Lecture 3: Introduction to Algorithms

BT 3051 – Data Structures and Algorithms for Biology

Karthik Raman
Department of Biotechnology
Bhupat and Jyoti Mehta School of Biosciences
Indian Institute of Technology Madras

Introduction

What is an algorithm?

Courtesy: Chris Lacher, Florida State University (CIS 4930)

- Well-defined computational procedure that operates on an input set of values (perhaps empty)
- An algorithm is characterised by the following:
  - Assumptions: Things that must be true before the algorithm is executed
  - Outcomes: Things asserted to be true after the algorithm is executed
  - Proof: "If the assumptions are true and the algorithm is executed, the outcomes are true"
  - Runtime: Time required to execute the algorithm — expressed as asymptotic estimate as a function of input size
  - Run space: Space required to execute the algorithm

What is an algorithm?

Courtesy: Chris Lacher, Florida State University (CIS 4930)

- Well-defined computational procedure that operates on an input set of values (perhaps empty)
- An algorithm is characterised by the following:
  - Assumptions: Things that must be true before the algorithm is executed
  - Outcomes: Things asserted to be true after the algorithm is executed
  - Proof: "If the assumptions are true and the algorithm is executed, the outcomes are true"
  - Runtime: Time required to execute the algorithm — expressed as asymptotic estimate as a function of input size
  - Run space: Space required to execute the algorithm

What is an algorithm?

Courtesy: Chris Lacher, Florida State University (CIS 4930)

- Well-defined computational procedure that operates on an input set of values (perhaps empty)
- An algorithm is characterised by the following:
  - Assumptions: Things that must be true before the algorithm is executed
  - Outcomes: Things asserted to be true after the algorithm is executed
  - Proof: “If the assumptions are true and the algorithm is executed, the outcomes are true”
  - Runtime: Time required to execute the algorithm — expressed as asymptotic estimate as a function of input size
  - Run space: Space required to execute the algorithm

What is an algorithm?

Courtesy: Chris Lacher, Florida State University (CIS 4930)

- Well-defined computational procedure that operates on an input set of values (perhaps empty)
- An algorithm is characterised by the following:
  - Assumptions: Things that must be true before the algorithm is executed
  - Outcomes: Things asserted to be true after the algorithm is executed
  - Proof: “If the assumptions are true and the algorithm is executed, the outcomes are true”
  - Runtime: Time required to execute the algorithm — expressed as asymptotic estimate as a function of input size
  - Run space: Space required to execute the algorithm

What is an algorithm?

Courtesy: Chris Lacher, Florida State University (CIS 4930)

- Well-defined computational procedure that operates on an input set of values (perhaps empty)
- An algorithm is characterised by the following:
  - Assumptions: Things that must be true before the algorithm is executed
  - Outcomes: Things asserted to be true after the algorithm is executed
  - Proof: "If the assumptions are true and the algorithm is executed, the outcomes are true"
  - Runtime: Time required to execute the algorithm — expressed as asymptotic estimate as a function of input size
  - Run space: Space required to execute the algorithm

What is an algorithm?

Courtesy: Chris Lacher, Florida State University (CIS 4930)

- Well-defined computational procedure that operates on an input set of values (perhaps empty)
- An algorithm is characterised by the following:
  - Assumptions: Things that must be true before the algorithm is executed
  - Outcomes: Things asserted to be true after the algorithm is executed
  - Proof: “If the assumptions are true and the algorithm is executed, the outcomes are true”
  - Runtime: Time required to execute the algorithm — expressed as asymptotic estimate as a function of input size
  - Run space: Space required to execute the algorithm

What is an algorithm?

Courtesy: Chris Lacher, Florida State University (CIS 4930)

- Well-defined computational procedure that operates on an input set of values (perhaps empty)
- An algorithm is characterised by the following:
  - Assumptions: Things that must be true before the algorithm is executed
  - Outcomes: Things asserted to be true after the algorithm is executed
  - Proof: "If the assumptions are true and the algorithm is executed, the outcomes are true"
  - Runtime: Time required to execute the algorithm — expressed as asymptotic estimate as a function of input size
  - Run space: Space required to execute the algorithm

History of Algorithms[a]

[a] http://cs-exhibitions.uni-klu.ac.at/index.php?id=193

History of Algorithms

FROM THE EDITORS

THE JOY OF ALGORITHMS

Francis Sullivan, Associate Editor-in-Chief

> *"For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing."*
>
> — *Francis Sullivan*

Top 10 Algorithms of the 20th Century

- Monte Carlo/Metropolis
- Simplex method (LP)
- Krylov Subspace Iteration
- Householder Matrix Decomposition
- Fortran Compiler
- QR algorithm
- Quicksort
- FFT
- Integer Relation Detection
- Fast Multipole

Self-assessment Exercise

- Read through the top ten algorithms of the century
- Write a paragraph (300–400 words) about one of these algorithms, highlighting the greatness (coolness!) of the algorithm and its applications
- Outcome
  - Inspiration!
  - Technical writing practice!

Features of an Algorithm

"An algorithm is a well-ordered collection of unambiguous and effectively computable operations that when executed produces a result and halts in a finite amount of time"
— Schneider, M. and J. Gersting (1995), *An Invitation to Computer Science*, West Publishing Company, New York, NY

- An algorithm must terminate! (Finiteness)
- An algorithm must be clearly defined (Definiteness)
- An algorithm must produce the correct result (Correctness)
- Should work on a defined class of inputs, to produce the expected output
- Algorithms must produce an output!

Features of an Algorithm

> *"An algorithm is a well-ordered collection of unambiguous and effectively computable operations that when executed produces a result and halts in a finite amount of time"*
> — Schneider, M. and J. Gersting (1995), *An Invitation to Computer Science*, West Publishing Company, New York, NY

- An algorithm must terminate! (Finiteness)
- An algorithm must be clearly defined (Definiteness)
- An algorithm must produce the correct result (Correctness)
- Should work on a defined class of inputs, to produce the expected output
- Algorithms must produce an output!

Features of an Algorithm

"An algorithm is a well-ordered collection of unambiguous and effectively computable operations that when executed produces a result and halts in a finite amount of time"
— Schneider, M. and J. Gersting (1995), *An Invitation to Computer Science*, West Publishing Company, New York, NY

- An algorithm must terminate! (Finiteness)
- An algorithm must be clearly defined (Definiteness)
- An algorithm must produce the correct result (Correctness)
- Should work on a defined class of inputs, to produce the expected output
- Algorithms must produce an output!

Features of an Algorithm

"An algorithm is a well-ordered collection of unambiguous and effectively computable operations that when executed produces a result and halts in a finite amount of time"
— Schneider, M. and J. Gersting (1995), *An Invitation to Computer Science*, West Publishing Company, New York, NY

- An algorithm must terminate! (Finiteness)
- An algorithm must be clearly defined (Definiteness)
- An algorithm must produce the correct result (Correctness)
- Should work on a defined class of inputs, to produce the expected output
- Algorithms must produce an output!

Features of an Algorithm

"An algorithm is a well-ordered collection of unambiguous and effectively computable operations that when executed produces a result and halts in a finite amount of time"
— Schneider, M. and J. Gersting (1995), *An Invitation to Computer Science*, West Publishing Company, New York, NY

- An algorithm must terminate! (Finiteness)
- An algorithm must be clearly defined (Definiteness)
- An algorithm must produce the correct result (Correctness)
- Should work on a defined class of inputs, to produce the expected output
- Algorithms must produce an output!

Features of an Algorithm

"An algorithm is a well-ordered collection of unambiguous and effectively computable operations that when executed produces a result and halts in a finite amount of time"
— Schneider, M. and J. Gersting (1995), *An Invitation to Computer Science*, West Publishing Company, New York, NY

- An algorithm must terminate! (Finiteness)
- An algorithm must be clearly defined (Definiteness)
- An algorithm must produce the correct result (Correctness)
- Should work on a defined class of inputs, to produce the expected output
- Algorithms must produce an output!

Examples

Examples: Fibonacci numbers

Pingala/Fibonacci sequence

The Fibonacci sequence $F_1, F_2, \ldots, F_n, \ldots$ is defined as:

$F_1 = 1$

$F_2 = 1$

$F_n = F_{n-1} + F_{n-2}$, for all $n > 2$

- Fibonacci numbers appear in Indian and Western math
- Named after Leonardo Fibonacci (book dating 1202 CE)
- Fibonacci numbers are intimately connected with the golden ratio
- They also appear in biological settings:
  - branching in trees
  - arrangement of leaves on a stem
  - ...

Pingala/Fibonacci sequence

The Fibonacci sequence $F_1, F_2, \ldots, F_n, \ldots$ is defined as:

$F_1 = 1$

$F_2 = 1$

$F_n = F_{n-1} + F_{n-2}$, for all $n > 2$

- Fibonacci numbers appear in Indian and Western math
- Named after Leonardo Fibonacci (book dating 1202 CE)
- Fibonacci numbers are intimately connected with the golden ratio
- They also appear in biological settings:
  - branching in trees
  - arrangement of leaves on a stem
  - ...

Pingala/Fibonacci sequence

The Fibonacci sequence $F_1, F_2, \ldots, F_n, \ldots$ is defined as:

$F_1 = 1$

$F_2 = 1$

$F_n = F_{n-1} + F_{n-2}$, for all $n > 2$

- Fibonacci numbers appear in Indian and Western math
- Named after Leonardo Fibonacci (book dating 1202 CE)
- Fibonacci numbers are intimately connected with the golden ratio
- They also appear in biological settings:
  - branching in trees
  - arrangement of leaves on a stem
  - ...

Pingala/Fibonacci sequence

The Fibonacci sequence $F_1, F_2, ..., F_n, ...$is defined as:

$F_1 = 1$

$F_2 = 1$

$F_n = F_{n-1} + F_{n-2}$, for all $n > 2$

- Fibonacci numbers appear in Indian and Western math
- Named after Leonardo Fibonacci (book dating 1202 CE)
- Fibonacci numbers are intimately connected with the golden ratio
- They also appear in biological settings:
  - branching in trees
  - arrangement of leaves on a stem
  - ...

How do we compute F_n?

Closed form solution exists:

$$F_n = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$$

where $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = \frac{1-\sqrt{5}}{2}$

but ...

How do we compute F_n?

Closed form solution exists:

$$F_n = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$$

where $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = \frac{1-\sqrt{5}}{2}$

but ...

Naïve Method using Recurrence

fib_v1.py

```python
def fibo(n):
    if n == 1 or n == 2:
        return 1
    else:
        return fibo(n - 1) + fibo(n - 2)

print fibo(10)
```

Output:

```
>>>
55
```

Naïve Method using Recurrence

fib_v1.py

```
def fibo(n):
    if n == 1 or n == 2:
        return 1
    else:
        return fibo(n - 1) + fibo(n - 2)

print fibo(10)
```

Output:

```
>>>
55
```

How much computation does it involve?

Iterative Method
fib_v2.py

```
def fibo(n):
    fib = [0] * n
    fib[0] = 1
    fib[1] = 1

    for i in range(2, n):
        fib[i] = fib[i - 1] + fib[i - 2]

    return fib[n - 1]

print fibo(10)
```

Output:

```
>>>
55
```

Iterative Method

fib_v2.py

```
def fibo(n):
    fib = [0] * n
    fib[0] = 1
    fib[1] = 1

    for i in range(2, n):
        fib[i] = fib[i - 1] + fib[i - 2]

    return fib[n - 1]

print fibo(10)
```

Output:

```
>>>
55
```

What's the drawback?

Improved Iterative Method

fib_v3.py

```
def fibo(n):
    fibn = 1   # F_n
    fibn1 = 1   # F_{n-1}

    for dummy in range(2,n):
        fib = fibn + fibn1
        fibn1 = fibn
        fibn = fib

    return fib

print(fibo(10))
```

Output:

```
>>>
55
```

Improved Iterative Method

fib_v3.py

```
def fibo(n):
    fibn = 1   # F_n
    fibn1 = 1   # F_{n-1}

    for dummy in range(2,n):
        fib = fibn + fibn1
        fibn1 = fibn
        fibn = fib

    return fib

print(fibo(10))
```

Output:

```
>>>
55
```

Can we do better?

Trick!

Consider the matrix **A**:

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

For $n \geq 2$,

$$A^{n-1} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix}$$

Trick!

Consider the matrix **A**:

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

For $n \geq 2$,

$$A^{n-1} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} = \begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix}$$

So what?

Examples: Exponentiating a number

How to compute x^n?

Algorithm 1: Naïve exponentiation

input : x, n
output: $p = x^n$
begin
  $p \longleftarrow 1$
  for $x = 1$ **to** n **do**
    $p \longleftarrow p * x$

How to compute x^n?

Algorithm 2: Naïve exponentiation

---

input : x, n

output: $p = x^n$

begin

  $p \longleftarrow 1$

  for $x = 1$ **to** n **do**

    $p \longleftarrow p * x$

Can we do better?

How to compute x^n faster?

```python
def fastExpo(x, n):
    if n == 1:
        return x
    if n == 2:
        return x * x
    if n % 2 == 0:
        p = fastExpo(x, n // 2)
        return p * p
    else:
        return x * fastExpo(x, n - 1)
```

How to compute x^n faster?

```
def fastExpo(x, n):
    if n == 1:
        return x
    if n == 2:
        return x * x
    if n % 2 == 0:
        p = fastExpo(x, n // 2)
        return p * p
    else:
        return x * fastExpo(x, n - 1)
```

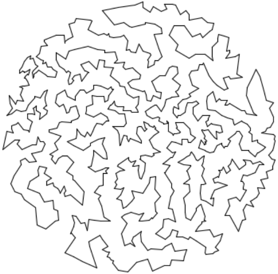
Can we do better?

Examples: Travelling Salesperson Problem

Travelling Salesperson Problem (TSP)

Given N cities and the distances between every pair of cities, the goal of a travelling salesperson is to visit all of cities exactly once (and return to the origin) while keeping the total distance travelled as short as possible.

1000 'cities'



Optimal Tour

http://www.cs.princeton.edu/courses/archive/fall14/cos126/assignments/tsp.html

Travelling Salesperson Problem

The importance of the TSP does not arise from an overwhelming demand of salespeople to minimize their travel distance, but rather from a wealth of other applications such as vehicle routing, circuit board drilling, VLSI design, robot control, X-ray crystallography, machine scheduling, and computational biology.

— Robert Sedgewick (Princeton COS 126)

Travelling Salesperson Problem

The importance of the TSP does not arise from an overwhelming demand of salespeople to minimize their travel distance, but rather from a wealth of other applications such as vehicle routing, circuit board drilling, VLSI design, robot control, X-ray crystallography, machine scheduling, and computational biology.

— Robert Sedgewick (Princeton COS 126)

How to solve this problem?