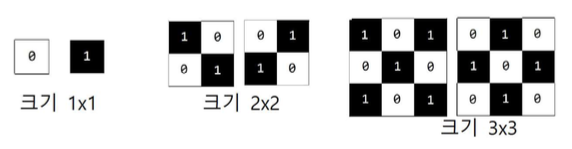
**알고리즘 숙제 #3 - 201901551 컴퓨터공학부 김정목**

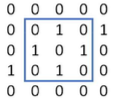
**1. 문제**

원소가 0 또는 1 인 nxn 2차원 배열 A가 주어질 때, 0과 1이 교대로 나타나는 가장 큰 정사각형의 크기를 찾는 동적 계획 알고리즘을 이용한 **1) c 프로그램**을 작성하고, **2) 시간복잡도** 와 **3) 작동 방법**을 자세하게 설명하시오.

\* 다음 그림은 크기가 각각 1x1, 2x2, 3x3 인 0과 1이 교대로 나타나는 정사각형이다.



\* 예를 들어 다음의 주어진 5x5 배열(A)에서 0과 1이 교대로 나타나는 가장 큰 정사각형의 크기는 각각 3x3 이다.



\*조건) 알고리즘 내 배열 M[i,j]만 생성한다. M[i,j]는 A[i, j] (인덱스 i,j는 1부터) 를 정사각형의 맨 아래 오른쪽 코너에 위치하면서 0 과 1이 교대로 나타는 정사각형의 한 변의 크기를 저장한다. 위 문제에 주어진 배열에서 최종 output은 M[4,4] = 3을 출력해야 한다.

\* 함수 원형은 int nxn(A) 이며, return: 가장 큰 정사각형의 길이, A: nxn 배열

**2. 시간 복잡도**

|  |
| --- |
| 위 문제를 해결한 알고리즘의 시간 복잡도는 O(n^2) 이다. 여기서 n은 배열 A의 행, 열의 길이를 뜻한다. 알고리즘은 A의 각 위치에 대해서 한 번 씩만 계산을 수행한다. 각 위치에서 수행되는 연산은 상수 시간으로 처리되며 총 전체는 n(행) x n(열) 번 연산이 수행된다. 따라서 이 알고리즘은 2차원 배열의 크기에 선형적으로 비례하는 시간 복잡도를 가지며, n이 커질수록 필요한 시간은 비례하여 증가한다. |

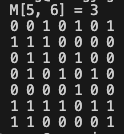
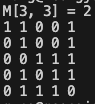
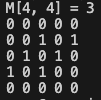
**3. 작동 방법**

|  |  |
| --- | --- |
| **주요 함수** | **설명** |
| int min(int a, int b, int c) | 세 정수 중에서 최솟값을 반환하는 함수 |
| int nxn(int\*\* A) | 행렬 A 안에서 조건에 맞는 가장 큰 정사각형의 길이를 찾는 함수 |

|  |  |
| --- | --- |
| **주요 변수** | **설명** |
| #define N | 행렬의 길이 |
| int M[N+1][N+1] | DP를 사용하여 현 위치에서의 가장 큰 정사각형의 길이를 저장할 2차원 배열 |
| int maxSize | 가장 큰 정사각형의 길이 |
| int maxI, maxJ | 가장 큰 정사각형의 우측 하단 좌표 |
| int\*\* A | N x N 행렬 A |

|  |
| --- |
| 위 문제는 n x n 크기의 2차원 배열 A 내에서 0과 1이 교대로 나타나는 가장 큰 정사각형을 찾는 동적 계획법 알고리즘을 구현하는 문제이다. 첨부된 코드는 동적 계획법의 특성을 활용하여 이중 반복문을 돌리면서 A[i][j]를 우측 하단으로 할 때 나오는 정사각형의 최대 길이를 M 배열에 저장한다.  첨부된 코드는 아래와 같은 단계로 알고리즘이 진행된다.  1. makeMatrix 함수를 통해 무작위로 동적 배열(행렬) A를 만든다. 2. M 배열은 0과 1이 나오는 정사각형의 최대 길이를 저장하는 배열이다. 배열의 첫 행과 첫 열은 경계 조건으로 해당 위치에서는 정사각형의 최대 길이가 1이 될 수 밖에 없으므로 M 배열의 첫 행과 첫 열은 1로 초기화한다. 3. A[i][j]를 기준으로 왼쪽 위 대각선과는 같고 왼쪽, 위쪽과는 다르다면 즉, A[i][j] = A[i-1][j-1], A[i][j] != A[i-1][j], A[i][j] = A[i][j-1] 조건이 만족한다면 정사각형이 형성이 됨으로 M[i-1][j-1], M[i-1][j-1], M[i][j-1] 중 최소값에 1을 더한 값을 M[i][j]에 갱신한다. 최소값으로 갱신하는 이유는 정사각형이라는 조건을 만족시키기 위해서다. 4. M[i][j] 값을 계산할 때마다, 해당 값이 현재까지 발견된 정사각형의 최대 길이보다 크다면 정사각형의 최대 길이를 M[i][j]로 갱신한다. 5. 최종적으로 가장 큰 정사각형의 위치와 길이, 생성된 행렬을 출력하고 할당된 메모리를 해제한다. 6. 여기서 문제 조건에 맞게 i와 j를 1부터 시작해야함으로 조건에 맞게 코드의 인덱스 값들을 조정하였다. |

**4. 출력 화면**



**1. 문제**

주어진 스트링 S=(𝑠!,𝑠",𝑠#, ... ,𝑠$) 의 부분 스트링 중에서 가장 긴 연속적인 회문의 길이를 계산하는 동적계획 알고리즘을 사용한 **1) c프로그램**을 작성하고, **2) 시간복잡도**와, **3) 작동방법**을 자세하게 설명하시오.

\* 예를 들어, 'K B C D X B M B X C A B A D' 에서는 XBMBX 가 가장 긴 연속된 회문이며 그 길이는 5 이다.  
\* 함수 원형은 int palindrome(s, i, j) 이며, return: 회문의 길이, s: 문자열, i: 문자열 시작 index, j: 문자열 마지막 index

**2. 시간 복잡도**

|  |
| --- |
| 위 문제를 해결한 알고리즘의 시간 복잡도는 O(n^2) 이다. 여기서 n은 문자열의 길이를 뜻한다. 위 알고리즘은 모든 가능한 부분 문자열에 대해 회문 여부를 확인하기 때문에 O(n^2)이다. 단일 문자와 연속된 두 문자에 대한 회문 검사는 상수 시간이 걸리지만, 길이가 3 이상인 부분 문자열에 대한 회문 검사는 이중 반복문을 통해 검사를 해야함으로 전체 시간 복잡도는 문자열 길이의 제곱에 비례한다. |

**3. 작동 방법**

|  |  |
| --- | --- |
| **주요 함수** | **설명** |
| palindrome(char\* s, int i, int j) | 주어진 문자열에서 가장 긴 연속된 회문을 찾는 함수 |

|  |  |
| --- | --- |
| **주요 변수** | **설명** |
| char save\_palindrome[100] | 가장 긴 회문을 저장할 충분히 큰 배열 |
| char\* str | 입력된 문자열 |
| int dp[n][n] | 동적 계획법을 위한 배열 |
| int maxlen | 회문의 최대 길이를 저장할 변수 |
| int start | 가장 긴 회문의 시작 인덱스 |

|  |
| --- |
| 위 문제는 n 길이를 가진 문자열에 대해서 연속되는 회문을 찾는 동적 계획법 알고리즘을 구현하는 문제이다. 동적 계획 배열인 dp[n][n] 배열을 사용하여 부분 문자열이 회문인지 여부를 저장한다.  첨부된 코드는 아래와 같은 단계로 알고리즘이 진행된다. 1. makeString 함수를 통해 무작위로 동적 배열 str을 만든다.  2. 동적 계획법 배열 초기화  - dp[n][n] 배열은 문자열의 각 부분 문자열이 회문인지 여부를 저장하는 데 사용된다.  - dp[x][y]는 문자열의 x번째부터 y번째 문자가 회문인지 나타낸다. 배열은 모든 값이 0으로 초기화 된다. 3. 단일 문자 회문 처리  - 모든 단일 문자는 자체적으로 회문이다. 따라서 dp[x][x]는 항상 1로 설정된다.  - 이 단계에서는 각 x에 대해 dp[x][x] = 1로 설정한다. 4. 인접한 두 문자의 회문 검사  - 두 인접한 문자가 동일한 경우, 이들은 길이가 2인 회문을 형성한다.  - 문자열을 순회하며, str[x] == str[x + 1]인 경우 dp[x][x + 1] = 1로 설정한다. 이 때 최대 회문 길이(maxlen)와 시작 인덱스(start)도 갱신된다. 5. 길이가 3 이상인 부분 문자열 검사   - 이 단계에서는 길이가 3 이상인 모든 부분 무자열을 검사한다. 부분 문자열 str[x...y]가 회문인지 확인하기 위해, str[x] == str[y] 이고, dp[x + 1][y - 1] 이 1인지 확인하다.   - 이중 루프를 사용하여 모든 가능한 x와 y에 대해 검사를 수행한다. 회문이 확인되면 dp[x][y] == 1로 설정하고 maxlen과 start를 갱신하다. 6. 최대 길이 회문 갱신 및 저장  - 이 알고리즘은 가장 긴 회문을 찾고 저장하는 것이다. 각 단계에서 회문을 찾을 때마다 maxlen과 start가 갱신된다.  - maxlen은 가장 긴 회문의 길이를, start는 해당 회문의 시작 인덱스를 나타낸다. 회문이 발견되면 strncpy를 사용하여 save\_palindrome에 해당 회문을 복사하고, 문자열의 마지막을 나타내는 널을 추가한다.  7. 최종적으로 입력된 문자열 str, 저장된 회문인 save\_palindrome, 회문의 길이 maxlen을 출력하며 할당된 메모리를 해제한다. |

**4. 출력 화면**





