코딩 테스트 # 46

문제

문자배열이 주어지면, 주어진 문자로 만들수 있는 모든 문자배열 조합을 프린트 하시오.

Given a string, print all permutations of characters in the string.

input: ABC

output: ABC ACB BAC BCA CBA CAB

풀이

Permutation 혹은 모든 조합을 구하는 문제는 주로 재귀함수로 풀 수 있습니다.

이 문제는 각 자리 마다 가능한 모든 문자를 넣고 재귀함수를 돌려주면 됩니다.

ABC로 예를 들자면,

"A", BC -> A를 고른뒤 BC로 재귀함수 콜.

-- "AB", C -> B를 고른뒤 C로 재귀함수 콜.

---- "ABC" -> 남은 문자가 없을때 프린트.

-- "AC", B -> C를 고른뒤 B로 재귀함수 콜.

---- "ACB" -> 남은 문자가 없을때 프린트.

"B", AC -> B를 고른뒤 AC로 재귀함수 콜.

-- 위와 같이 반복

"C", BA -> C를 고른뒤 BA로 재귀함수 콜.

-- 위와 같이 반복

확실히 고른 문자들과 고르지 않은 문자들을 구별할 정수(l)를 지정해줍니다.

(l)인덱스 전의 문자는 바꾸지 않습니다.

문자를 추가한뒤 재귀함수를 콜할때 이 정수(l)를 +1 해줍니다.

각 문자를 바꾸려면 substring(str, l, r) 사이에서 l 포지션의 문자를 l 부터 r 까지의 문자와 바꾸어 주며 재귀함수를 콜해줍니다.

이때 중요한건 재귀함수가 끝나면 바꾸었던 문자를 되돌려놓아야합니다.

그래야 재귀함수 안의 함수가 끝났을때 이어서 다른 재귀함수를 제대로된 문자열로 콜할수 있습니다.

```
void main() {
  String str = "ABC";
  solve(str, 0, str.length() - 1);
}

void solve(String str, int l, int r) {
  if (l == r) {
    System.out.println(str);
  }
  else {
    for (int i = l; i <= r; i++) {
      str = swap(str, l, i);
      solve(str, l + 1, r);
      str = swap(str, l, i);
    }
  }
}

String swap(String str, int i, int j) {
  StringBuilder sb = new StringBuilder(str);
  sb.setCharAt(i, str.charAt(j));
  sb.setCharAt(j, str.charAt(i));
  return sb.toString();
}
```

시간복잡도: O(n!)

n 개의 문자열로 만들수 있는 모든 가지의 수는 n!