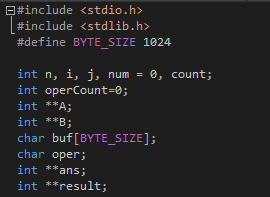
2014312516 김형준

김형식 교수님

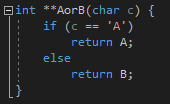
자료구조개론

Assignment1



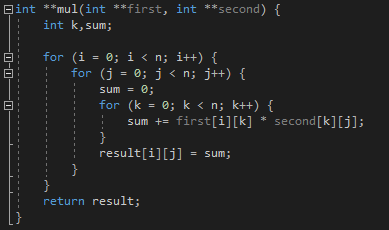
우선 stdio.h 와 stdlib.h 헤더파일을 사용하였다. stdio.h 는 파일입출력 함수를 사용하기 위함이고, stdlib.h는 malloc을 통해 동적 배열을 만들어주기 위함이다. BYTE\_SIZE는 후에 연산행을 입력받기 위한 char 배열에 쓰일 것인데, m(연산의 수)가 최대 500이기에 연산되는 matrix명까지 생각하여 넉넉하게 2배수 근처인 1024로 설정해주었다.

전역변수로는 main함수를 제외한 함수들에 들어가야 하는 변수들을 설정해주었다. 이는 main함수를 제외한 다른 함수에서 값을 할당하고 유지하기 위함이다. 동적배열(matrix)은 A,B, ans, result 총 4개로 설정하였다. 처음엔 result를 제외하고 3개를 사용해 코드를 짜보았지만, 함수에서 결과값을 반환하기위해, mul(\*)함수에서 값이 덮어 쓰이지 않고 계산이 잘 되도록 하기 위해 추가적인 result배열을 선언해주었다. 따라서 계산함수의 결과값은 result로 반환하고 main함수에서 ans, 출력할 matrix에 이 값을 전달하는 식으로 코드를 진행하였다.



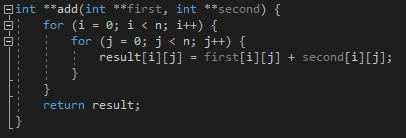
이 함수는 input의 맨 아래 줄에서 연산 문자열을 받을 때에, 가장 처음으로 오는 matrix가 A인가, B인가를 따져주어 반환해주는 함수이다. 맨 처음 ans배열에 A를 넣을 지, B를 넣을 지 결정하기 위해 사용하였다. 또한 연산행에서 연산 기호 다음에 나오는 문자가 A인지 B인지 파악할 떄에도 사용하였다.

Performance analysis : 1



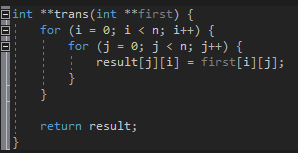
인수로 받은 두 matrix를 곱하는 함수이다. 일반적으로 행렬의 곱셈을 하는 방식으로 코드를 진행하였다. Result를 사용하기 전에 result자리에 ans를 그대로 사용했었는데, 값이 할당되면서 결과값이 달라지는 버그가 발생하여 새로운 배열을 사용하게 되었다.

Performance analysis : O(n^3)



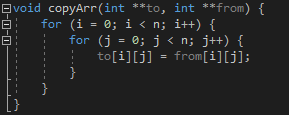
인수로 받은 두 matrix를 더하는 함수이다. 같은 자리의 원소끼리 더하는 방법을 사용하였다.

Performance analysis : O(n^2)



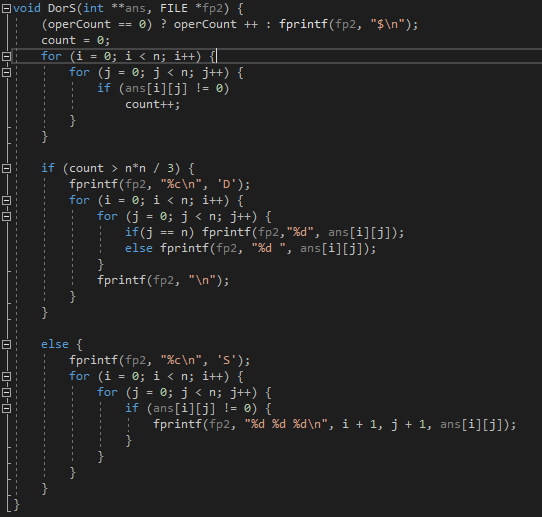
인수로 받은 하나의 matrix의 row와 column의 원소를 바꾸는 방법을 사용하였다.

Performance analysis : O(n^2)



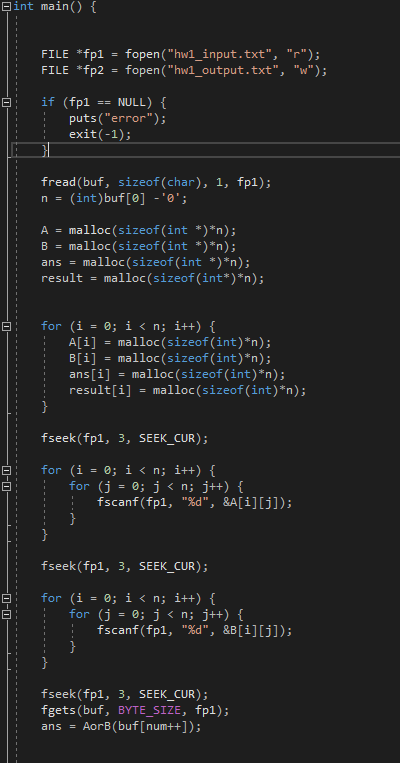
인수로 받은 두 matrix를 그대로 복사해주는 함수이다. 위의 계산함수에서 반환된 result를 출력할 ans로 옮기기 위해 사용하였다. 이 방법이 비효율적이라고 생각할 수도 있지만, 계산의 결과가 하나의 다른 matrix에 저장이 되어야 다음 계산을 할 수 있기 때문에 계산 후 저장해주는 방법을 사용하였다.

Performance analysis : O(n^2)



Dense 인지 Sparse인지 결정하는 함수이다. 이 함수에서 파일 출력이 이루어진다. 가장 먼저 연산을 구별하는 $를 출력한다. 맨 첫 연산에서는 출력하지 않는데 이를 구별하기 위해 operCount변수를 사용해주었다. 그 후 원소의 개수를 세어 dense인지 sparse인지 결정한다. dense라면 D를 먼저 출력한 후에 matrix 형식으로 원소를 전부 출력해준다. sparse라면 S를 출력해준 후에 sparse matrix 형식으로 non-zero element만 출력해준다. fprintf 시에 조건을 지정해준 이유는 행의 마지막 원소일 경우에는 공백을 출력하지 않기 위함이다.

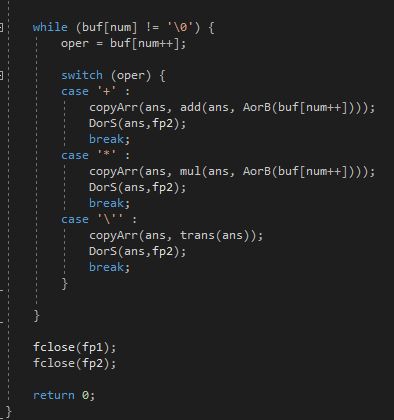
Performance analysis : O(n^2)



main함수이다. 파일포인터를 이용해 input 파일과 output 파일을 읽기모드와 쓰기모드로 설정한다.

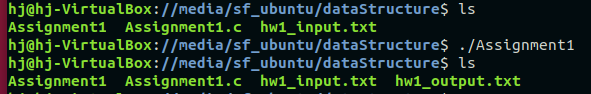
읽을 파일이 없다면 에러를 출력하고 -1을 반환한다. 입력된 n의 크기만큼의 동적배열 4개에 메모리를 할당해준다. 이는 A matrix, B matrix, 출력 matrix, 연산 반환matrix에 사용된다. fseek 함수는 줄이 구별되는 $를 건너뛰기 위함이다. A와 B에 원소를 전부 입력 받아준 후에 마지막 연산행을 문자열로 char 배열 buf에 입력받는다.

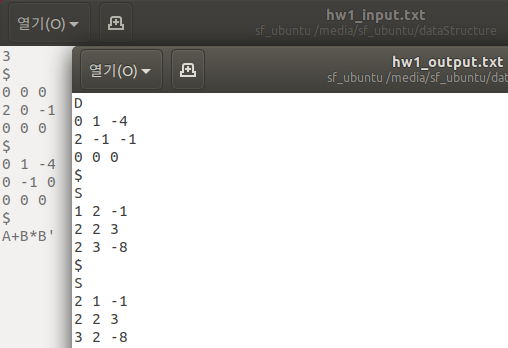
Performance analysis : O(n^2)



buf에서 한칸 씩 이동하면서 연산기호와, A와 B중 어떤 것이 연산에 사용되는지를 파악한다. 연산기호는 switch문을 사용해 구별하였고, AorB함수를 사용해 matrix를 구분하였다. buf 값이 파일 끝에 도달하면 while문은 종료된다. 파일포인터를 닫아준 후에 main함수가 종료된다.

Linux 에서 실행 화면 :





Whole Performance analysis : O(n^3)

여기서 n은 최댓값이 1000이기에 대략 1sec 정도가 걸리게 된다. 만약 O(n^2)로 줄일 수 있다면 1millisec으로 확연히 시간이 단축되기에 sparse matrix연산 시에 element를 전부 계산하지 않고, 필요한 것만 계산할 수 있는 방법을 생각한다면 더욱 효율적인 프로그램이 될 것이다. 이번 과제를 통해 기본적인 파일입출력, 연산기호에 따른 함수 처리, 동적배열의 사용법을 익히게 되었다. 이번엔 간단한 matrix의 자료구조였지만, 앞으로 배울 수많은 자료구조들의 기본개념을 익힐 수 있는 시간이었다.

