

## Time complexity as recurrence

To analyze the time complexity of recursive procedures, we often have to deal with recurrence relations

```
PROC expo(u,v)
  IF v == 1 THEN RETURN u
  ELSEIF v is even THEN
    RETURN sq(expo(u,v DIV 2))
  ELSE RETURN u*sq(expo(u,v DIV 2))
END
END
PROC sq(x)
  RETURN x*x
END
```

Suppose we define  $T(n)$  as “the number of floating point multiplications performed by  $expo(x, n)$  for any  $x$ .”

We would like to be able to prove that this procedure is *very efficient*, and in fact that  $T(n) \in \mathcal{O}(\log_2(n))$ .

What do we know about  $T$ ?

We have that

$$T(n) = \left\{ \right.$$

What do we do with recurrence relations like this?