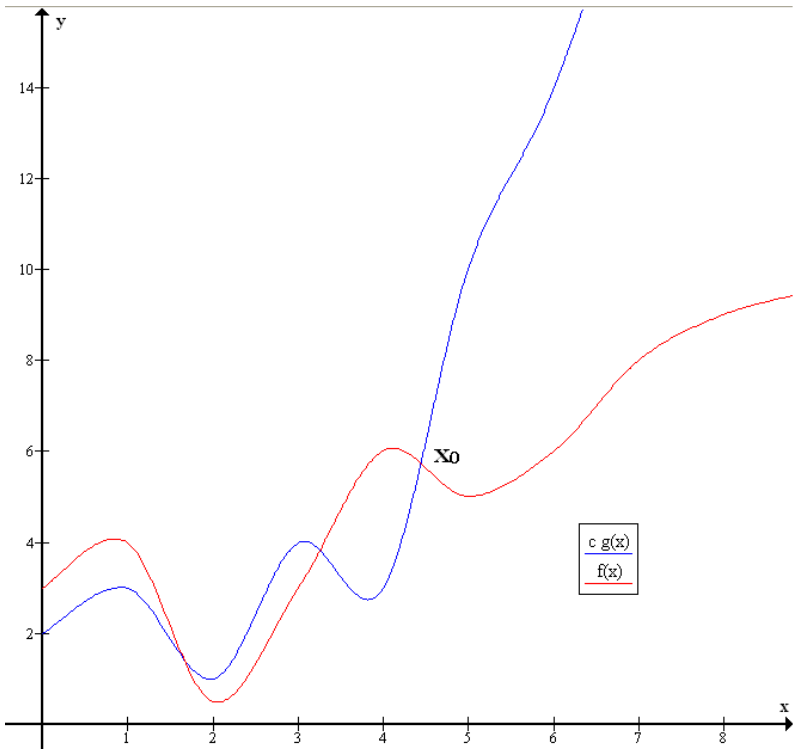
알고리즘 개요

복잡도

- 1억 번의 연산 = 약 1초
  - 생각한 풀이 방법이 제한 시간을 만족하는지 확인하는 방법으로 활용.
- 알려진 알고리즘의 경우 점근 표기법을 활용한 복잡도를 표시.
  - 예) Big O (대문자 O) 표기법
    - n개의 입력에 대한 연산량을 계산한 식에서 가장 큰 차수만으로 표시.
    - '최악의 경우 이만큼 연산이 필요함'으로 해석하면 된다.
    - $O(\log n)$, $O(n)$, $O(n^2)$, $O(2^n)$,

- Big O (상한 점근)



Example of Big O notation:
$f(x) \in O(g(x))$ as there exists $c > 0$ (e.g., $c = 1$) and $x0$ (e.g., $x0 = 5$) such that $f(x) \leq cg(x)$ whenever $x \geq x0$.
(출처 : Wikipedia Big O notation)

복습

- 정수형
  - 연산결과는 64-bit 정수형이 필요한 경우가 있음.
    - long long (C언어), long (Java)
  - 64-bit를 넘어가는 정수형을 다루는 경우 수식으로 처리해야 함.
  - C의 경우 scanf()의 서식문자로 입력을 한 자리 씩 읽을 수 있음.
    - 예) %1d, %1x
  - for문에는 int형 사용.
  - 대략 10자리 숫자.

- 자료형
  - 실수형
    - double형 사용 (float은 표현할 수 있는 범위가 작음).
    - 고정형 sign bit를 사용하므로 unsinged를 붙여도 자리수가 늘어나지 않음.
    - 출력에 서식문자를 사용하면 마지막 자리 뒤에서 반올림 됨.
      - 예) %.3f

■ 배열

- 배열의 크기에 주의해야 함.
  - 인덱스를 0번부터 사용하는 경우와 1번부터 사용하는 경우.
- 인덱스 연산은 범위를 미리 정의하고 코딩을 시작해야 함.
  - 배열의 끝 부분을 읽지 않는 경우가 많으므로 주의.
- C언어의 경우 큰 배열은 전역으로 선언한다.
  - 스택 영역의 크기가 제한되어 있기 때문
- 배열의 최대 크기.
  - 정수형 백만개를 저장하는 1차원 배열(혹은 1000x1000 2차원 배열).
  - 배열에 저장할 수 없는 경우 수식을 알아내야 함.

프로그래밍 도구

- C/C++
  - Visual Studio 2013 Express 이상 버전
  - Dev-C++ 최신버전
  - ✓ Linux 기반의 채점사이트 서버에서는 gcc 사용.
  - ✓ C 문법으로 작성했더라도 .cpp로 작성한 경우, 서버 제출시 사용언어를 C++로 선택해야 함.
- Java
  - Eclipse + JDK

연습

- 크기가 NxN인 2차원 배열에 다음과 같은 모양으로 숫자가 저장된다.
- 크기N과 행, 열이 주어지면 해당 위치에 저장된 숫자를 출력하시오.

	0	1	2	3	4	5	열
0		1	6	10	13	15	
1			2	7	11	14	
2				3	8	12	
3					4	9	
4						5	
5							
행							

입력
1 // 테스트케이스 개수 6 2 4// N 행 열
출력
#1 8

■ 접근 방법

인덱스로 표시한 접근 순서

1	2	3	4	5	→행과 열의 차이 I
0, 1 (1)	0, 2 (6)	0, 3 (10)	0, 4 (13)	0, 5 (15)	
1, 2 (2)	1, 3 (7)	1, 4 (11)	1, 5 (14)		
2, 3 (3)	2, 4 (8)	2, 5 (12)			
3, 4 (4)	3, 5 (9)				
4, 5 (5)					
4	3	2	1	0	마지막 행 번호 N-I

```
k = 1;
for (I = 1; I <= N; I++)   // 행과 열의 차이
{
   for (i = 0; i <= N − I; j++)  // 행 번호
   {
      j = i +I ;  // 열 번호
      arr[i][j] = k++;  // 열 번호 j+i
   }
}
```

연습

- 0번에서 출발해 N번(종점) 정류장까지 전기버스를 운행하려고 한다. 한 번 충전으로 최대한 이동할 수 있는 정류장 수 K와 종점 번호 N, 충전기가 설치된 정류장 수 M, M개의 정류장 번호가 주어지면 최소한 몇 번의 충전을 해야 종점에 도착할 수 있는지 출력하라.

입력
1 => 노선 수
3 10 5 => K, N, M
1 3 5 7 9 => M개의 정류장 번호

출력
#1 3

정류장	0	1	2	3	4	5	6	7	8	9	10
충전기		O		O		O		O		O	
충전				1		2		3			

■ 접근 방법

- 이전 정류장과의 거리가 K보다 큰 경우 -〉 도착 불가
- 이동가능한 경우,
  - s[i]정류장과 마지막 충전 정류장과의 거리가 K보다 크면 i-1에서 충전.
  - 마지막 충전 정류장을 i-1로 변경
  - 종점까지 반복

i	0	1	2	3	4	5	6
s	0	1	3	5	7	9	10
last	↑		↑	↑	↑		
charge			1	2	3		