알고리즘 과제 4. 연속 행렬 곱셈

제출일 : 2017/04/15

경영학과 201101328　박성환

목차 :

1. 문제 정의

-> 연속으로 곱해지는 행렬들의 크기를 입력 받아 그 중 계산횟수의 최소값을 구한다.
동적 계획법 방법을 사용한다.

2. 해결방안

-> 동적 계획법의 특징대로 부분 문제부터 해결해서 원하는 입력값까지 확장해서 문제를 해결한다. 가장 작은 단위인 행렬 1개는 곱셈 횟수 0으로 부분해결하고 차례대로 2개일 때, 3개일 때, … , n개 일 때까지 순서대로 해결하는 것이다.

알고리즘의 핵심 부분은 다음과 같다.

```
        for(j = 1; j <= size; j++)
            matrix[i][j] = 0;
        // Initilize matrix with 0.
    }

    for(mul_depth = 1; mul_depth < size; mul_depth++) {
        // mul_depth is multiplication depth.
        // If it is 1, cals are like (10x20), (20x30)
        // if it is 2, cals are like (10x20x30), (10x20x10)

        for(i = 1; i <= size - mul_depth; i++) {
            j = i + mul_depth;
            matrix[i][j] = 100000000l;

            for(k = i; k < j; k++) {
                temp = dims[i-1] * dims[k] * dims[j] + matrix[i][k] + matrix[k+1][j];
                if(temp < matrix[i][j]) {
                    matrix[i][j] = temp;
                }
            }
        }
}
```

-> 행렬 1개일 때는 곱셈이 없기 때문에 matrix[1][1], matrix[2][2], …, matrix[n][n]까지 0으로 부분해결되었다. 이제 그 다음부터 행렬 2개일 때는 본격적인 계산에 돌입하는데 어떤 부분 행렬의 곱셉의 횟수는 '피연산 행렬들이 계산되는데 소요되었던 각각의 계산 횟수 + 두 행렬의 곱셈 횟수'가 된다. 그래서 matrix[i][j](i번째부터 j번째까지의 행렬의 최소 곱셈횟수)는 매우 큰 값을 정의해 놓은 뒤, k값이 움직임에 따라 산출되는 최소 값으로 치환한다. 여기서 나온 행렬 계산횟수는 그 다음의 더 큰 차원의 부분문제 사용에 활용된다. 결국에는 가장 큰 행렬인 입력된 행렬의 계산횟수를 구할 수 있다.

3. 테스트

-> 과제 요구대로 정상적으로 출력된다.

Chained Matrix Multiplications 201101328 박성환

# of Matrices : 4

Enter d0 d1 ... d4 :
10 20 5 15 30
Problem is [10:20]x[20:5]x[5:15]x[15:30]

```
  C   [ 1 ]  [ 2 ] [ 3 ] [ 4 ]
[ 1 ]     0  1000  1750  4750
[ 2 ]     0     0  1500  5250
[ 3 ]     0     0     0  2250
[ 4 ]     0     0     0     0
```

Final solution is 4750

4. 결론

-> 동적계획법을 통해 행렬의 최소 계산횟수를 구할 수 있었다.

일단 배열을 쓴 점이 매우 흥미로웠다. 맨 처음 이 문제를 접하고 부분문제를 어떻게 만들어야 할지를 잘 몰랐다. 각 부분문제에 맞게 노드나 구조체를 따로 선언해서 연결해야 하는가 등의 고민을 했는데 예제에서 배열로 깔끔하게 처리해서 인상깊었다.
도메인 지식에 대한 필요를 많이 느꼈다. 알고리즘 자체는 차치하고 각 부분 문제의 값을 구할 때 행렬에 대한 기본지식이 있어야 문제에 접근할 수 있다는 것을 알았다.
그리고 이번 과제에서는 사실 테스트 코드 작성이 과제 전체 시간의 80% 이상을 차지했다. 언어에 대한 지식이 얕아 과제에 필요한 함수에 대해 공부하고, 디버깅하는 데에 너무나 많은 시간이 소모되어 힘이 들었다. 지금까지의 과제들을 리뷰하면서 C에 대해 더 익숙해져야 할 것 같다.