

ICE3020 알고리즘설계

<Time Complexity>

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2022년 03월 16일

학부 정보통신공학

학년 3

성명 이동건

학번 12181879



1. 개요

* 아래와 같은 프로그램 작성
  + (Cubic complexity)를 만족하는 3중 for루프 곱(2차 행렬 a, b, c 곱셈)
  + (Exponential complexity)를 만족하는 피보나치 수열 알고리즘 또는 임의의 알고리즘
* 입력
  + Cubic complextiy 알고리즘의 경우 n=10, 50, 100, 150, 200 을 사용하여 결과 출력
  + Exponential complexity 알고리즘의 경우 n=10, 20, 30 을 사용하여 결과 출력
* 출력
  + n^3 (Cubic complexity) 소요 시간 = 1.232222 (나노sec, 마이크로sec..)
  + 2^n (Exponential complexity) 소요 시간 = 25.232 (나노sec, 마이크로sec..)

1. 상세 설계 내용

* Cubic complexity

텍스트이(가) 표시된 사진

자동 생성된 설명

matrixmult 함수는 3개의 행렬 a, b, c의 곱셈을 수행하는 함수이다. 본 알고리즘은 3중 for loop를 돌면서 곱셈을 수행하므로, 알고리즘의 time complexity는 이다.

먼저, 계산을 수행하기 전, temp와 res라는 (0으로 채워진) 빈 행렬을 만들었다. 행렬 temp는 행렬 b와 c의 곱(b\*c)을 표현하는데에 쓰이고, 행렬 res는 행렬 a와 temp 즉, 행렬 a, b, c의 곱(a\*b\*c)을 표현하는데에 사용하였다. 빈 행렬을 만들기 위해, python의 numpy라는 라이브러리를 import하였고, numpy의 zeros 함수를 사용하여 빈 행렬을 만들었다.

10번 줄부터 13번 줄까지는 행렬의 곱셈의 원리대로 행렬 b와 c의 곱(b\*c)을 행렬 temp에 저장하는 코드이다. 마찬가지로, 15번 줄부터 18번 줄까지는 행렬 a와 행렬 temp의 곱(a\*temp)을 행렬 res에 저장하는 코드이다.

위에서 설명한대로, temp = b\*c 이고, 행렬 곱셉의 결합법칙에 따라 a\*b\*c = a\*(b\*c)이다. 이때, b\*c = temp이므로, a\*b\*c = a\*(b\*c) = a\*temp이다.

* Exponential complexity

텍스트이(가) 표시된 사진

자동 생성된 설명

fib 함수는 n을 인자로 받아 피보나치 수열의 n번째 수를 구하는 함수이다. 본 알고리즘은 실행이 될 때마다, 아래 **Figure 1**처럼 자기자신을 두 번 불러내는 행위 즉, 2번의 재귀호출을 수행하므로 알고리즘의 time complexity는 이다.

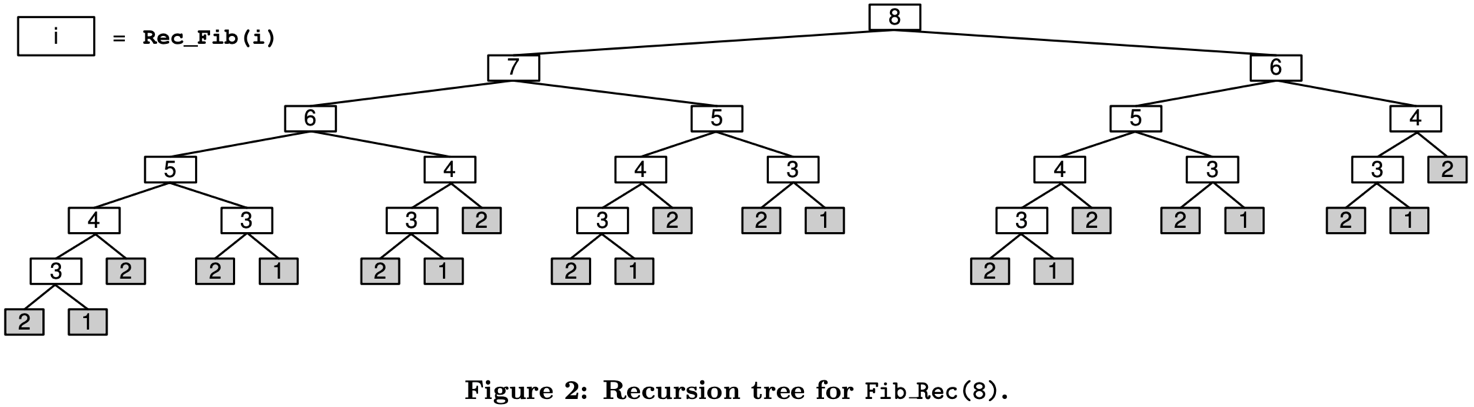


Figure 1 Recursion tree for fib(8) [1]

먼저, 함수의 base값으로서 n이 1일 때와 2일 때에 반환값을 1로 설정해주었다. 피보나치 수열에서 1번째 수와 2번째 수는 둘 다 1이기 때문이다.

만약 n이 1 혹은 2가 아니라면, fib(n-2) + fib(n-1) 값을 return하도록 설계하였다. fib(n-2) + fib(n-1) 값을 return하기 위해 프로그램은 n이 1 혹은 2가 될 때까지 재귀호출하고, n이 1 혹은 2가 되어 반복 실행을 멈추게 되면, 아래에서부터 순차적으로 올라오면서 (n-1) + (n-2)의 값을 구하고 반환하게 된다.

1. 실행 화면

* Cubic complexity

텍스트이(가) 표시된 사진

자동 생성된 설명

* Exponential complexity

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 결론

* Cubic complexity

실행 결과에서 n과 소요 시간의 관계를 보았을 때, 실행 시간이 에 비례하여 증가한다는 것을 확인할 수 있다. 예를 들어 n이 150일 때의 소요 시간이 약 14.887초이고, n이 200일 때의 소요 시간이 약 33.577초인데 에 비례식이 성립한다는 것을 확인할 수 있다.

* Exponential complexity

실행 결과를 보았을 때, n이 10씩 증가할 때마다, 소요 시간이 약 100배 정도 차이나는 것을 확인할 수 있다. 이를 통해, n이 50이라면 약 3300초(=55분)가 걸릴 것으로 예상할 수 있고, 소요 시간이 기하급수적으로(exponentially) 늘어난다는 것을 확인할 수 있다.

1. 참고문헌

[1] Rubio-Sánchez, M. and Pająk, B. “Fibonacci numbers using mutual recursion.” in Proceedings of the 5th Annual Finnish/Baltic Sea Conference on Computer Science Education, Koli Calling 2005 (Finland: TUCS General Publications), 174–177.