

Permutations-II

```
def permutation2(n, r):
    items = [0] * n
    permutationUtil(n, r, items, 0, '', 0)
    return

def permutationUtil(n, r, items, totalItems, ssf, cb):
    if cb == n:
        if totalItems == r:
            print(ssf)
            return

    for i in range(0, r):
        if items[i] == 0:
            items[i] = 1
            permutationUtil(n, r, items, totalItems + 1, ssf + str(i +
1), cb + 1)
            items[i] = 0

    permutationUtil(n, r, items, totalItems, ssf + '-', cb + 1)
```

```
permutation2(4, 2)
```

```
'''
```

Now this is another way of solving the permutations.

In this we are allowing the box to choose.

So box will have 2 choices:

1. Choose any of the item from r item
2. Not choose the item at all (permutationUtil(n, r, items, totalItems, ssf + '-', cb + 1))

So, we will allow the boxes to choose from the items.

Now, because we have different r items, we can choose any of the r items. If the items were identical, the box had to choose any item from r identical items.

In short the Euler tree wouldn't have spread far in case of identical item.

```
'''
```

'''

Now this is another way of solving the permutations. In this we are allowing the box to choose. So box will have 2 choices:

1. Choose any of the item from r item
2. Not choose the item at all (permutationUtil(n, r, items, totalItems, ssf + '-',cb+1))

So, we will allow the boxes to choose from the items. Now, because we have different r items, we can choose any of the r items. If the items were identical, the box had to choose any item from r identical items. In short the Euler tree wouldn't have spread far in case of identical item.

'''