

Recursive Runtime Trick

Master Theorem. Never use telescoping again.

Julian Lotzer, Daniel Steinhauser

Website: n.ethz.ch/~jlotzer

Step 1: Rewrite the recursive Function $T(n)$ into following mathematical form:

$$T(n) = \underbrace{a \cdot T\left(\frac{n}{b}\right)}_{\text{recursive call(s)}} + \underbrace{\theta(n^k \cdot \log^p(n))}_{\substack{\text{additional runtime in each call of } T(n), \\ \text{(example: for-loop)} \\ \text{Tipp: determine } \theta() \text{ directly and do} \\ \text{Koeffizientenvergleich with the} \\ \text{Formula above}}}$$

Step 2: Runtime is one of these cases:

1. if $a > b^k$, then $T(n) = \theta(n^{\log_b(a)})$
2. if $a = b^k$, and
 - a) if $p > -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log^{p+1}(n))$
 - b) if $p = -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log(\log(n)))$
 - c) if $p < -1$, then $T(n) = \theta(n^{\log_b(a)})$
3. if $a < b^k$, and
 - a) if $p \geq 0$, then $T(n) = \theta(n^k \cdot \log^p(n))$
 - b) if $p < 0$, then $T(n) = \theta(n^k)$

Example 1:

How often is $f()$ called given n ?

```
#pre: n is a positive integer
def T(n):
    if n >= 1:
        f()
        T(n//2)
```

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \theta(n^k \cdot \log^p(n))$$

1. if $a > b^k$, then $T(n) = \theta(n^{\log_b(a)})$
2. if $a = b^k$, and
 - a) if $p > -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log^{p+1}(n))$
 - b) if $p = -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log(\log(n)))$
 - c) if $p < -1$, then $T(n) = \theta(n^{\log_b(a)})$
3. if $a < b^k$, and
 - a) if $p \geq 0$, then $T(n) = \theta(n^k \cdot \log^p(n))$
 - b) if $p < 0$, then $T(n) = \theta(n^k)$

Example 1:

How often is $f()$ called given n ?

```
#pre: n is a positive integer
def T(n):
    if n >= 1:
        f()
        T(n//2)
```

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \theta(n^k \cdot \log^p(n))$$

1. if $a > b^k$, then $T(n) = \theta(n^{\log_b(a)})$
2. if $a = b^k$, and
 - a) if $p > -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log^{p+1}(n))$
 - b) if $p = -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log(\log(n)))$
 - c) if $p < -1$, then $T(n) = \theta(n^{\log_b(a)})$
3. if $a < b^k$, and
 - a) if $p \geq 0$, then $T(n) = \theta(n^k \cdot \log^p(n))$
 - b) if $p < 0$, then $T(n) = \theta(n^k)$

- **Step 1:** $T(n) = 1 \cdot T\left(\frac{n}{2}\right) + \theta(1)$ ($\theta(1)$ because $f()$ is only called once per $T(n)$ call)
Koeffizientenvergleich: $T(n) = 1 \cdot T\left(\frac{n}{2}\right) + \theta(n^0 \cdot \log^0(n))$ $a=1, b=2, k=0, p=0$
- **Step 2:** $1 = 2^0$ and $0 > -1$, so we have Case 2a):
 $\theta(n^{\log_2(1)} \cdot \log^{0+1}(n)) = \theta(n^0 \cdot \log^1(n)) = \theta(\log(n))$

Example 2:

How often is $f()$ called given n ?

```
#pre: n is a positive integer
def T(n):
    if n >= 1:
        for i in range(n):
            f()
        T(n//2)
```

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \theta(n^k \cdot \log^p(n))$$

1. if $a > b^k$, then $T(n) = \theta(n^{\log_b(a)})$
2. if $a = b^k$, and
 - a) if $p > -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log^{p+1}(n))$
 - b) if $p = -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log(\log(n)))$
 - c) if $p < -1$, then $T(n) = \theta(n^{\log_b(a)})$
3. if $a < b^k$, and
 - a) if $p \geq 0$, then $T(n) = \theta(n^k \cdot \log^p(n))$
 - b) if $p < 0$, then $T(n) = \theta(n^k)$

Example 2:

How often is $f()$ called given n ?

```
#pre: n is a positive integer
def T(n):
    if n >= 1:
        for i in range(n):
            f()
        T(n//2)
```

- **Step 1:** $T(n) = 1 \cdot T\left(\frac{n}{2}\right) + \theta(n)$ ($\theta(n)$ because $f()$ is called n times per $T(n)$ call)
Koeffizientenvergleich: $T(n) = 1 \cdot T\left(\frac{n}{2}\right) + \theta(n^1 \cdot \log^0(n))$ $a=1, b=2, k=1, p=0$
- **Step 2:** $1 < 2^1$ and $0 = 0$, so we have Case 3a):
 $\theta(n^1 \cdot \log^0(n)) = \theta(n)$

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \theta(n^k \cdot \log^p(n))$$

1. if $a > b^k$, then $T(n) = \theta(n^{\log_b(a)})$
2. if $a = b^k$, and
 - a) if $p > -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log^{p+1}(n))$
 - b) if $p = -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log(\log(n)))$
 - c) if $p < -1$, then $T(n) = \theta(n^{\log_b(a)})$
3. if $a < b^k$, and
 - a) if $p \geq 0$, then $T(n) = \theta(n^k \cdot \log^p(n))$
 - b) if $p < 0$, then $T(n) = \theta(n^k)$

Example 3:

How often is $f()$ called given n ?

```
#pre: n is a positive integer
def T(n):
    if n >= 1:
        for i in range(n):
            f()
            T(n//2)
            T(n//2)
```

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \theta(n^k \cdot \log^p(n))$$

1. if $a > b^k$, then $T(n) = \theta(n^{\log_b(a)})$
2. if $a = b^k$, and
 - a) if $p > -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log^{p+1}(n))$
 - b) if $p = -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log(\log(n)))$
 - c) if $p < -1$, then $T(n) = \theta(n^{\log_b(a)})$
3. if $a < b^k$, and
 - a) if $p \geq 0$, then $T(n) = \theta(n^k \cdot \log^p(n))$
 - b) if $p < 0$, then $T(n) = \theta(n^k)$

Example 3:

How often is $f()$ called given n ?

```
#pre: n is a positive integer
def T(n):
    if n >= 1:
        for i in range(n):
            f()
            T(n//2)
            T(n//2)
```

- **Step 1:** $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \theta(n)$ ($\theta(n)$ because $f()$ is called n times per $T(n)$ call)
Koeffizientenvergleich: $T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \theta(n^1 \cdot \log^0(n))$ $a=2, b=2, k=1, p=0$
- **Step 2:** $2 = 2^1$ and $0 > -1$, so we have Case 2a):
 $\theta(n^{\log_2(2)} \cdot \log^{0+1}(n)) = \theta(n^1 \cdot \log^1(n)) = \theta(n \cdot \log(n))$

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + \theta(n^k \cdot \log^p(n))$$

1. if $a > b^k$, then $T(n) = \theta(n^{\log_b(a)})$
2. if $a = b^k$, and
 - a) if $p > -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log^{p+1}(n))$
 - b) if $p = -1$, then $T(n) = \theta(n^{\log_b(a)} \cdot \log(\log(n)))$
 - c) if $p < -1$, then $T(n) = \theta(n^{\log_b(a)})$
3. if $a < b^k$, and
 - a) if $p \geq 0$, then $T(n) = \theta(n^k \cdot \log^p(n))$
 - b) if $p < 0$, then $T(n) = \theta(n^k)$