#03-1. 시간 복잡도

나정휘

https://justiceHui.github.io/

목차

알고리즘의 성능

알고리즘의 성능을 판단하는 척도

- 작업량: 얼마나 적은 연산을 필요로 하는가?
- 메모리 사용량: 얼마나 적은 공간을 사용하는가?
- 정확성: 얼마나 정확한 답을 구할 수 있는가?
- 최적성: 더 이상 개선할 여지가 없을 만큼 최적화가 잘 되었나?
- 단순성: 알고리즘의 작동 과정이 얼마나 단순한가?

알고리즘의 성능

작업량

- 알고리즘 문제를 풀 때는 작업(연산)의 횟수가 실행 시간에 가장 큰 영향을 미침
- 대충 어림 잡아서 1초에 연산 1억 번 정도
  - 메모리 접근 횟수와 방법도 중요하지만 지금은 생략...

- 실제 프로그램을 개발할 때는 연산 횟수 말고 다른 부분에서 시간이 더 오래 걸릴 수 있음
  - 네트워크
    - 광섬유에서의 빛의 속도는 진공에 비해 약 2/3 정도 (약 200'000'000m/s)
    - 서울과 뉴욕은 약 11000km 떨어져 있으므로 아무리 빨라도 55ms 이상의 딜레이가 발생할 수밖에 없음
  - 프로세스 스케줄링
    - 컴퓨터에서는 하나의 프로그램만 실행되는 것이 아님
    - 운영체제는 매 순간 어떤 프로세스를 처리할지 결정
  - 병목 현상
    - 프로그램이 여러 코어에서 병렬로 실행되는 경우
  - ...

알고리즘의 성능

메모리 사용량

- 요즘 나오는 문제는 대부분 메모리 제한을 여유롭게 주기 때문에 별로 신경 쓰지 않아도 됨
- 하지만 가끔씩 메모리 최적화를 요구하는 문제가 나옴
  - IOI ‘13 Day 1. Wombats (BOJ 8874)
  - IOI ‘13 Day 2. Game (BOJ 8876)
  - CEOI ’15 Day 1. Pipes (BOJ 10957)
  - KOI ‘18 고등부. 조화로운 행렬 (BOJ 15977)
  - PA 2018 Round 3. Palindrom (BOJ 26640)
  - PtzCamp Winter ’23 Day 7. Classical Data Structure Problem (BOJ 28204)

- 메모리 제한이 128MB보다 작으면 의심해 보자.
  - 요즘은 대부분 512MB ~ 1024MB 정도로 줌

알고리즘의 성능

정확성

- 이 세상에는 빠르게 풀 수 없음이 증명된 문제들이 있음
  - P ≠ NP이면 Satisfiability, Chromatic Number, Traveling Salesman Problem 등을 다항 시간에 풀 수 없음
    - 관심이 있다면... (링크)
  - 3-SUM Conjecture이 참이면 Collinear, Separator, Convolution3SUM 등을 $O(N^{2-\varepsilon})$보다 빠르게 풀 수 없음
    - 관심이 있다면... (링크)
  - APSP Conjecture이 참이면 2D maximum subarray 등을 $O(N^{3-\varepsilon})$보다 빠르게 풀 수 없음
    - 관심이 있다면... (링크)

- 이런 문제는 정확한 값을 구하는 것이 어렵기 때문에 적당히 좋은 답을 빠르게 찾는 시도를 많이 함

알고리즘의 성능

정확성

- Traveling Salesman Problem
  - N개의 도시와 도시 간의 거리가 주어짐
  - N개의 도시를 모두 방문하고 다시 시작점으로 돌아오는 최소 거리를 구하는 문제
  - 최소 가중치 해밀턴 회로를 찾는 문제
  - P ≠ NP이면 다항 시간에 해결할 수 없음
  - 단, $D(i, j) \leq D(i, k) + D(k, j)$를 만족하면 다항 시간에 최적해의 1.5배 이하인 해를 구할 수 있음
    - Christofides' algorithm
- Set Cover
  - 집합 X의 부분 집합 N개가 주어졌을 때, 합집합이 X가 되도록 최소 개수의 부분 집합을 선택하는 문제
  - P ≠ NP이면 다항 시간에 해결할 수 없음
  - 하지만 아주 간단한 전략을 이용해 최적해의 ln N배 이하인 해를 구할 수 있음

- TSP에서 1.5를 더 줄일 수 있을까?
- Set Cover에서 ln N 대신에 상수 배의 해를 구할 수 있을까?
- ...

알고리즘의 성능

최적성

- 문자열 S 안에 문자열 P가 등장하는지 확인하는 문제
  - 일단 S와 P를 한 번씩은 훑어야 하기 때문에 최소한 |S| + |P| 만큼의 연산이 필요함
  - 단순하게 구현하면 |S| * |P| 만큼의 연산이 필요함 → 최적이 아님
  - KMP 알고리즘을 사용하면 |S| + |P| 만큼의 연산이 필요함 → 더 좋아질 수 없으므로 이게 최적

- N개의 원소를 정렬하는 문제
  - 최소한 약 N log N번의 연산이 필요함을 증명할 수 있음
  - 간단한 알고리즘을 사용하면 N^2 만큼의 연산이 필요함 → 최적이 아님
  - 조금 더 복잡한 알고리즘(합병 정렬 등)을 사용하면 N log N 만큼의 연산이 필요함 → 더 빠르게 만들 수 없음

시간 복잡도

시간 복잡도

- 문제를 해결하는데 걸리는 시간과 입력 크기의 관계를 나타내는 함수
- 연산이 많아질수록 오래 걸림
- PS에서는 네트워크 통신, CPU 점유 등은 신경 안 씀
- 연산을 몇 번 수행하는지 확인하면 수행 시간을 대략적으로 유추할 수 있음
- 최악의 경우가 중요함

- 문제를 해결하는데 필수적인 기본 연산을 정한 다음
- 기본 연산의 수행 횟수를 계산

시간 복잡도

길이가 N인 배열에서 값이 K인 원소가 존재하는지 확인

- 기본 연산: 배열의 한 원소와 K의 값을 비교
- 기본 연산의 수행 횟수: 최대 N회
- 시간 복잡도: T(N) = N

시간 복잡도

길이 N인 배열에서 최댓값 찾기

- 기본 연산: 배열에서 두 원소의 값 비교
- 기본 연산의 수행 횟수: 최대 N-1회
- 시간 복잡도 T(N) = N-1

시간 복잡도

길이가 각각 N, M인 정렬된 두 배열을 정렬된 하나의 배열로 합치는 과정

- 기본 연산: 두 배열의 원소를 서로 비교
- 기본 연산의 수행 횟수: 최대 N+M-1번
- 시간 복잡도 T(N, M) = N+M-1

시간 복잡도

정수 N이 소수인지 판별하는 것

- 기본 연산: N이 어떤 정수 k로 나누어 떨어지는지 확인하는 것
- 기본 연산의 수행 횟수: 최대 $\sqrt{N}$번
- 시간 복잡도 $T(N) = \sqrt{N}$
  - 입력의 크기에 대해 나타내면 $T(L) = B^{L/2}$, B는 진법

질문?

점근 표기법

함수에서 가장 중요한 정보

- $f(x) = x^2 + 5x, g(x) = 27x$
  - x = 1: f(x) = 6, g(x) = 27
  - x = 10: f(x) = 150, g(x) = 270
  - x = 100: f(x) = 10500, g(x) = 2700
  - x = 1000: f(x) = 1005000, g(x) = 27000

- x가 적당히 큰 수면 f(x) > g(x)를 만족함
- $x > x_0$이면 항상 f(x) > g(x)를 만족함

점근 표기법

함수에서 가장 중요한 정보

- 다항 함수: 최고차항의 차수가 중요
  - ex) $f(x) = x^2 + 5x, g(x) = 27x$
  - x > 22이면 항상 f(x) > g(x)
- 지수 함수는 밑수의 최댓값이 중요
  - $f(x) = 3^x + x - 10, g(x) = 2^x + x^3 + 5x$
  - x > 4.6이면 항상 f(x) > g(x)
- x가 충분히 크면 최고차항의 차수 or 밑수의 최댓값이 큰 함수가 더 큼

점근 표기법

Big-O Notation

- $f(x) \in O\big(g(x)\big)$
  - 어떤 실수 x_0과 양의 실수 c가 있어서
  - $x > x_0$을 만족하는 모든 x에 대해
  - $f(x) \leq c \times g(x)$를 만족한다.
- 어떤 실수 x_0보다 큰 모든 x
  - x가 한 없이 커지면 항상 부등호가 성립
- 어떤 양의 실수 c
  - 최고차항의 계수 무시
  - 최고차항의 차수와 지수 항의 밑수가 중요함

점근 표기법

Big-O Notation

- Big-O Notation의 의미
  - $f(x)$가 아무리 빨리 증가해도
  - 최대 $g(x)$에 비례하는 수준으로 증가한다
  - $f(x)$의 상계를 나타냄

- $f(x) = 5x, g(x) = x^2$
  - $x_0 = 5, c = 1$이면 $5x \leq 1 \times x^2 : 5x \in O(x^2)$
- $f(x) = 2^x + 10x + 5, g(x) = 2^x$
  - $x_0 = 7, c = 2$이면 $2^x + 10x + 5 \leq 2 \times 2^x : 2^x + 10x + 5 \in O(2^x)$

질문?

점근 표기법

Big-Omega Notation

- Big-O Notation과 반대로 함수의 하계를 나타냄
- $f(x) \in \Omega(g(x))$
  - 어떤 실수 x_0과 양의 실수 c가 있어서
  - $x > x_0$을 만족하는 모든 x에 대해
  - $f(x) \geq c \times g(x)$를 만족한다.

Big-Theta Notation

- 상계와 하계의 교집합(Tight Bound)
- $f(x) \in \Theta\big(g(x)\big) \Leftrightarrow f(x) \in O\big(g(x)\big) \wedge f(x) \in \Omega(g(x))$
  - f(x) is the order of g(x), f(x) and g(x) are the same order
  - 시간 복잡도가 order of n이다 = Θ(n)이다

점근 표기법

극한을 이용한 표현

- $f(x) \in O\big(g(x)\big) \Leftrightarrow \lim_{n\to\infty} \frac{f(x)}{g(x)} = c \geq 0$로 수렴
- $f(x) \in \Omega\big(g(x)\big) \Leftrightarrow \lim_{n\to\infty} \frac{f(x)}{g(x)} > 0$
- $f(x) \in \Theta\big(g(x)\big) \Leftrightarrow \lim_{n\to\infty} \frac{f(x)}{g(x)} = c > 0$로 수렴

점근 표기법

다양한 시간 복잡도

- 복잡도의 위계: $1 \ll \log\log n \ll \log^k n \ll \sqrt{n} \ll n \ll n\log^k n \ll n\sqrt{n} \ll n^2 \ll 2^n \ll 3^n \ll n! \ll n^n$
- 헷갈리는 복잡도
  - 양의 상수 c에 대해 $c^{\log n} = n^{\log c}$이므로 n에 대한 다항 시간
  - $\log(n!)$
    - $\log(n!) = \log 1 + \log 2 + \cdots + \log n \le \log n + \log n + \cdots + \log n = n\log n$
    - $\log(n!) = \log 1 + \log 2 + \cdots + \log n \ge \log\frac{n}{2} + \log\left(\frac{n}{2}+1\right) + \cdots + \log n \ge \log\frac{n}{2} + \log\frac{n}{2} + \cdots + \log\frac{n}{2} = \frac{n}{2}\log\frac{n}{2}$
    - $\frac{n}{2}\log\frac{n}{2} \le \log(n!) \le n\log n$이므로 $\log(n!) \in \Theta(n!)$
    - 또는 Stirling's Approximation에 의해 $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$이므로 $\log(n!) \approx n(\log n - 1) + \frac{1}{2}\log(2\pi n)$임을 알 수 있음
  - $(\log n)!$
    - Stirling's Approximation에 의해 $(\log n)! \approx \sqrt{2\pi\log n}\left(\frac{\log n}{e}\right)^{\log n}$
    - $(\log n)^{\log n} = \left(e^{\log\log n}\right)^{\log n} = e^{\log n \log\log n} = \left(e^{\log\log n}\right)^{\log n} = n^{\log\log n}$이므로 모든 다항 함수보다 빠르게 증가함

- 사실 log (n!) 빼고는 몰라도 됨

질문?