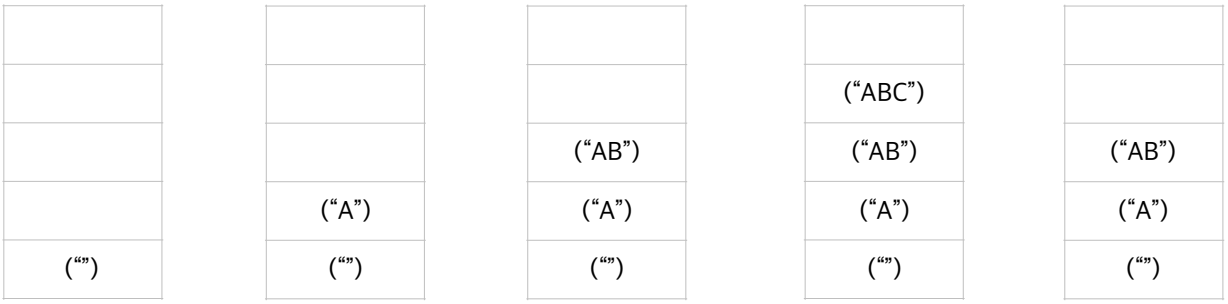
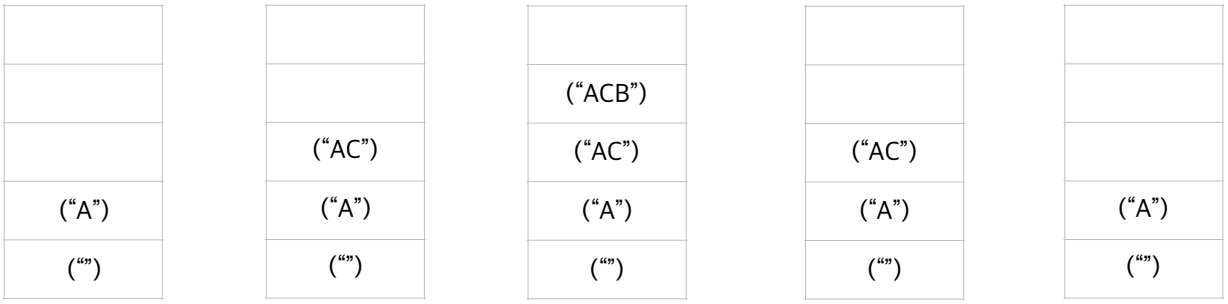
Backtracking

이번에 배울 내용은 **Backtracking(백트래킹)** 입니다. Backtracking 은 **재귀함수**로 구현되며, **콜스택**의 개념이 유용하게 사용됩니다. Backtracking 의 용도는 다음과 같습니다. 모든 경우를 조사하는 **전수조사, 순열이나 조합** 등이 그 예가 되겠습니다. 백트래킹의 콜스택은 다음과 같이 쌓이게 됩니다. A, B, C 문자로 만들 수 있는 모든 문자 순열을 구하는 Backtracking Code 가 있는 것을 가정하고 그려보겠습니다.





위 그림이 콜스택의 움직임입니다. 빈 문자열로 시작하여 A가 쌓이고, AB가 그 위에 쌓이고 ABC가 쌓이고, 그러다가 더 쌓을 수 없으니 콜스택을 비우면서 다음 프로세스를 시작합니다. 이렇게 콜스택을 비우는 과정이 퇴각하는 것과 비슷하다 하여 Backtracking 이라는 이름이 붙은것 같습니다.

자 이제, ABC 문자열로 순열을 만드는 psedo code를 봐봅시다.

```
N = length("ABC")
string S = ""

Backtracking(len):
    if len equal N:
        print S
        return
    for c in "ABC"
        if using c ?
        c is used.
        S.append(c)
        Backtracking(len + 1)
        S.erase(last_append)
        c is not used
```

이 코드에서 보시면 아시겠지만 S에 문자를 붙인 다음 호출된 함수에서 탈출한다면 다시 맨 뒤에 문자를 지워줌을 알 수 있습니다. 동시에 c를 사용하지 않았다고 알려주는데요, 이것을 통하여 순열을 출력함과 동시에 나중에 다시 사용하지 않음을 알릴 수 있습니다.

이제 조합에 대해서 설명할텐데요, 조합은 순열과 완전히 동일하나 사용한 위치를 기억해서 그 이전것을 사용하지 못하게 막는 것이 핵심입니다. 아래의 psedo code를 봐주세요.

```
N = length(“ABC”)
string S = “”

Backtracking(len, pos):
    if len equal 2:
        print S
        return
    for c in range(pos, N)
        S.append(“ABC”[c])
        Backtracking(len + 1, c + 1)
        S.erase(last_append)
```

이 코드를 통하여 알 수 있는 것은 고른 지점에서 +1 한 지점을 다음 상태에서 고르게 한다는 점입니다. 이것을 통하여 우리는 조합과 순열을 모두 구현해 보았습니다. 백트래킹의 함수 맨 윗줄은 보통 실패 조건을 씁니다. 이 경우에 return을 하여 더이상 탐색하지 못하게 막죠. 실패 조건을 쓴 뒤에는 성공 조건을 씁니다. 성공조건을 쓰면 출력을 하거나 정답 목록에 더해주거나 하는 행위를 하고 return을 해주면 됩니다. 반드시 return을 하지 않아도 되는 문제도 있으니 주의하세요. 다만 보통의 경우에는 return 합니다. 마지막으로 for문은 보통 고르는 구간입니다. 함수 호출이 일어나는 구간이죠.

이러한 구조를 알고 있으면 일반적인 백트래킹을 구현하기는 매우 쉬워집니다.

백트래킹의 **Cutting(커팅)**에 대해 알아봅시다. 커팅이란, 실패조건을 추가시키거나 탐색 도중에 탐색하지 않아도 되는 구간을 미리 발견하여 탐색 공간을 줄임으로서 퍼포먼스를 올리는 기법입니다. N-Queen 문제에서 대각선, 열 등을 처리하여 만약 다른 퀸이 해당 대각선이나 열을 점거하고 있으면 continue등을 사용하여 방문하지 못하도록 막는것이 그 예시가 되겠습니다.