알고리즘 숙제 #3: DP 알고리즘

201901621 공진성

1.

```
Matrix
0 0 0 0 0
0 0 1 0 1
0 1 0 1 0
1 0 1 0 0
0 0 0 0 0

DP Matrix
1 1 1 1 1
1 1 1 1 1
1 1 2 2 2
1 2 2 3 1
1 1 1 1 1
M[4,4]=3
```

우선 주어진 코드를 이용하여 랜덤한 배열 생성 및 출력함수를 사용하였습니다. 전체적인 작동 방식은 랜덤하게 생성된배열과 같은 크기의 DP 배열을 생성합니다. 그 후, 아래 후술할 4개의 점인지를 판단하는 것만으로 가장 큰 정사각형의 크기를 판단하였습니다. 4개의 점은 과제에 설명되어있는 아래 오른쪽 코너 A[I,j]을 포함해 윗칸, 왼쪽칸, 왼쪽 윗칸을 확인하였으며, 아래 표와 같은 2 * 2의 0과 1이 반복되는 정사각형인지 아닌지를 판단합니다.

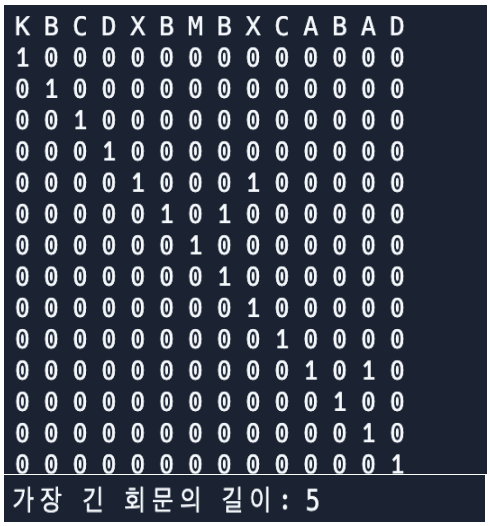
0	1
1	0

1	0
0	1

왼쪽 끝(1행)이거나 첫번째 줄(1열)일 경우는 비교할 4개의 점이 존재하지않아, 그 위치에서 가장 큰 정사각형의 길이는 1이 됩니다. 이 값을 수동으로 기입한 후, I = 1, j = 1인 지점부터 반복문이 진행됩니다. 4개의 점을 판단할 수 있는 지점부터는 4개의 점이 만약 위 표와 같은 모양새가 아니라면, 그 자리에서는 1을 기입하였습니다.

만약 위 표와 같은 모양새라면 그 점에서 가장 큰 정사각형의 크기는 2거나, 그 이상입니다. 여기서 DP알고리즘의 특성을 활용하여 구현하였습니다. 만약 위 표와 같은 모양새인데, A[I,j]점을 기준으로 나머지 3개의 점이 1, 2, 2라면 거기서의 A[I,j]점도 2일 것입니다. 한쪽 점에서는 0과 1이 반복되지않았으므로, 더 큰 정사각형으로 확장 불가능합니다. 하지만, 나머지 3개의 점이 모두 2, 2, 2라면 위의 출력 예시와 같이 3이 기입됩니다. 3가지 점에서 이미 0과 1의 반복된 정사각형이 만들어지기 때문입니다. 이 방식을 활용하여 A[I,j]점에 3가지 점의 최소값 + 1을 기입해주었습니다. 이렇게 코드를 작성한 결과, 시간복잡도는 O(n^2)(n은 배열 한 변의 길이)로 확인했습니다.

2.



이해를 돕기위해 보고서 스크린샷엔 DP테이블(2차원 배열)을 같이 출력했습니다. 예시로 들은 문자열은
"KBCDXBMBXCABAD"로 과제에 주어진 예시 문자열을 기준으로 진행하였습니다.

작동방식은 우선, 한글자, 두글자, 세글자 이상의 회문일때를 각각 분류하여 DP 테이블에 넣는 방식으로 진행했습니다. 위의 스크린샷을 확인해보면 좌측상단부터 우측 하단까지 대각선으로 1이 되어있는데, 이는 1글자 회문들을 전부 처리해준 것입니다.
여기서의 1글자 회문이란, ABCDEABCDE 같이 2글자 이상의 회문이 전혀 생성되지않는 경우를 말합니다. 1글자 회문을 처리해줘야 3글자 회문에서 1글자회문이 중간에 들어있을때, 위의 예시에선 ABA같은 회문을 받아올 수 있습니다.

추가로, 만약 ABCBBA라는 문자열이 들어왔을때, BB라는 두글자(같은 글자가 두번) 나오는 경우를 처리해준 뒤, 마지막 3글자 이상의 회문일 경우까지 진행하였습니다.

앞서 기록한 1글자, 2글자 회문들을 다 기록해두었기 때문에, 문자열 전체를 검색하며 하는 방식으로 진행하지않았습니다. start, end, length변수를 사용하여 문자열 길이만큼 반복문을 진행하는데, 만약 start변수와 length를 이용하여 구한 end변수의 문자가 같고, 그 사이의 문자가 회문이라면 결국 현재 찾은 split문자열도 회문이라는 점을 이용하여 가장 긴 회문의 길이를 갱신하며 풀어나갔습니다.