CSC236 fall 2014

correct after & before

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~csc236h/fall/
416-978-5899

Using Introduction to the Theory of Computation,
Chapter 2
Outline

power

notes

annotations
Master Theorem
(for divide-and-conquer recurrences)

If \( f \) from the previous slide has \( f \in \theta(n^d) \), then

\[
T(n) = \begin{cases} 
\theta(n^d) & \text{if } a < b^d \\
\theta(n^d \log n) & \text{if } a = b^d \\
\theta(n^{\log_b a}) & \text{if } a > b^d 
\end{cases}
\]
def pow(b, n, mod):
    if n == 0:
        return 1
    r = b if n % b == 1 else 1
    half_pow = pow(b, n//b, mod)
    return (half_pow * half_pow * r) % mod

def pow2(b, n, mod):
    if n == 0:
        return 1
    r = b if n % b == 1 else 1
    return (pow2(b, n//b, mod) * pow2(b, n//b, mod) * r) % mod

def pow3(b, n, mod):
    return b**n % mod

(see power.py)
iterative power...

```python
def power(x, y):
z = 1
m = 0
while m < y:
z = z * x
m = m + 1
return z
```

- precondition?
- postcondition?
- notation for mutation
partial correctness
precondition + execution + termination imply postcondition
a loop invariant helps get us closer
partial correctness
precondition + execution + termination imply postcondition
a loop invariant helps get us closer
prove partial correctness
prove termination
associate a decreasing sequence in \( \mathbb{N} \) with loop iterations
it helps to add claims to the loop invariant
put it together — correctness
correctness by design
draw pictures of before, during, after
pre: A sorted, comparable with x
post: 0 ≤ p ≤ n and A[0..p-1] < x ≤ A[p .. n-1]
“derive” conditions from pictures
do we have termination?
notes
annotated slides