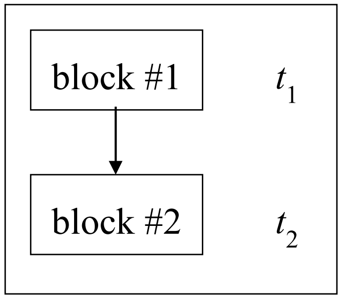
Analysis of Algorithms

Algorithm Control Structures

Algorithm Control Structures

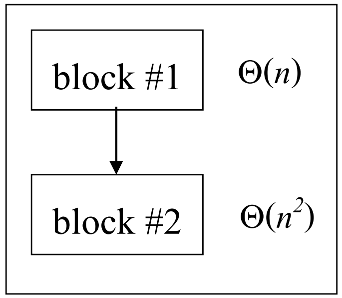
- Sequencing
- If-Then-Else
- "For" loop
- "While" loop
- Recursive calls

Sequencing



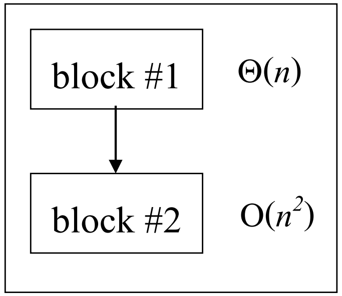
$t_1 + t_2 = \max(t_1,\ t_2)$

Sequencing



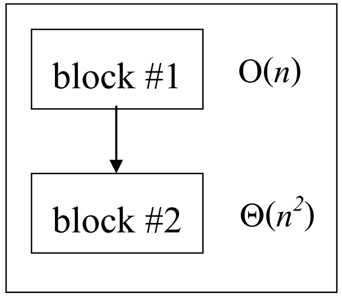
$\Theta(\ n^2\ )$

Sequencing



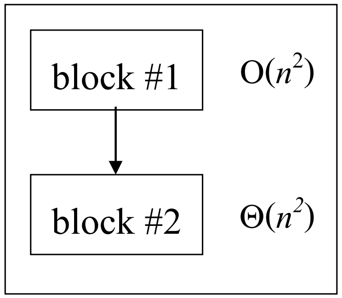
$O(n^2)$

Sequencing



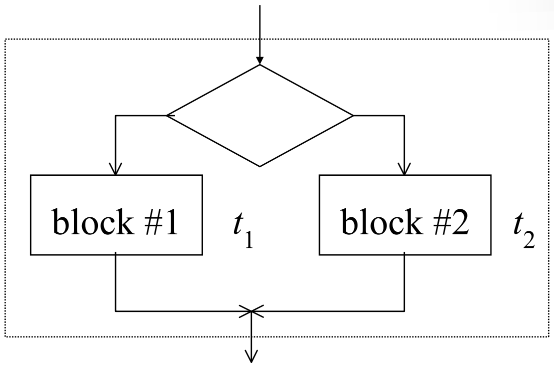
$\Theta(n^2)$

Sequencing



$\Theta(n^2)$

If-Then-Else



$\max(t_1, t_2)$

“For” Loop

```
for ( i = 1 to m )
{
  P(i)
}
```

$$\sum_{i=1}^{m} t_i$$

ให้ P(i) ใช้เวลา t_i

“For” Loop

```
for ( i = 1 to m )
  s += A[i][j]
```

$$\sum_{i=1}^{m} \Theta(1) = \Theta\left(\sum_{i=1}^{m} 1\right) = \Theta(m)$$

“For” Loop

```
for ( i = 1 to m )
  for ( j = 1 to m )
    s += A[i][j]
```

$$\sum_{i=1}^{m}\sum_{j=1}^{m} \Theta(1) = \sum_{i=1}^{m} \Theta(m) = \Theta\left(\sum_{i=1}^{m} m\right) = \Theta(m^2)$$

“For” Loop

```
for ( i = 1 to m )
  for ( j = 1 to i )
    s += A[i][j]
```

$$\sum_{i=1}^{m}\sum_{j=1}^{i} \Theta(1) = \sum_{i=1}^{m} \Theta(i) = \sum_{i=1}^{m} O(m) = O\left(\sum_{i=1}^{m} m\right) = O(m^2)$$

"For" Loop

```
for ( i = 1 to m )
  for ( j = 1 to i )
    s += A[i][j]
```

$$\sum_{i=1}^{m}\sum_{j=1}^{i}\Theta(1) = \sum_{i=1}^{m}\Theta(i)$$
$$= \Theta\left(\sum_{i=1}^{m} i\right)$$
$$= \Theta\left(\frac{m(m+1)}{2}\right)$$
$$= \Theta(m^2)$$

"For" Loop

```
for ( i = 2 to m-1 )
  for ( j = 3 to i )
    s += A[i][j]
```

$$\sum_{i=2}^{m-1}\sum_{j=3}^{i}\Theta(1) = \sum_{i=2}^{m-1}\Theta(i)$$
$$= \Theta\left(\sum_{i=2}^{m-1} i\right)$$
$$= \Theta\left(\frac{m^2}{2} + \Theta(m)\right)$$
$$= \Theta(m^2)$$

"While" Loop

```
while ( n > 0 ) {
   …
   n = n - 1
}
```

$\Theta(n)$

```
while ( n > 0 ) {
   …
   n = n / 2
}
```

$\Theta(\log n)$

```
i = 0; j = n
while ( i < j ) {
   …
   i++; j--;
}
```

$\Theta(n)$

"While" Loop: Insertion Sort

```
Insertion_Sort( A[1..n] )
  for j = 2 to n
  {
    key = A[j]
    i = j-1
    while i>0 and A[i]>key
    {
      A[i+1] = A[i]
      i = i-1
    }
    A[i+1] = key
  }
```

$O(i)$

“While” Loop: Insertion Sort

```
Insertion_Sort( A[1..n] )
  for j = 2 to n
  {
    key = A[j]
    i = j-1
    while i>0 and A[i]>key
    {
      A[i+1] = A[i]
      i = i-1
    }
    A[i+1] = key
  }
```

$O(j)$

“While” Loop: Insertion Sort

```
Insertion_Sort( A[1..n] )
  for j = 2 to n
  {
    key = A[j]
    i = j-1
    while i>0 and A[i]>key
    {
      A[i+1] = A[i]
      i = i-1
    }
    A[i+1] = key
  }
```

$O(n^2)$

“While” Loop: Euclid’s GCD

```
GCD( k, n )
{
  while  k > 0
  {
    t = k
    k = n mod k
    n = t
  }
  return n
}
```

t	k	n
	88	128
88	40	88
40	8	40
8	0	8

“While” Loop: Euclid’s GCD

```
GCD( k, n )
{
  while  k > 0
  {
    t = a
    k = n mod k
    n = t
  }
  return n
}
```

$O(\log n)$

ให้ $n \geq k$ จะพิสูจน์ว่า $n \bmod k < n/2$ เสมอ

1: ถ้า $k \leq n/2$

$n \bmod k < k \leq n/2$

2: ถ้า $k > n/2$

$n/k < 2$, $\text{int}(n/k) = 1$

$n \bmod k = n - k * \text{int}(n/k)$

$= n - k$

$< n - n/2 = n/2$

"While" Loop: FindMin_BST

```
Position FindMin_BST( TREE T )
{
  Position p;

  p = T;
  if ( p != NULL ) {
    while ( p->left != NULL ) {
      p = p->left;
    }
  }
  return p;
}
```

$O(h)$ $O(n)$

Recursive Calls

```
waste( n )
{
  if ( n = 0 ) return 0
  for ( i = 1 to n )
    for (j = 1 to i )
      print i,j,n
  for( i = 1 to 3 )
    waste( n/2 )
}
```

`if ( n = 0 ) return 0` → $\Theta(1)$

`for ( i = 1 to n ) for (j = 1 to i ) print i,j,n` → $\Theta(n^2)$

`for( i = 1 to 3 ) waste( n/2 )` → $3T(n/2)$

$$T(n) = 3T(n/2) + \Theta(n^2)$$

"Guessing"+ Induction

$$T(n) = T(0.7n) + T(0.2n) + O(n)$$

Guess: $T(n) = O(n)$, $T(n) \le cn$

Proof:

Basis: obvious

Induction: $T(n) \le 0.7cn + 0.2cn + O(n)$

$= 0.9cn + O(n)$

$\le 0.9cn + dn$

$= cn$ (choose $d = 0.1c$)

Theorem

If $\sum_{i=1}^{k} \alpha_i < 1$ then the solution to the recurrence

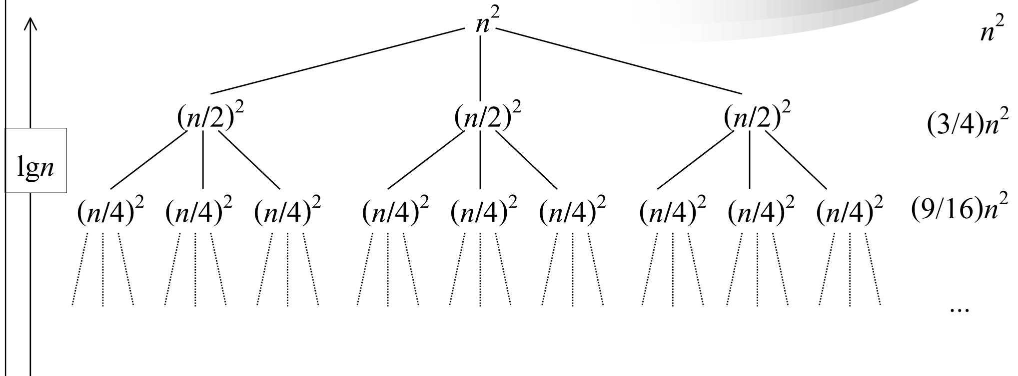
$$T(n) = \sum_{i=1}^{k} T(\alpha_i n) + O(n)$$

is $T(n) = O(n)$

$$T(n) = T(0.3n)+T(0.2n)+T(0.09n)+T(0.4n) + O(n)$$

$$= O(n)$$

Recursion Trees

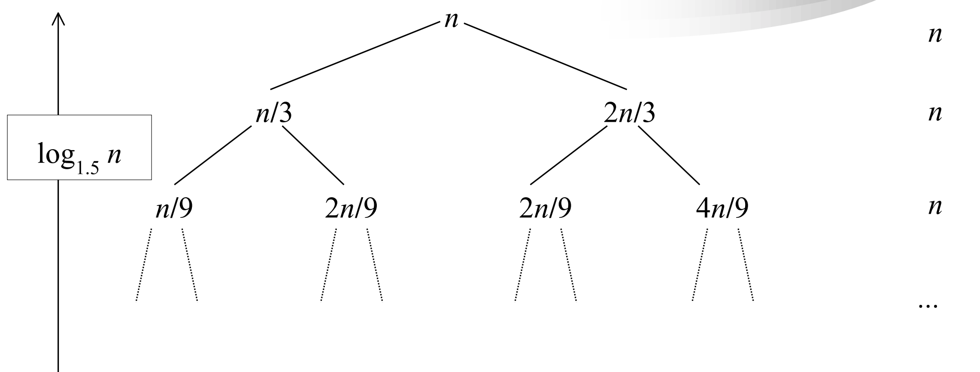


$$T(n) = 3T(\ n/2\ ) + \Theta(n^2\ )$$

$$T(n/2) = 3T(\ n/4\ ) + \Theta((n/2)^2\ )$$

$\Theta(n^2\ )$

Recursion Trees



$$T(n) = T(\ n/3\ ) + T(2n/\mathbf{3}) + n$$

$\Theta(n \log n\ )$

Master Method

- $T(n) = aT(n/b) + f(n) \qquad a \ge 1,\ b > 1$
- Let $c = \log_b a$
- If $f(n) = O(\ n^{c\ -\ \varepsilon}\ )$ for some constant $\varepsilon > 0$, then

  $T(n) = \Theta(\ n^c)$
- If $f(n) = \Theta(\ n^c\ )$ for some constant $\varepsilon > 0$, then

  $T(n) = \Theta(\ n^c \log n\ )$

Master Method

- $T(n) = aT(n/b) + f(n) \qquad a \ge 1,\ b \ge 1$
- Let $c = \log_b a$
- If $f(n) = \Omega(\ n^{c\ +\ \varepsilon}\ )$ for some constant $\varepsilon > 0$

  and if $a\,f(n/\mathrm{b}) \le d\,f(n)$ for some constant $d < 1$

  and all sufficiently large n, then
- $T(n) = \Theta(\ f(n))$

Master Method

- $T(n) = aT(n/b) + f(n) \qquad a \geq 1, b > 1$
- Let $c = \log_b a$
- $f(n) = O(n^{c - \varepsilon}) \quad \rightarrow \quad T(n) = \Theta(n^c)$
- $f(n) = \Theta(n^c) \quad \rightarrow \quad T(n) = \Theta(n^c \log n)$
- $f(n) = \Omega(n^{c + \varepsilon}) \quad \rightarrow \quad T(n) = \Theta(f(n))$

Master Method : Example

- $T(n) = 9T(n/3) + n$
- $a = 9, b = 3, c = \log_b a = 2, \quad n^c = n^2$
- $f(n) = n = O(n^{2 - 0.1})$
- $T(n) = \Theta(n^c) = \Theta(n^2)$

Master Method : Example

- $T(n) = T(n/3) + 1$
- $a = 1, b = 3, c = \log_b a = 0, \quad n^c = 1$
- $f(n) = 1 = \Theta(n^c) = \Theta(1)$
- $T(n) = \Theta(n^c \log n) = \Theta(\log n)$

Master Method : Example

- $T(n) = 3T(n/4) + n \log n$
- $a = 3, b = 4, c = \log_b a < 0.793, \quad n^c < n^{0.793}$
- $f(n) = n \log n = \Omega(n^{0.793})$
- $a f(n/b) = 3 ((n/4) \log (n/4)) \leq (3/4) n \log n = d f(n)$
- $T(n) = \Theta(f(n)) = \Theta(n \log n)$

Conclusion

- แบ่งอัลกอริทึมออกเป็น blocks ตาม control structures
- วิเคราะห์แต่ละ block
- วิเคราะห์ความสัมพันธ์ของ blocks ตาม control structure
- จะง่ายขึ้นถ้าวิเคราะห์แบบ asymptotic