Iterative time complexity tricks

(i) for nested loop, if the iterators are independent, figure out each loops time and multiply.

(ii) dependant → you can estimate the answer for the largest value of input which is independent, and try to figure out the dependent loop and multiply.

or, you can use summation of those two loop to get the answer.

⇒
```
x = n
while x>0:
    x- = x
```
$O(1)$

$\{x = x - x = 0\}$

⇒
```
y = n
while (y>0):
    y = y//2
```
$O(\log(n))$

⇒
```
i = 1
while i<n:
    i = i+i
```
$O(\log(n))$

$i = i+i \Rightarrow i = 2i \Rightarrow i* = 2$

⇒
```
x = 0
for (i=1; x<n; i++) {
    x = x+i}
```
$O(\sqrt{n})$

explanation :

for 1 iteration $\Rightarrow 1$.

2 " $\Rightarrow 1+2$

3 " $\Rightarrow 1+2+3$

k" $\Rightarrow 1+2+3 \dots k \Rightarrow \frac{k(k+1)}{2}$

for the loop to break in kth step, $\frac{k(k+1)}{2} \geq n$

$\Rightarrow k = \sqrt{n}$

$\Rightarrow$ for i in r(0, n, 2)

for j in r (0, i, 1):

for k in r (0, i*i, 1):

break

$\rightarrow O(N^2)$

max value of i will be $\rightarrow n$, so(n)

$n/2 \simeq O(N)$

1

if we use sum formula for dependant loop:

it 1 $\Rightarrow 1$

it 2 $\Rightarrow 1+2$

it 3 $\Rightarrow 1+2+3$

it n $\Rightarrow 1+2+3+\dots N$

$\Rightarrow \frac{n(n+1)}{2}$

so, outer two loop combines as $\frac{n(n+1)}{2}$ and the innermost loop (1) makes $\Rightarrow O\left(\frac{n(n+1)}{2} * 1\right) \Rightarrow O(n^2)$

divide = 1

```
for a in r(n, 0, -2):
    for b in r(a, 0, -1):
        for c in r(0, a+a, 2):
            d = d // (a/b)
        break
```

n n 1

same like the previous. $O(N^2)$

---

sum - 23 mid

by estimation

```
for i in r(1, n):
    j = 1
    while j < i*i:
        j += 1
```

n

N^2

the max value of i will be N, so it will go as high as $N*N = N^2$

$O(N^3)$

using summation:

it 1 ⇒ 1

it 2 ⇒ 1+4

it 3 ⇒ 1+4+9

it n ⇒ $1+4+9+\dots n^2$

⇒ $1^2+2^2+3^2+\dots n^2$

$$\frac{n(n+1)(2n+1)}{6}$$

$$\Rightarrow O(n^3)$$

```
for i in r(1,n):
    j = 1
    while j*j < i:
        j += 1
```

if $j^2 = i$; i will run at max n times. then j will run at ~~at~~ max $\sqrt{n}$ times $(\sqrt{n})$

n times

$O(n.\sqrt{n})$

Summation:

it 1 ⇒ 1

it 2 ⇒ 1+2

it 3 ⇒ 1+2+2

it 4 ⇒ 1+2+2+2

it 5 ⇒ 1+2+2+2+3

if you watch carefully,

for each n, its running $N^{\wedge 0.5}$ times. $[\sqrt{N} = N^{0.5}]$

the series is like:

$T(N) = 1^{0.5} + 2^{0.5} + 3^{0.5} + 4^{0.5} + 5^{0.5} + \dots N^{0.5}$

$\Rightarrow \sqrt{1} + \sqrt{2} + \sqrt{3} + \dots \sqrt{N} \Rightarrow$

$\Rightarrow$ there is no direct way of doing it, the integration result for this:

$$\int \sqrt{n}\, dn = \frac{2n\sqrt{n}}{3} + c$$

$$\Rightarrow 2/3 \left(n\sqrt{n}\right) + c$$

$$\text{so, } T(n) = \frac{2}{3} \cdot n^{3/2}$$

$$\Rightarrow O(n\sqrt{n})$$

---

tips: you can use multiplying methods since we are dealing with asymptotic notation, usually don't need the exact value. for example:

```
for i in r(1, n):   — loop 1
j = 1
while  j * j < i:   — loop 2
        j += 1
```

first, independent iterator i can go max n.

so, loop 1 = N, dependant iterator can go as high as $j*j < i$; since i can go n; $j = \sqrt{n}$; so loop 2 = $\sqrt{n}$. so, loop 1 * loop 2 $\Rightarrow O(N\sqrt{N})$