Python Recap – III

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 1

Computing gcd

- Both versions of gcd take time proportional to $\min(m,n)$
- Can we do better?

```
def gcd(m,n):
  cf = []   # List of common factors
  for i in range(1,min(m,n)+1):
    if (m%i) == 0 and (n%i) == 0:
      cf.append(i)
  return(cf[-1])
```

```
def gcd(m,n):
  for i in range(1,min(m,n)+1):
    if (m%i) == 0 and (n%i) == 0:
      mrcf = i
  return(mrcf)
```

Computing gcd

- Both versions of gcd take time proportional to $\min(m,n)$
- Can we do better?
- Suppose d divides m and n
  - $m = ad$, $n = bd$
  - $m - n = (a-b)d$
  - d also divides $m - n$

```
def gcd(m,n):
  cf = []   # List of common factors
  for i in range(1,min(m,n)+1):
    if (m%i) == 0 and (n%i) == 0:
      cf.append(i)
  return(cf[-1])
```

```
def gcd(m,n):
  for i in range(1,min(m,n)+1):
    if (m%i) == 0 and (n%i) == 0:
      mrcf = i
  return(mrcf)
```

Computing gcd

- Both versions of `gcd` take time proportional to $\min(m, n)$
- Can we do better?
- Suppose d divides m and n
  - $m = ad$, $n = bd$
  - $m - n = (a - b)d$
  - d also divides $m - n$
- Recursively defined function
  - Base case: n divides m, answer is n
  - Otherwise, reduce $gcd(m, n)$ to $gcd(n, m - n)$

```
def gcd(m,n):
  (a,b) = (max(m,n), min(m,n))
  if a%b == 0:
    return(b)
  else
    return(gcd(b,a-b))
```

Computing gcd

- Unfortunately, this takes time proportional to $\max(m, n)$

```
def gcd(m,n):
  (a,b) = (max(m,n), min(m,n))
  if a%b == 0:
    return(b)
  else
    return(gcd(b,a-b))
```

Computing gcd

- Unfortunately, this takes time proportional to $\max(m, n)$
- Consider `gcd(2,9999)`
  - $\rightarrow$ `gcd(2,9997)`
  - $\rightarrow$ `gcd(2,9995)`
  - ...
  - $\rightarrow$ `gcd(2,3)`
  - $\rightarrow$ `gcd(2,1)`
  - $\rightarrow$ 1

```
def gcd(m,n):
  (a,b) = (max(m,n), min(m,n))
  if a%b == 0:
    return(b)
  else
    return(gcd(b,a-b))
```

Computing gcd

- Unfortunately, this takes time proportional to $\max(m, n)$
- Consider `gcd(2,9999)`
  - $\rightarrow$ `gcd(2,9997)`
  - $\rightarrow$ `gcd(2,9995)`
  - ...
  - $\rightarrow$ `gcd(2,3)`
  - $\rightarrow$ `gcd(2,1)`
  - $\rightarrow$ 1
- Approximately 5000 steps

```
def gcd(m,n):
  (a,b) = (max(m,n), min(m,n))
  if a%b == 0:
    return(b)
  else
    return(gcd(b,a-b))
```

Computing gcd

- Unfortunately, this takes time proportional to $\max(m, n)$
- Consider `gcd(2,9999)`
  - $\rightarrow$ `gcd(2,9997)`
  - $\rightarrow$ `gcd(2,9995)`
  - ...
  - $\rightarrow$ `gcd(2,3)`
  - $\rightarrow$ `gcd(2,1)`
  - $\rightarrow$ 1
- Approximately 5000 steps
- Can we do better?

```
def gcd(m,n):
  (a,b) = (max(m,n), min(m,n))
  if a%b == 0:
    return(b)
  else
    return(gcd(b,a-b))
```

Euclid's algorithm

- Suppose n does not divide m

Euclid's algorithm

- Suppose n does not divide m
- Then $m = qn + r$

Euclid's algorithm

- Suppose n does not divide m
- Then $m = qn + r$
- Suppose d divides both m and n

Euclid's algorithm

- Suppose n does not divide m
- Then $m = qn + r$
- Suppose d divides both m and n
- Then $m = ad$, $n = bd$

Euclid's algorithm

- Suppose n does not divide m
- Then $m = qn + r$
- Suppose d divides both m and n
- Then $m = ad$, $n = bd$
- $m = qn + r \rightarrow ad = q(bd) + r$

Euclid's algorithm

- Suppose n does not divide m
- Then $m = qn + r$
- Suppose d divides both m and n
- Then $m = ad$, $n = bd$
- $m = qn + r \rightarrow ad = q(bd) + r$
- r must be of the form cd

Euclid's algorithm

- Suppose n does not divide m
- Then $m = qn + r$
- Suppose d divides both m and n
- Then $m = ad$, $n = bd$
- $m = qn + r \rightarrow ad = q(bd) + r$
- r must be of the form cd
- Euclid's algorithm
  - If n divides m, $\gcd(m, n) = n$
  - Otherwise, compute $\gcd(n, m \bmod n)$

```
def gcd(m,n):
  (a,b) = (max(m,n), min(m,n))
  if a%b == 0:
    return(b)
  else
    return(gcd(b,a%b))
```

Euclid's algorithm

- Suppose n does not divide m
- Then $m = qn + r$
- Suppose d divides both m and n
- Then $m = ad$, $n = bd$
- $m = qn + r \to ad = q(bd) + r$
- r must be of the form cd
- Euclid's algorithm
  - If n divides m, $\text{gcd}(m, n) = n$
  - Otherwise, compute $\text{gcd}(n, m \bmod n)$

```
def gcd(m,n):
  (a,b) = (max(m,n), min(m,n))
  if a%b == 0:
    return(b)
  else
    return(gcd(b,a%b))
```

- Can show that this takes time proportional to number of digits in $\max(m, n)$

Euclid's algorithm

- Suppose n does not divide m
- Then $m = qn + r$
- Suppose d divides both m and n
- Then $m = ad$, $n = bd$
- $m = qn + r \rightarrow ad = q(bd) + r$
- r must be of the form cd
- Euclid's algorithm
  - If n divides m, $\gcd(m, n) = n$
  - Otherwise, compute $\gcd(n, m \bmod n)$

```
def gcd(m,n):
  (a,b) = (max(m,n), min(m,n))
  if a%b == 0:
    return(b)
  else
    return(gcd(b,a%b))
```

- Can show that this takes time proportional to number of digits in $\max(m, n)$
- One of the first non-trivial algorithms