Introduction to Computer and Programming

Lecture 5

Yue Zhang
Westlake University

August 1, 2023

WestlakeNLP

Chapter 5.

Problem Solving

WestlakeNLP

Typical Problems

- Summation
- Maximum
- Decision
- Counting
- Iterative Calculation

WestlakeNLP

Summation

The General Form

$$s = \sum_{i=m}^{n} f(i)$$

WestlakeNLP

Summation

The General Form

$$s = \sum_{i=m}^{n} f(i)$$

when $f(i) = i$,

$$s = \sum_{i=m}^{n} i$$

WestlakeNLP

Summation

The General Form

$$s = \sum_{i=m}^{n} f(i)$$

when $f(i) = i$,

$$s = \sum_{i=m}^{n} i$$

First enumerate i, and then accumulate s.

WestlakeNLP

Summation

$$s = \sum_{i=m}^{n} i$$

sum.py

```
# initialization
m = int(input("m="))
n = int(input("n="))
s = 0
# summation
i = m             # the iterator
while i <= n:
    s += i        # update s
    i += 1
# output
print("The sum of integer from", m, "to", n, "is", s)
```

Summation

```
Yues~MacBook~Pro:code$ python sum.py
m=1
n=10
The sum of integer from 1 to 10 is 55
Yues~MacBook~Pro:code$ python sum.py
m=5
n=10
The sum of integer from 5 to 10 is 45
```

WestlakeNLP

Summation

The General Form

$$s = \sum_{i=m}^{n} f(i)$$

when $f(i) = log(i)$,

$$s = \sum_{i=m}^{n} log(i)$$

First enumerate i, and then accumulate s.

WestlakeNLP

Summation

$$s = \sum_{i=m}^{n} log(i)$$

sum_log.py

```
import math                          # NEW
# initialization
m = int(input("m="))
n = int(input("n="))
s = 0
# summation
i = m  # the iterator
while i <= n:
    s += math.log(i)                 # NEW
    i += 1
# output
print("The sum from log(", m, ") to log(", n, ") is ", s)
```

Summation

```
Yues~MacBook~Pro:code$ python sum_log.py
m=1
n=2
The sum from log(1) to log(2) is 0.6931471805599453
Yues~MacBook~Pro:code$ python sum_log.py
m=4
n=8
The sum from log(4) to log(8) is 8.812843433517195
```

WestlakeNLP

Product

$$s = \prod_{i=m}^{n} f(i)$$

WestlakeNLP

Product

$$s = \prod_{i=m}^{n} f(i)$$

- Still enumerate i, and then update s.
- The initial value should be?
- The incremental step should be?

WestlakeNLP

Product

$$s = \prod_{i=m}^{n} i$$

product.py

```
# initialization
m = int(input("m="))
n = int(input("n="))
s = 1                                   # NEW
# product
i = m  # the iterator
while i <= n:
    s *= i                              # NEW
    i += 1
# output
print("The product of integer from", m, "to", n, "is", s)
```

Product

```
Yues~MacBook~Pro:code$ python product.py
m=1
n=3
The product of integer from 1 to 3 is 6
Yues~MacBook~Pro:code$ python product.py
m=5
n=7
The product of integer from 5 to 7 is 210
```

WestlakeNLP

Maximum and Minimum

$$s = max_{i=m}^{n} f(i)$$

$$s = min_{i=m}^{n} f(i)$$

WestlakeNLP

Maximum and Minimum

$$s = max_{i=m}^{n} f(i)$$

$$s = min_{i=m}^{n} f(i)$$

- Still enumerate i, and then update s.
- The initial value should be?
- The incremental step should be?

WestlakeNLP

Maximum and Minimum

$$s = max_{i=m}^{n} f(i)$$

$$s = min_{i=m}^{n} f(i)$$

- Still enumerate i, and then update s.
- The initial value should be? $-\infty$ / $+\infty$
- The incremental step should be?

$-\infty$

```
if f(i) > s:
    s = f(i)
```

$+\infty$

```
if f(i) < s:
    s = f(i)
```

WestlakeNLP

Maximum and Minimum

$$s = max_{i=m}^{n} sin(i^2)$$

max.py

```
import math
# initialization
m = int(input("m="))
n = int(input("n="))
s = float("-inf")                # NEW
# summation
i = m  # the iterator
while i <= n:
    f = math.sin(i*i)            # NEW
    s = f if f > s else s        # NEW
    i += 1
# output
print("The maximum from sin(", m, "^2 ) to sin(", n, "^2 ) is ", s)
```

Maximum and Minimum

```
Yues~MacBook~Pro:code$ python max.py
m=1
n=8
The maximum from sin(1^2) to sin(8^2) is 0.9200260381967906
Yues~MacBook~Pro:code$ python max.py
m=3
n=6
The maximum from sin(3^2) to sin(6^2) is 0.4121184852417566
```

WestlakeNLP

Maximum and Minimum

$$s = max_{i=m}^{n} sin(i^2)$$
$$a = argmax_{i=m}^{n} sin(i^2)$$

- argmax: Get the value of i when s reaches the maximum.
- Need to record both s and a.
- Only the incremental step needs a modification.

WestlakeNLP

Maximum and Minimum

$$s = max_{i=m}^{n} sin(i^2),\ a = argmax_{i=m}^{n} sin(i^2)$$

argmax.py

```
import math
# initialization
m = int(input("m="))
n = int(input("n="))
s = float("-inf")
# argmax
i = m  # the iterator
a = m                                   # NEW
while i <= n:
    f = math.sin(i*i)
    if f > s:                           # NEW
        s = f                           # NEW
        a = i                           # NEW - obatin the value of i
    i += 1
# output
print("Then the maximum's i from sin(m^2) to sin(n^2) is", a)
```

Maximum and Minimum

```
Yues~MacBook~Pro:code$ python argmax.py
m=1
n=8
Then the maximum's i from sin(m^2) to sin(n^2) is 8
Yues~MacBook~Pro:code$ python argmax.py
m=3
n=6
Then the maximum's i from sin(m^2) to sin(n^2) is 3
```

WestlakeNLP

Deciding

- Is there an integer root for $x^4 - 93x - 19620 = 0$ between m and n?

WestlakeNLP

Deciding

- Is there an integer root for
  $$x^4 - 93x - 19620 = 0$$ between m and n?
- Enumerate i, check equation value.
- The method is called **searching**.

WestlakeNLP

Deciding

root.py

```
# initialization
m = int(input("m="))
n = int(input("n="))
# enumeration
i = m  # loop variable
while i <= n:
    if i**4 - 93*i - 19620 == 0:
        print("There is a root.")
        break
    i += 1
else:
    print("There is no root in the range.")
```

WestlakeNLP

Deciding

```
Yues~MacBook~Pro:code$ python root.py
m=50
n=100
There is no root in the range.
Yues~MacBook~Pro:code$ python root.py
m=1
n=100
There is a root.
```

WestlakeNLP

Counting

- Count the number of integers i in $[m, n]$ where $sin(i^3) > 0$

$$\{i : i \in [m, n], s.t. sin(i^3) > 0\}$$

- enumerate, check and accumulate

WestlakeNLP

Counting

count.py

```
import math
# initialization
m = int(input("m="))
n = int(input("n="))
s = 0
i = m
# enumeration
while i <= n:
    if math.sin(i*i*i) > 0:
        s += 1
    i += 1
print("There are " + str(s) + " integers that satisfy the
    condition sin(i^3) > 0 in the given range.")
```

WestlakeNLP

Counting

```
Yues~MacBook~Pro:code$ python count.py
m=1
n=10
There are 8 integers that satisfy the condition sin(i^3) > 0
    in the given range.
Yues~MacBook~Pro:code$ python count.py
m=-5
n=6
There are 6 integers that satisfy the condition sin(i^3) > 0
    in the given range.
```

WestlakeNLP

Iterative Computation

Fibonacci Numbers

$$f_0 = 1$$
$$f_1 = 1$$
$$f_i = f_{i-1} + f_{i-2}, i >= 2$$

$$1, 1, 2, 3, 5, 8, 13, 21, \cdots$$

WestlakeNLP

Iterative Computation

Fibonacci Numbers

time step i:

$$f_i \leftarrow f_{i-1} + f_{i-2}$$

time step $i+1$:

$$f_{i-1} \leftarrow \text{old } f_i$$

$$f_{i-2} \leftarrow \text{old } f_{i-1}$$

WestlakeNLP

Iterative Computation

Fibonacci Numbers

fib_iter.py

```
# initialization
n = int(input("Input the index of n="))
x0 = 1                      # f_{i-2}
x1 = 1                      # f_{i-1}
# iteration
i = 2
while i <= n:
    x = x0 + x1             # f_i
    x1 = x0
    x0 = x
    i += 1
print("The " + str(n+1) + "the fibonacci number is " + str(x))
```

Can x1=x0 and x0=x be swapped?

Iterative Computation

Fibonacci Numbers

```
Yues~MacBook~Pro:code$ python fib_iter.py
Input the index of n=10
The 10th fibonacci number is 55
Yues~MacBook~Pro:code$ python fib_iter.py
Input the index of n=11
The 11th fibonacci number is 89
Yues~MacBook~Pro:code$ python fib_iter.py
Input the index of n=20
The 20th fibonacci number is 6765
```

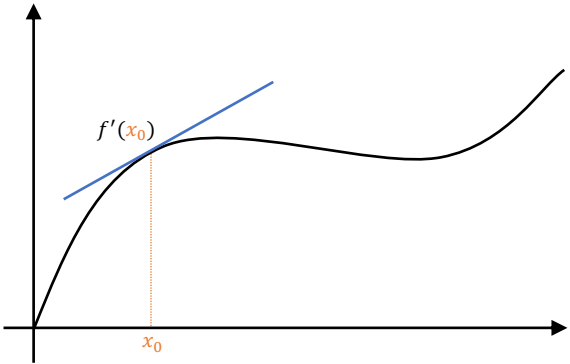
WestlakeNLP

Numerical Analysis

- Numerical Differentiation
- Numerical Integration
- Monte Carlo Method

WestlakeNLP

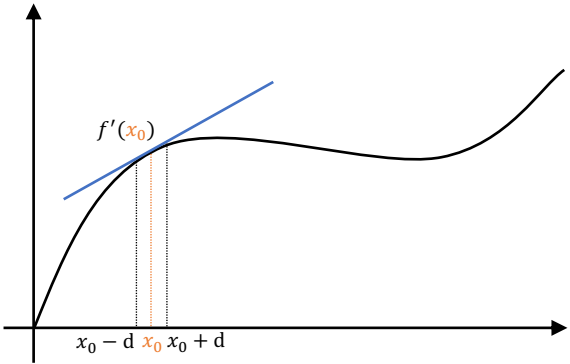
Numerical Differentiation



$$f'(x_0) = ?$$

WestlakeNLP

Numerical Differentiation



$$f'(x_0) \approx (f(x_0 + d) - f(x_0))/d$$

WestlakeNLP

Numerical Differentiation

$$f(x) = x^2$$

differentiation.py

```
f = lambda x: x*x              # f(x)=x^2
x = float(input("x="))
d = float(input("d="))
fp = (f(x+d) - f(x)) / d
print("The derivative is %.2f."%fp)
```

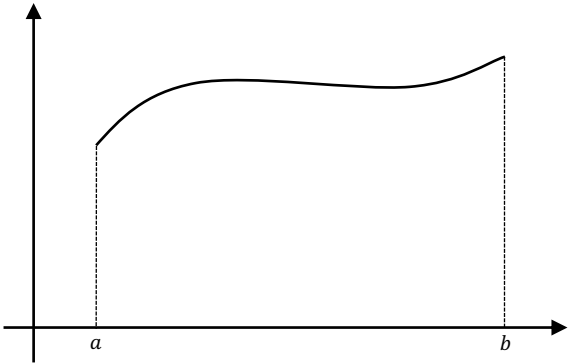
WestlakeNLP

Numerical Differentiation

```
Yues-MacBook-Pro:code$ python differentiation.py
x=3.5
d=0.1
The derivative is 7.10.
Yues-MacBook-Pro:code$ python differentiation.py
x=3.5
d=0.001
The derivative is 7.00.
Yues-MacBook-Pro:code$ python differentiation.py
x=3.5
d=0.00001
The derivative is 7.00.
```

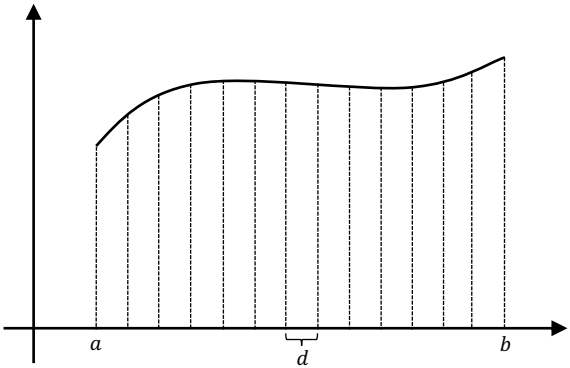
- Exact result: $f'(x) = 2x \rightarrow 7.0$
- Try other functions
  - f = lambda x: math.sin(x)
  - f = lambda x: log(x)
  - f = lambda x: 1/x

Numerical Integration



WestlakeNLP

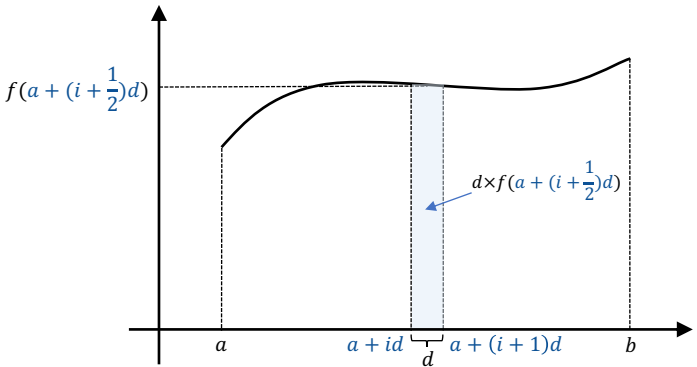
Numerical Integration



n sections, $d = \frac{b-a}{n}$

WestlakeNLP

Numerical Integration



$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} d \cdot f(a+(i+\frac{1}{2}) \cdot d), d = \frac{b-a}{n}$$

WestlakeNLP

Numerical Integration

$$f(x) = x^2$$

integration.py

```
# initialization
f = lambda x: x*x              # f(x)=x^2
a = float(input("a="))
b = float(input("b="))
n = int(input("n="))
d = (b-a)/n
s = 0.0
# iteration
i = 0
while i <= n-1:
    s += f(a+(i+0.5)*d)*d      # sum
    i += 1
print("The integral is %.2f"%s)
```

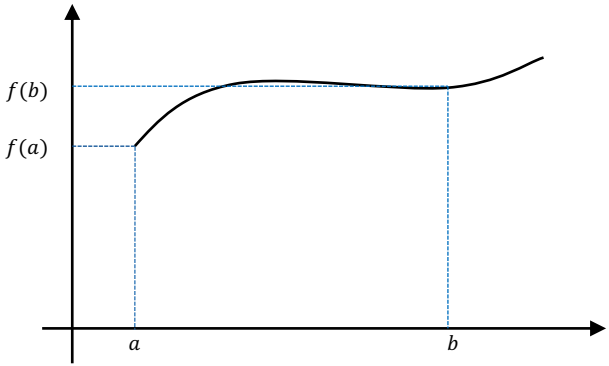
Numerical Integration

```
Yues-MacBook-Pro:code$ python integration.py
a=0
b=10
n=10
The integral is 332.50
Yues-MacBook-Pro:code$ python integration.py
a=0
b=10
n=100
The integral is 333.33
```

- Exact result: $\int_a^b f(x)dx = \frac{1}{3}(b^3 - a^3) = \frac{1}{3} \times 10^3$ where $a = 0, b = 10$
- Try other functions

WestlakeNLP

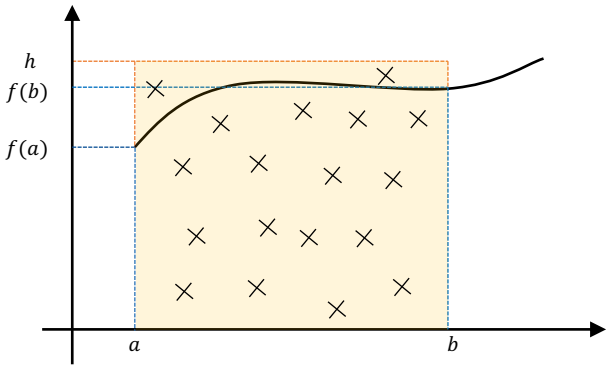
Monte Carlo Integration



WestlakeNLP

Monte Carlo Method

Monte Carlo Integration



- Samples
  M – in the integral area
  N – in the box
- $\frac{M}{N} \approx \frac{\int_a^b f(x)dx}{(b-a)\cdot h}$

$$\therefore \int_a^b f(x)dx \approx h(b-a)\frac{M}{N}$$

WestlakeNLP

Monte Carlo Integration

mc_integration.py

```
import random
f = lambda x: x*x
a = float(input("a="))
b = float(input("b="))
N = int(input("n="))
h = 150                    # Be careful! > max_f(x), x in [a,b]
M = 0
i = 0
while i < N:     # random.random() returns a float in (0,1)
    x = random.random()*(b-a) + a  # random float in (a,b)
    y = random.random()*h          # random float in (0,h)
    if y < f(x):
        M += 1
    i += 1
result = float(h) * (b-a) * (M/N)
print("The integral is %.2f"%result)
```

Monte Carlo Integration

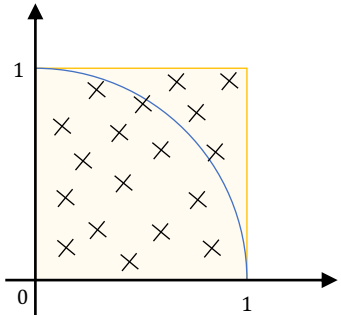
```
Yues-MacBook-Pro:code$ python mc_integration.py
a=0
b=10
n=10000
The integral is 327.30

Yues-MacBook-Pro:code$ python mc_integration.py
a=0
b=10
n=1000000
The integral is 333.76
Yues-MacBook-Pro:code$ python mc_integration.py
a=0
b=10
n=1000000
The integral is 334.65
```

Due to random, the results are different(n=1000000).

Monte Carlo Method

Estimating π



- Samples
  $\frac{M}{N} \approx \frac{1}{4} \times \pi$
  $\pi \approx \frac{4M}{N}$

inside shade: $x^2 + y^2 < 1$

WestlakeNLP

Monte Carlo Method

Estimating π

mcpi.py

```
import random
import math
N = int(input("n="))
M = 0
i = 0
while i < N:    # random.random() returns a float in (0,1)
    x = random.random()
    y = random.random()
    if x*x + y*y < 1:
        M += 1
    i += 1
pi = 4 * (M/N)
print("Approxmate Value: %f"%pi)
print("Error: %f"%(math.pi-pi))
```

Estimating π

```
Yues-MacBook-Pro:code$ python mcpi.py
n=1000
Approxmate Value: 3.124000
Error: 0.017593
Yues-MacBook-Pro:code$ python mcpi.py
n=10000
Approxmate Value: 3.139600
Error: 0.001993
Yues-MacBook-Pro:code$ python mcpi.py
n=1000000
Approxmate Value: 3.142072
Error: -0.000479
```

WestlakeNLP

Tuples

Literals

```
>>> t=(1,2,3)
>>> type(t)
<class 'tuple'>
>>> x='a'
>>> y=True
>>> u=(1.5,x,y) # heterogeneous data
>>> u
(1.5, 'a', True)
>>> type(u)
<class 'tuple'>
>>> w=(1.3,)    # one element
>>> w
(1.3,)
>>> type(w)
<class 'tuple'>
>>> d=(1.3)
>>> d
1.3
>>> type(d)
<class 'float'>
```

Tuples

Operators

```
>>> t1=(1,2,3)
>>> t2=(4,5,6)
>>> t3=t1+t2              # concatenate
>>> t3
(1, 2, 3, 4, 5, 6)
>>> t4=t1*3               # duplicate
>>> t4
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

WestlakeNLP

Tuple Operators

Operator – Slicing

```
>>> t=('a',0,1.5)
>>> t[0]
'a'
>>> t[1:3]
(0, 1.5)
>>> t[-2:]
(0, 1.5)
```

WestlakeNLP

Tuple Operators

Operator – Members

```
>>> t=(1,'a',5.0)
>>> 1 in t
True
>>> 1.0 not in t
False
>>> 'a' in t
True
>>> u=(1,'a',5)
>>> t==u
True
>>> t!=u
False
>>> s='abcdef'
>>> 'abc' in s
True
```

Tuple Operators

Operator – Length

```
>>> t=('a',0,1.5)
>>> len(t)
3
>>> len(t+t)
6
>>> len(t*3)
9
>>> len((2))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: object of type 'int' has no len()
>>> len((1,))
1
>>> s="abcdef"
>>> len(s)
6
```

Tuple Assignment

```
>>> (x,y)=(1,2)
>>> x
1
>>> y
2
>>> (x,y,z)=(3,'a',x)
>>> x
3
>>> y
'a'
>>> z
1
>>> (x,y)=(y,x)     # swapping, no need to worry about overloading
>>> x
'a'
>>> y
3
```

The *for* loop

```
>>> t=(1,'a',True)
>>> for x in t:
...     print(x)
...
1
a
True
```

WestlakeNLP

The *for* loop

```
>>> t=(1,'a',True)
>>> for x in t:
...     print(x)
...
1
a
True
```

Equivalent to

```
>>> t=(1,'a',True)
>>> i=0
>>> while i<len(t):
...     print(t[i])
...     i+=1
...
1
a
True
```

The *for* loop

break and continue

```
>>> t=(1,2,3,4,5,7,10)
>>> for i in t:
...     if i>5:
...         break
...     print(i)
...
1
2
3
4
5
```

```
>>> t=(1,2,3,4,5,7,10)
>>> for i in t:
...     if i%2 == 1:
...         continue
...     print(i)
...
2
4
10
```

WestlakeNLP

The *for* loop

else

```
>>> t=(1,2,3,4,5,7,10)
>>> for i in t:
...     if i==5:
...         print("Found 5!")
...         break
... else:
...     print("'5' is not in tuple")
...
Found 5!
```

WestlakeNLP

The *for* loop

Summation

```
>>> t=(1,2,3,4,5,7,10)
>>> sum(t)
32
```

WestlakeNLP

The *for* loop

Summation

```
>>> t=(1,2,3,4,5,7,10)
>>> sum(t)
32
```

Equivalent to

```
>>> t=(1,2,3,4,5,7,10)
>>> s=0
>>> for i in t:
...     s+=i
...
>>> print(s)
32
```

The *for* loop

Maximum

```
>>> t=(1,2,3,5,4,10,7)
>>> max(t)
10
```

WestlakeNLP

The *for* loop

Maximum

```
>>> t=(1,2,3,5,4,10,7)
>>> max(t)
10
```

Equivalent to

```
>>> t=(1,2,3,5,4,10,7)
>>> m=float('-inf')
>>> for i in t:
...     if i>m:
...         m=i
...
>>> print(m)
10
```

This week check-off:

Math Questions

WestlakeNLP