Running Time of a Recursive Function

```
// n >= 0
public static int powerOf2A( int n )
{
  if( n == 0 )
     return 1;
  else
     return powerOf2A( n – 1 ) + powerOf2A( n – 1);
}
```

Running Time of a Recursive Function

```
// n >= 0
public static int powerOf2B( int n )
{
  if( n == 0 )
     return 1;
  else
     return 2 * powerOf2B( n – 1 );
}
```

Derivation

- TA(n) = running time of powerOf2A
- TB(n) = running time of powerOf2B

Derivation

- $TA(n) = C + TA(n-1) + TA(n-1)$
- $TA(n) = 2\,TA(n-1) + C$

- $TA(n) = 2\,(2\,TA(n-2) + C) + C$

- $TA(n) = 2^2\,TA(n-2) + 2C + C$

Derivation

- $TA(n) = 2\,TA(n-1) + C$
- $TA(n) = 2^2\,TA(n-2) + 2C + C$

- $TA(n) = 2^2\,(2\,TA(n-3) + C) + 2C + C$
- $TA(n) = 2^3\,TA(n-3) + 2^2\,C + 2C + C$

Derivation

- $TA(n) = 2\,TA(n-1) + C$
- $TA(n) = 2^3\,TA(n-3) + 2^2\,C + 2C + C$
- $TA(n) = 2^3\,(2\,TA(n-4) + C) + 2^2\,C + 2C + C$
- $TA(n) = 2^4\,TA(n-4) + 2^3\,C + 2^2\,C + 2C + C$

Derivation

- $TA(n) = 2\,TA(n-1) + C$
- $TA(n) = 2^2\,TA(n-2) + 2C + C$
- $TA(n) = 2^3\,TA(n-3) + 2^2\,C + 2C + C$
- $TA(n) = 2^4\,TA(n-4) + 2^3\,C + 2^2\,C + 2C + C$

- Do we have a pattern?

Derivation

- $TA(n) = 2^k TA(n - k) + 2^{k-1} C + \ldots + 2^3 C + 2^2 C + 2C + C$

- Base case when ?

Derivation

- $TA(n) = 2^k TA(n – k) + 2^{k-1} C + … + 2^3 C + 2^2 C + 2C + C$

- Base case when $n – k = 0$ i.e. $n = k$
- $TA(n) = 2^n TA(0) + 2^{n-1} C + … + 2^3 C + 2^2 C + 2C + C$

Derivation

- $TA(n) = 2^n TA(0) + 2^{n-1} C + \ldots + 2^3 C + 2^2 C + 2C + C$
- $TA(n) = 2^n TA(0) + C(2^{n-1} + \ldots + 2^3 + 2^2 + 2 + 1)$
- $TA(n) = 2^n TA(0) + C(???)$

Derivation

- $TA(n) = 2^n TA(0) + C(2^{n-1} + \ldots + 2^3 + 2^2 + 2 + 1)$
- $TA(n) = 2^n TA(0) + C (2^n - 1)$
- $TA(n)$ is $\Theta(2^n)$

Running Time of a Recursive Function

```
// n >= 0
public static int powerOf2B( int n )
{
  if( n == 0 )
     return 1;
  else
     return 2 * powerOf2B( n – 1 );
}
```

Derivation

- $TB(n) = D + TB(n – 1)$
- $TB(n) = TB(n – 1) + D$

- $TB(n) = TB(n – 2) + D + D$

- $TB(n) = TB(n – 2) + 2D$

Derivation

- $TB(n) = TB(n – 1) + D$
- $TB(n) = TB(n – 2) + 2D$

- $TB(n) = TB(n – 3) + D + 2D$
- $TB(n) = TB(n – 3) + 3D$

Derivation

- $TB(n) = TB(n - 1) + D$
- $TB(n) = TB(n - 3) + 3D$

- $TB(n) = TB(n - 4) + D + 3D$

- $TB(n) = TB(n - 4) + 4D$

Derivation

- $TB(n) = TB(n - 1) + D$ or
- $TB(n) = TB(n - 1) + 1D$
- $TB(n) = TB(n - 2) + 2D$
- $TB(n) = TB(n - 3) + 3D$
- $TB(n) = TB(n - 4) + 4D$

- Do we see a pattern?

Derivation

- $TB(n) = TB(n – k) + kD$
- We reach the base case
- when $n – k = 0$, i.e. $n = k$

- $TB(n) = TB(0) + Dn$
- $TB(n)$ is $\Theta(n)$

Derivation

- TA(n) is $\Theta(2^n)$
- TB(n) is $\Theta(n)$

- This confirms our earlier code simulation