = fact*i;
11         return fact;
12     }
13
14     public static long calc_rank(String str)
15     {
16         if(str==null)
17             return 0;
18         long rank = 1;
19         for (int i=0; i<str.length(); i++)
20         {
21             // count all smaller characters than str[i] to the right of i
22             int count = 0;
23             for(int j=i+1; j<str.length(); j++)
24                 if(str.charAt(i) > str.charAt(j))
25                     count++;
26
27             rank += count * factorial(str.length()-1-i);
28         }
29         return rank;
30     }
31 }
```

DCBA

A _ _ _

1 2 3 4

3! = 3 * 2 = 6

no. of ways to fill 3 positions

1. BCD
2. BAC
3. CBD
4. CDB
5. DBC
6. DCB

CBA

A	B	C
1	2	3

2! = 2 * 1 = 2

no. of ways to fill 2 positions

Ln 27, Col 29 Spaces: 4 UTF-8 LF {} Java

rank += count * factorial(str.length()-1-i);

$$\begin{aligned} &= 1 + [2 \times f(2)] = 5 + [1 \times f(1)] \\ &= 1 + [2 \times 2] = 5 + 1 \\ &= 1 + 4 = 6 \end{aligned}$$

The screenshot shows an IDE with a Java file named `Main.java`. The code defines a method `calc_rank(String str)` that calculates the rank of a string. The code is annotated with handwritten notes and a diagram.

Handwritten notes and annotations:

- `count = 0` is written above the `count` variable.
- `i = 0` is written above the `i` variable.
- `j = 1` is written above the `j` variable.
- `rank = 1` is written above the `rank` variable.
- `rank = 5` is written above the `rank +=` statement.
- A diagram shows the string `str` as a table with columns indexed 0, 1, 2 containing characters 'C', 'B', 'A' respectively.
- A box highlights the loop `for (int j=i+1; j<str.length(); j++)` with the note `j = 1` and `rank = 5`.

```
13     return fact;
14 }
15
16 public static long calc_rank(String str)
17 {
18     if(str==null)
19         return 0;
20     long rank = 1;
21     for (int i=0; i<str.length(); i++)
22     {
23         // count all smaller characters than str[i] to the right of i
24         int count = 0;
25         for(int j=i+1; j<str.length(); j++)
26             if(str.charAt(i) > str.charAt(j))
27                 count++;
28
29         rank += count * factorial(str.length()-1-i);
30     }
31     return rank;
32 }
33
34 public static void main(String[] args)
35 {
36     String str = "UDVNXVWE";
37     System.out.println("Rank of "+str+" is "+calc_rank(str));
38 }
```

