(Time) Complexities

(時間) 複雜度們

2023 (C) 黃敦紀 Elton

January 4, 2024

```python
n = int (input ()) # 總共有幾個數字
look = int (input ())
seq = list (map (int, input().split()))

i = 0
while i < n:
    if seq [i] == look:
        break
    else:
        i = i + 1

print (-1 if i == n - 1 and seq [-1] != look else i)
```

O (?)

```
print (123)
```

O (?)

```
total, start, npass = map (int, input().split())
print ((start + npass) % total)
```

O (?)

```
n = int (input ())

if n >= 60:
    print ("及格")
elif n >= 55:
    print ("勉强及格")
elif n >= 40:
    print ("補考")
else:
    print ("直接重補修")
```

O (?)

```
n = int (input ())
acc = 0

while n > 0:
    acc = acc + n
    n = n - 1

print (acc)
```

O (?)

```
n = int (input ())
factorial = 1

while n > 0:
    factorial = factorial * n
    n = n - 1

print (factorial)
```

O (?)

```
n = int (input ())
acc = 0

while n > 0:
    acc = acc + n
    n = n - 1

print (acc)
```

```
n = int (input ())
factorial = 1

while n > 0:
    factorial = factorial * n
    n = n - 1

print (factorial)
```

```
a1, b1, c1 = map (int, input().split())
a2, b2, c2 = map (int, input().split())
n = int (input ())

mx = -9999999999
x1 = 0

while x1 <= n:
    y1 = a1 * x1 * x1 + b1 * x1 + c1
    x2 = n - x1
    y2 = a2 * x2 * x2 + b2 * x2 + c2
    if mx < y1 + y2:
        mx = y1 + y2
    x1 = x1 + 1

print (mx)
```

O (?)

```
n, r = map (int, input().split())

pnr = 1

i = 0
while i < r:
    pnr = pnr * (n - i)
    i = i + 1

print(pnr)
```

O (?)

```
m, n = map (int, input().split())

i = 1
while i <= m:
    j = 1
    while j <= n:
        print (i * j)
        j = j + 1
    i = i + 1
```

O (?)

```
n = int (input ())

i = 1
while i <= n:
    j = 1
    while j <= n:
        pass # or do something
        j = j + 1
    i = i + 1
```

O (?)

```python
m, n = map (int, input().split())

i = 1
while i <= m:
    j = 1
    while j <= n:
        print (i * j)
        j = j + 1
    i = i + 1
```

```python
n = int (input ())

i = 1
while i <= n:
    j = 1
    while j <= n:
        pass # or do something
        j = j + 1
    i = i + 1
```

```python
n = int (input())

while n % 2 == 0:
    n = n // 2

print (n)
```

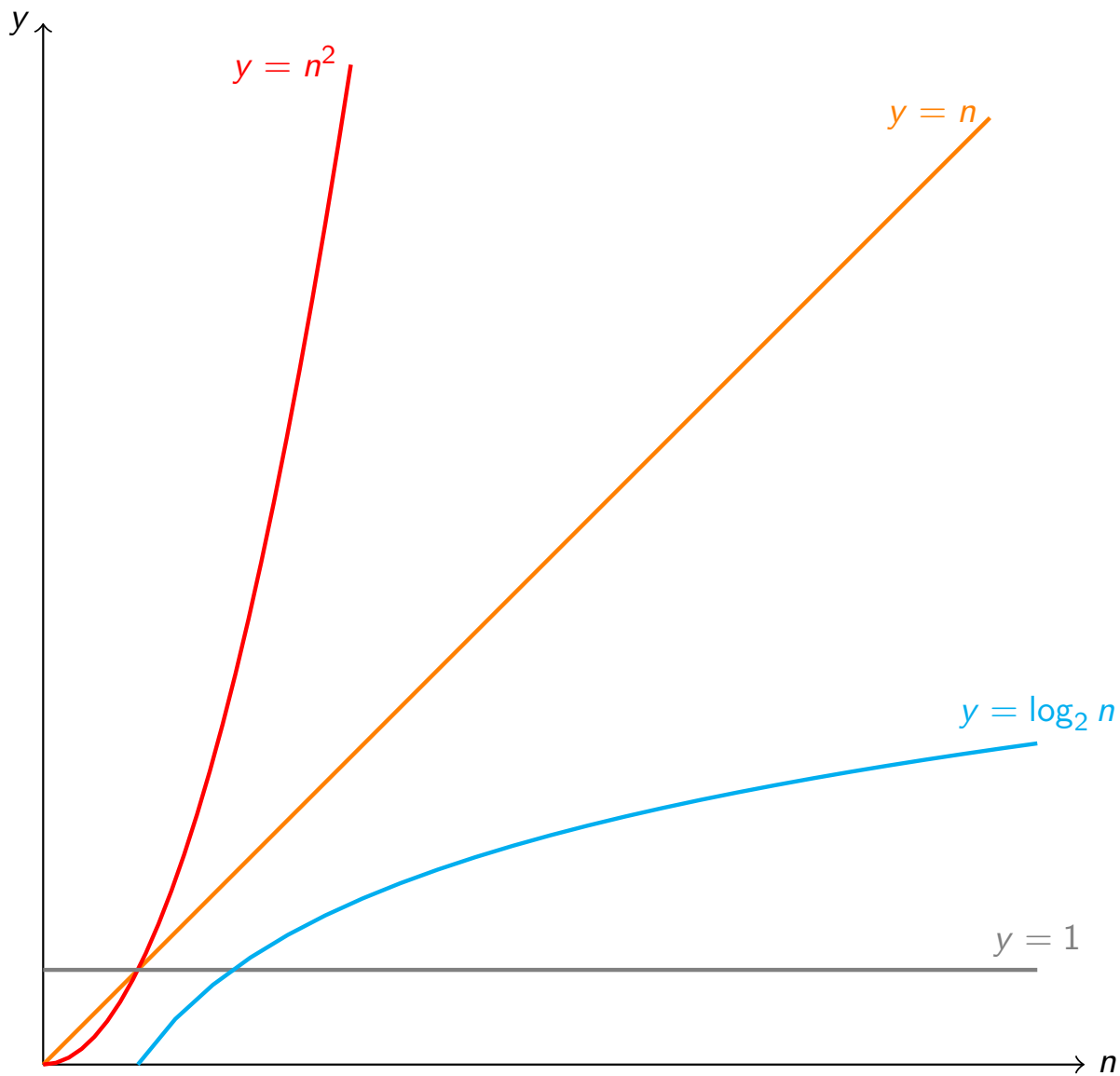
O (?)

```python
n = int (input())
i = 1
c = 0
while i < n:
    c = c + 1
    i = i * 2

print (i, c)
```

O (?)

```python
n = int (input())


while n % 2 == 0:

    n = n // 2

print (n)
```

```python
n = int (input())
i = 1
c = 0
while i < n:
    c = c + 1
    i = i * 2

print (i, c)
```


y
$y = n^2$
$y = n$
$y = \log_2 n$
$y = 1$
n

```
x = -r
on = True
while x <= r and on:
  y = -r
  while y <= r and on:
    if a1*x + b1*y == c1 and \
       a2*x + b2*y == c2:
      on = False
    else:
      y = y + 1
  x = x + 1
print (x - 1, y)
```

O (?)

```
x = (c1*b2-c2*b1)//(a1*b2-a2*b1)
y = (c1*a2-c2*a1)//(b1*a2-b2*a1)
print (x, y)









#
```

O (?)

參考：for/break 版

```
x = -r
found = False
for x in range (-r, r + 1):
  for y in range (-r, r + 1):
    if a1*x + b1*y == c1 and \
       a2*x + b2*y == c2:
      found = True
      break
  if found:
    break
print (x, y)
```

O (?)

```
x = (c1*b2-c2*b1)//(a1*b2-a2*b1)
y = (c1*a2-c2*a1)//(b1*a2-b2*a1)
print (x, y)









#
```

O (?)

```
n = int (input())

c = 0
i = 0
while i < n:
    j = 1
    while j < n:
        c = c + 1
        j = j * 2
    i = i + 1

print (c)
```

O (?)

```
n = int (input())

c = 0
i = 0
while i < n:
    j = 1
    while j < n:
        c = c + 1
        j = j + 1
    i = i * 2

print (c)
```

O (?)

```python
n = int (input())

c = 0
i = 0
while i < n:
    j = 1
    while j < n:
        c = c + 1
        j = j * 2
    i = i + 1

print (c)
```

```python
n = int (input())

c = 0
i = 0
while i < n:
    j = 1
    while j < n:
        c = c + 1
        j = j + 1
    i = i * 2

print (c)
```

```
...
s.sort ()

while n > 1:
    s[1] = s[0] + s[1]
    s.pop (0) # O(1)
    s.sort () # O(n*log(n))




    n = n - 1
...
```

O (?)

```
...
s.sort ()
i = 0
while i < n - 1:
  s[i+1] = s[i] + s[i+1]
  j = i + 1
  while j < n - 1:
    if s[j] > s[j+1]:
      s[j],s[j+1] = s[j+1],s[j]
      j = j + 1
    else:
      break
  i = i + 1
...
```

O (?)

```
n = int (input ()) # 總共有幾個數字
look = int (input ())
seq = list (map (int, input().split()))

i = 0
while i < n:
    if seq [i] == look:
        break
    else:
        i = i + 1

print (-1 if i == n - 1 and seq [-1] != look else i)
```

O (?)

```python
n = int (input ())  # 總共有幾個數字
look = int (input ())
seq = list (map (int, input().split()))

left = 0
rite = n - 1

while rite >= left:
    mid = (left + rite) // 2
    if seq [mid] == look:
        break
    elif seq [mid] < look:     # look 會在 seq [mid] 的右邊
        left = mid + 1
    else: # seq [mid] > look, # look 會在 seq [mid] 的左邊
        rite = mid - 1

print (mid if rite >= left else -1)
```

O (?)

```
n = int (input ())
look = int (input ())
seq = list (map (int, ...))

i = 0


while i < n:

    if seq [i] == look:
        break
    else:
        i = i + 1



print (-1 if ... else i)
```

```
n = int (input ())
look = int (input ())
seq = list (map (int, ...))

left = 0
rite = n - 1

while rite >= left:
    mid = (left + rite) // 2
    if seq [mid] == look:
        break
    elif seq [mid] < look:
        left = mid + 1
    else:
        rite = mid - 1

print (mid if ... else -1)
```

```
seq = list (map (int, ...))
# 不用一次性的前置作業

while True:
  try: # 每次查詢 O(n)
    look = input ("Look for? ")

    i = 0
    while i < n:

      if seq [i] == look:
        break
      else:
        i = i + 1


    print (-1 if ... else i)
  except EOFError:
    break
```

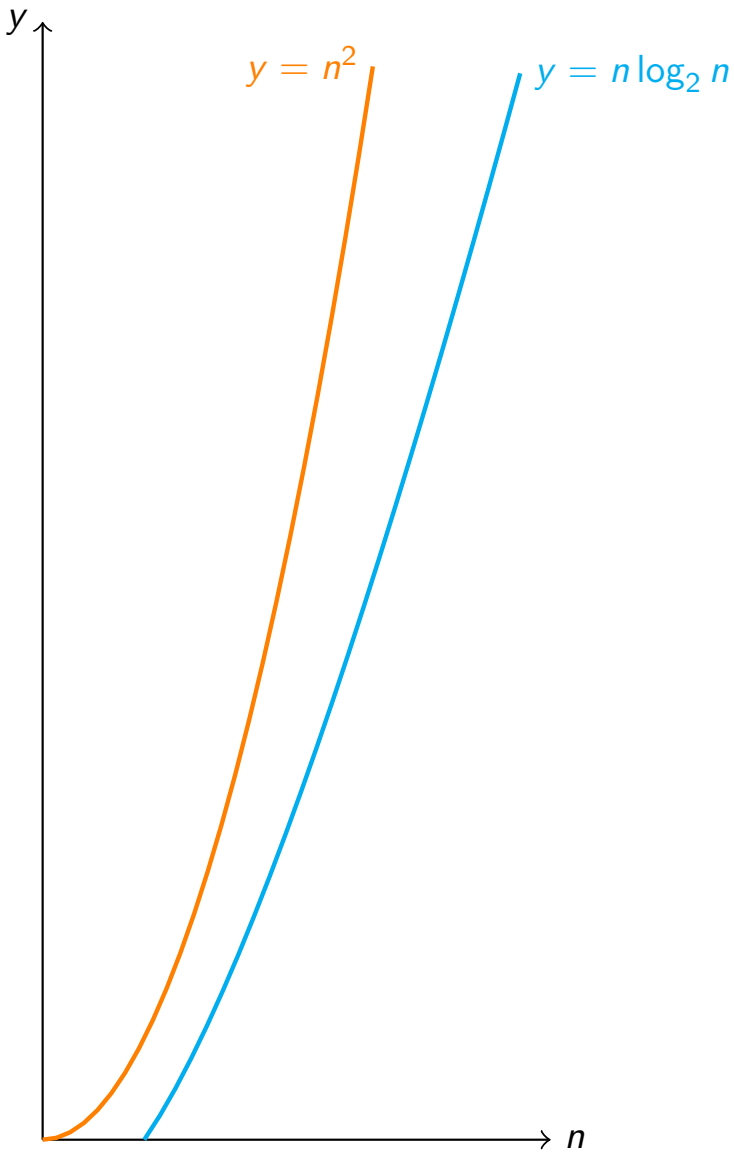
```
seq = list (map (int, ...))
sort (seq) # O(n*log(n))

while True:
  try: # 每次查詢 O(log(n))
    look = input ("Look for? ")
    left = 0
    rite = n - 1
    while rite >= left:
      mid = (left + rite) // 2
      if seq [mid] == look:
        break
      elif seq [mid] < look:
        left = mid + 1
      else:
        rite = mid - 1
    print (mid if ... else -1)
  except EOFError:
    break
```

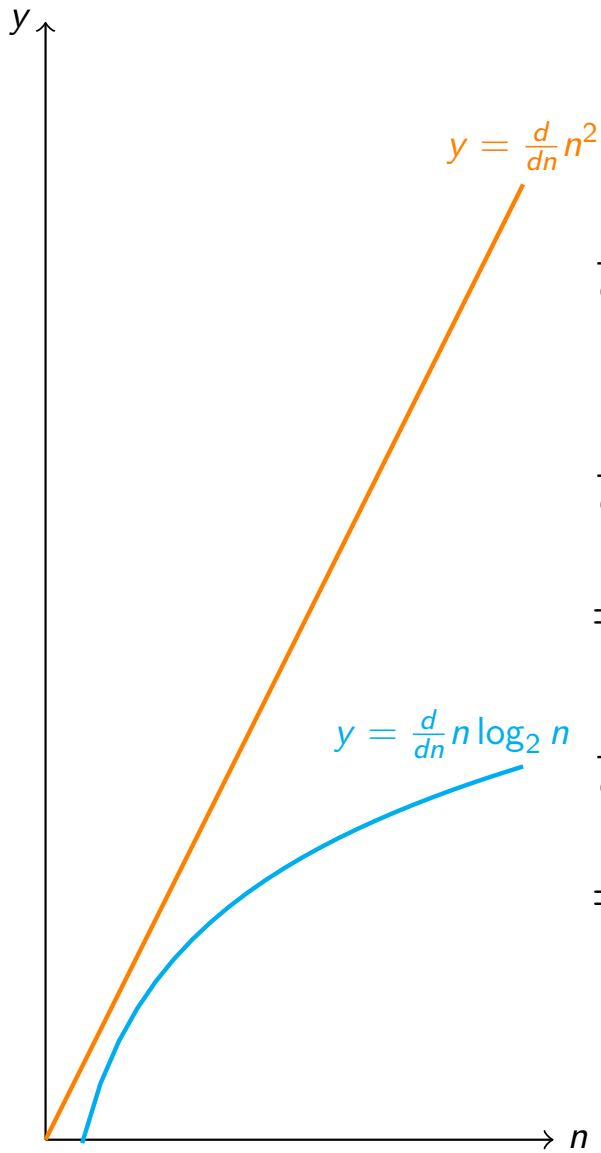
```
seq = list (map (int, ...))
# 不用一次性的前置作業

for _ in range (n):
    look = input ("Look for? ")

    i = 0
    while i < n:

        if seq [i] == look:
            break
        else:
            i = i + 1


    print (-1 if ... else i)
```

$O(n^2)$

```
seq = list (map (int, ...))
sort (seq) # O(n*log(n))

for _ in range (n):
    look = input ("Look for? ")
    left = 0
    rite = n - 1
    while rite >= left:
        mid = (left + rite) // 2
        if seq [mid] == look:
            break
        elif seq [mid] < look:
            left = mid + 1
        else:
            rite = mid - 1
    print (mid if ... else -1)
```

$O(n \times \log_2 n)$





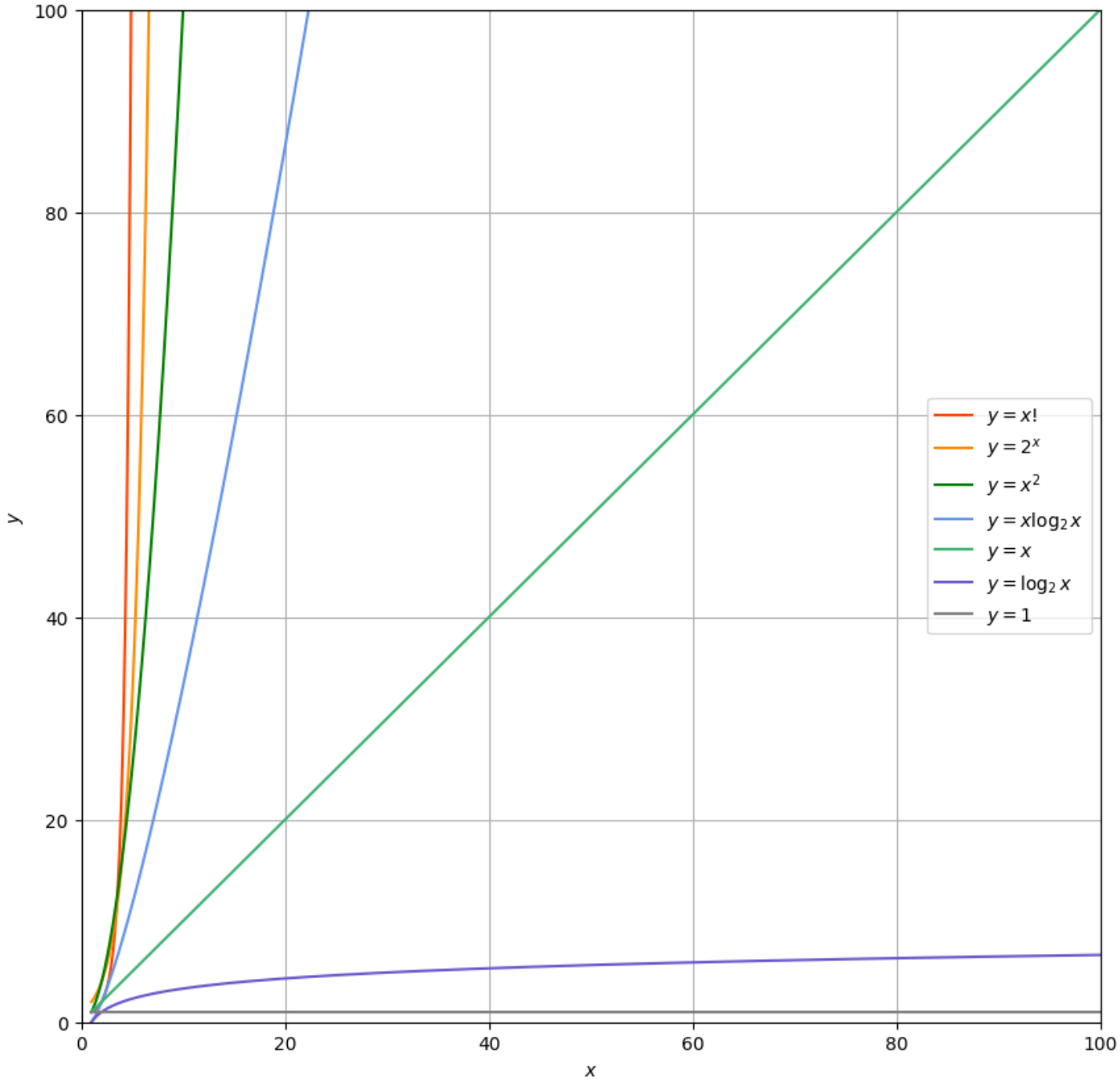
$$\frac{d}{dx} x^2 = 2x$$

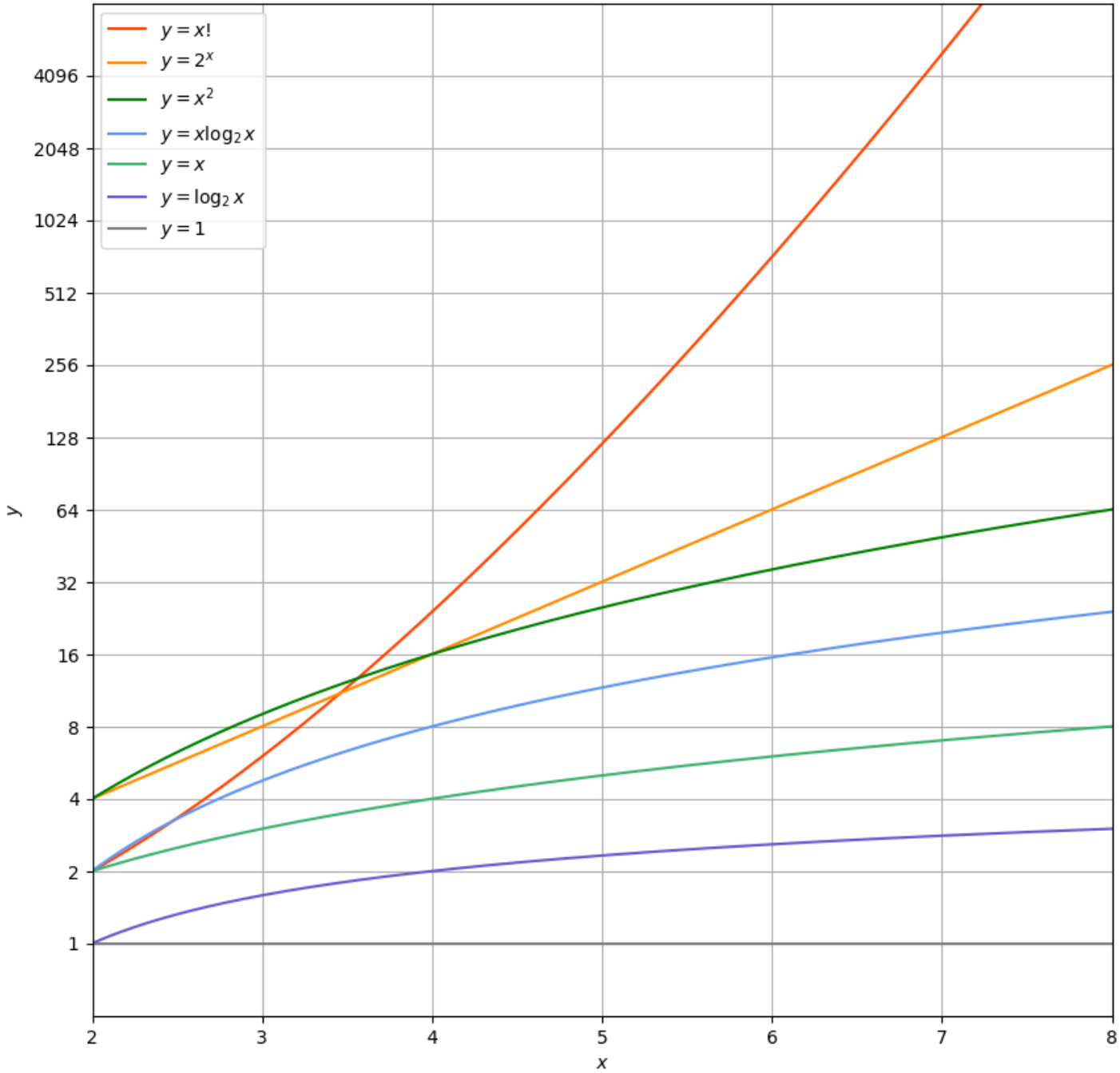
$$\frac{d}{dx} x log_2 x$$

$$= log_2 x + c$$

$$\frac{d^2}{dx^2} x log_2 x$$

$$= \frac{c}{x}$$


100
80
60
40
20
0
y
0
20
40
60
80
100
x
$y = x!$
$y = 2^x$
$y = x^2$
$y = x\log_2 x$
$y = x$
$y = \log_2 x$
$y = 1$


$y = x!$
$y = 2^x$
$y = x^2$
$y = x\log_2 x$
$y = x$
$y = \log_2 x$
$y = 1$
4096
2048
1024
512
256
128
64
32
16
8
4
2
1
y
2
3
4
5
6
7
8
x

```
look = int (input ())
wheres = list (map (int, input().split()))

print (wheres [look]) # 查表
```

O (?)

建表

```
n = int (input ()) # 總共有幾個數字
seq = list (map (int, input().split()))
# 假設 seq 中的數字為 1～99 不重複

wheres = [ -1 ] * 100

i = 0
while i < n:
    wheres [seq [i]] = i
```

O (?)

```
n = int (input ()) # 總共有幾個數字
seq = list (map (int, input().split()))
# 假設 seq 中的數字為 1～99 不重複

wheres = [ -1 ] * 100 # 犧牲空間複雜度

i = 0
while i < n: # 一次性作業 O (n)
    wheres [seq [i]] = i
    i = i + 1

while True:
    try: # 每次查詢 O(1)
        print (wheres [input ("Look for? ")])
    except EOFError:
        break
```

Click here for the Complexity of Python `in` operator

連續測資處理

```
#

a, b = map (int, input().split())
print (a + b)

#
```

```
while True:
  try:
    a, b = map (int, input().split(
    print (a + b)
  except EOFError:
    break
```

```cpp
#include <bits/stdc++.h>
using namespace std;

int main () {
    ios::sync_with_stdio (0); cin.tie (0); cout.tie (0);
    int a, b;
    while (cin >> a >> b) {
        cout << a + b << "\n";
    }
    return 0;
}
```

```cpp
    ...

    while (cin >> n) { // 每組測資的第一個 token 輸入
        ...
    }

    ...
```