CS381: Numerical Computation & Softwares

Newton Interpolation

Jakramate Bootkrajang

Department of Computer Science

Chiang Mai University

Outline

- Recursive function
- Data type
- Implementing divided difference

Fibonacci numbers

Fibonacci numbers คืออนุกรมของตัวเลขจำนวนเต็มที่ค่าของพจน์ที่ n มีค่าเท่ากับค่าของพจน์ที่ $n-1$ บวกกับ พจน์ที่ $n-2$

$$F_n = F_{n-1} + F_{n-2}$$

โดยที่กำหนดไว้ว่า พจน์ที่ 0 มีค่าเท่ากับ 0 และพจน์ที่ 1 มีค่าเท่ากับ 1 ตัวอย่างของ Fibonacci numbers

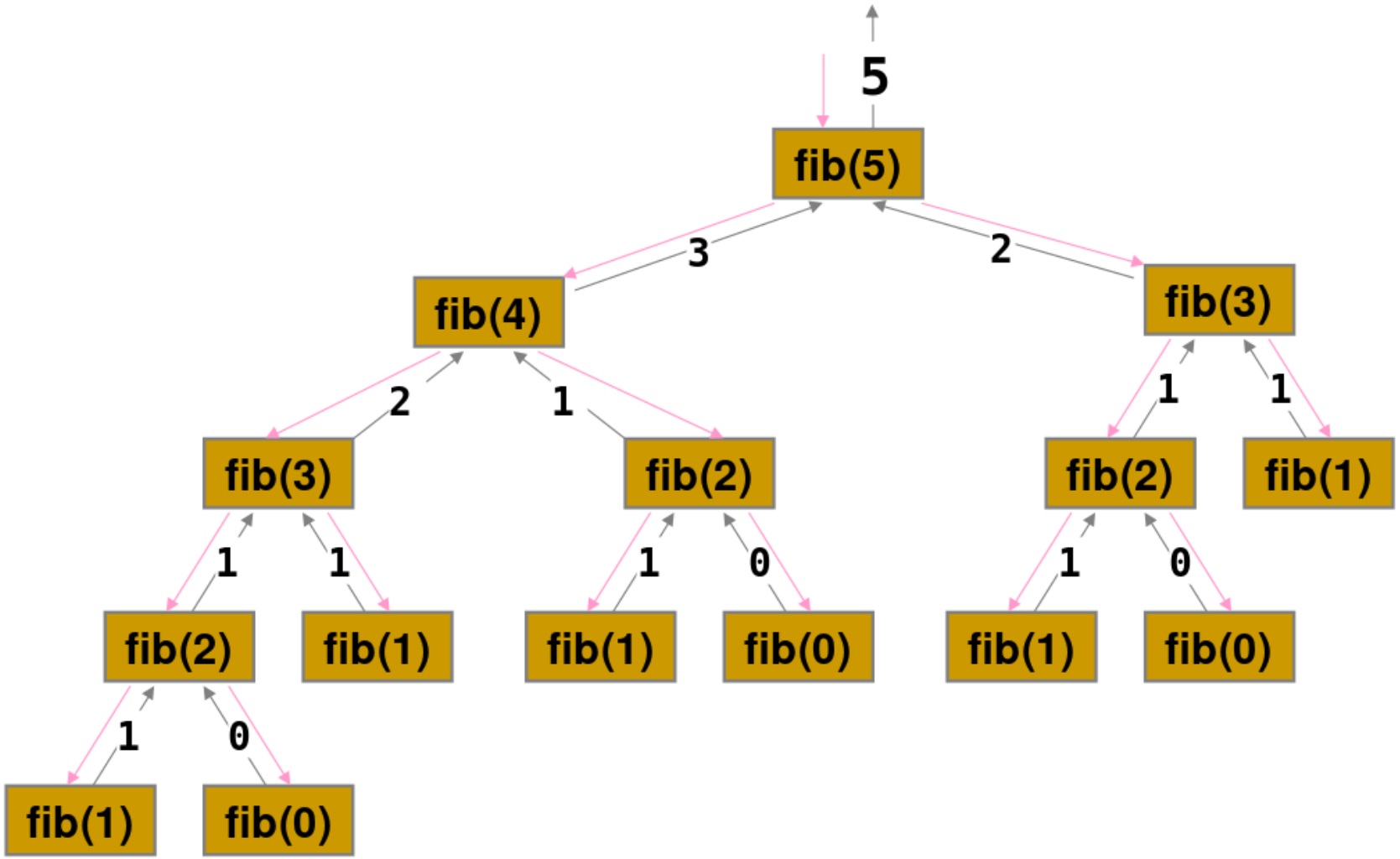
$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

การคำนวณ Fibonacci number

- โดยทั่วไปแล้ว การคำนวณพจน์ที่ n ในอนุกรม Fibonacci สามารถทำได้ไม่ยาก หากเราแตกปัญหาออกเป็นปัญหาย่อยหลายๆปัญหา
- หากต้องการจะคำนวณ F_5 เราจะทำอย่างไร

แผนผังการคำนวณ F_5

$0, 1, 1, 2, 3, 5, \ldots$



Recursive calculation

- การคำนวณ Fibonacci number ข้างต้นสามารถแตกปัญหาออกเป็นปัญหาย่อยที่เหมือนกัน
- นั่นคือการจะหา F_5 ได้เราจำเป็นจะต้องไปหา F_4 และ F_3 และย่อยลงไปเรื่อยๆจนถึง F_1 กับ F_0
- เราเรียกการคำนวณแบบนี้ว่าการคำนวณแบบ Recursive
- สังเกตว่าการคำนวณแบบนี้จะไปสิ้นสุดตรงปัญหาเล็กที่สุดที่สามารถหาคำตอบได้โดยง่าย ในที่นี้คือ F_1 กับ F_0

Recursive function

- โดยปกติแล้วการมองปัญหาเป็นแบบ Recursive เราจะต้องแบ่งกรณีการคำนวณเป็น
  - Base case = เคสที่คำตอบปัญหาสามารถหาได้โดยง่าย
  - Recursive case = เคสที่ยังหาคำตอบไม่ได้ทันที แต่ต้องแก้ปัญหาที่มีลักษณะที่มีขนาดเล็กลงให้ได้ก่อน
  - ใช้ if-else เพื่อตรวจสอบว่าเป็น case ไหน
- สำหรับ Fibonacci numbers problem
  - Base case = F_0 และ F_1
  - Recursive case = F_n

Writing a recursive Fibonacci function

```
function fibonacci(n)
    if n == 0 || n == 1
        return 1
    else
        return fibonacci(n-1) + fibonacci(n-2)
    end
end
```

How about factorial ?

ให้นักศึกษาลองเขียนโปรแกรมคำนวณ factorial โดยวิธี Recursion

Data type

- ประเภทของข้อมูลใน Julia แบ่งออกเป็นสองแบบคือ
  - Primitive type เช่น Integer, Boolean, Float, String
  - User defined type
- ซึ่งเราก็จะเลือกประเภทของข้อมูลให้เหมาะสมกับข้อมูลที่เราต้องการจะเก็บ
- ในบางครั้งข้อมูลที่เราต้องการจะเก็บไม่ตรงกับประเภทของข้อมูลใน primitive type
- ทำให้เราต้องนิยามประเภทของข้อมูลขึ้นมาใหม่

เหตุการณ์จำลอง

- สมมุติว่าเราต้องการสร้างโปรแกรมเพื่อเก็บข้อมูล (ชื่อจริง ชื่อเล่น อายุ) ของเพื่อนนักศึกษาหลายๆคน

- แต่ Julia ไม่มี data type ที่ใช้เก็บข้อมูลบุคคลอยู่ เราจึงจะนิยามประเภทข้อมูลขึ้นมาใหม่โดยใช้ syntax การสร้าง type

```
type nameOfType
    variable1::primitive_type
        ...
    variableN::primitive_type
end
```

เหตุการณ์จำลอง

- ตัวอย่าง Type ที่เอาไว้ระบุบุคคล

```
type person
    name::string
    nickname::string
    age::integer
end
```

Working with type

การประกาศตัวแปรที่มี type แบบที่เราเพิ่งกำหนดไป ทำได้โดยเรียกฟังก์ชันที่มีชื่อเดียวกันกับ type แล้วใส่ parameter ให้ตรงกับจำนวนตัวแปรของ type นั้น

```
closefriend = person("Tom","Hank", 58)
friends = [person("Donald","Trump", 55),
           person("Teresa", "May", 56)]
```

การเข้าถึงตัวแปรภายใน type

หากต้องการเข้าถึงตัวแปรภายใน type ทำได้โดยใช้เครื่องหมาย . (dot)

```
closefriend = person("Tom","Hank", 58)
println(closefriend.name)  # will print "Tom"
```

Implementing Newton interpolation

เราอาจเริ่มจากการสร้าง type เพื่อเอาไว้เก็บข้อมูลที่อยู่ในรูป (x, y) ก่อน

```
type point
    x::Float64    # IEEE floating point 64 bits
    y::Float64    # IEEE floating point 64 bits
end
```

จากนั้นเราอาจเก็บ point หลายๆตัวไว้ในตัวแปรประเภท Vector ที่ชื่อ data

```
data = Vector(10)    # this is just 1D array
for i=1:10
    data[i] = point(i,sind(i))
end
```

แอบดู point

เมื่อสร้างชุดข้อมูลแล้วเราสามารถเข้าดู point ต่างๆได้เหมือนการเข้าถึง array ทั่วไป

```
println(data[1].x)  # x value of the 1st data point
println(data[1].y)  # y value of the 1st data point
```

ค่าของมันคืออะไรเอ่ย

Newton polynomial

$$Q_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \ldots + f[x_0, \ldots, x_n] \prod_{k=0}^{n-1} (x - x_k)$$

โดยที่

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_2, \ldots, x_n] - f[x_1, \ldots, x_{n-1}]}{x_n - x_0}$$

Computing a divided difference recursively

- Base case

$$f[x_n] = y_n$$

- Recursive case

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_2, \ldots, x_n] - f[x_1, \ldots, x_{n-1}]}{x_n - x_0}$$

(Some of) The code

```
function divdiff(d)     # d is the input dataset
    if length(d)==1
        return d[1].y
    else
        return (divdiff(d[2:end]) - divdiff(d[1:end-1
    end
end
```

Must read

- Learn X in Y minutes for Julia

`https://learnxinyminutes.com/docs/julia/`