



Math Review

CptS 223 – Advanced Data Structures

Larry Holder

School of Electrical Engineering and Computer Science
Washington State University



Why do we need math in a data structures course?

- Analyzing data structures and algorithms
 - Deriving formulae for time and memory requirements
 - Will the solution scale?
- Proving algorithm correctness



Floors and Ceilings

- *floor*(x), denoted $\lfloor x \rfloor$, is the greatest integer $\leq x$
- *ceiling*(x), denoted $\lceil x \rceil$, is the smallest integer $\geq x$
- Normally used to divide input into integral parts
$$\left\lfloor \frac{N}{2} \right\rfloor + \left\lceil \frac{N}{2} \right\rceil = N$$



Example

- Consider Algorithm1 that divides the input array in half and calls Algorithm2 on each half

```
Algorithm1 (A,n)
  // A is an integer array of size n
  x = floor(n/2)
  Algorithm2 (A,1,x)
  Algorithm2 (A,x+1,n)
```

- Assume Algorithm2(A,i,j)'s running time is proportional to (j-i+1)
- What is the running time of Algorithm1?



Exponents

$$X^A X^B = X^{A+B}$$

$$\frac{X^A}{X^B} = X^{A-B}$$

$$(X^A)^B = X^{AB}$$

$$X^N + X^N = 2X^N \neq X^{2N}$$

$$2^N + 2^N = 2^{N+1}$$



Logarithms

$$\log_x B = A \Leftrightarrow X^A = B \quad \text{"logarithm of B base X"}$$

$$\log_A B = \frac{\log_C B}{\log_C A}; \quad A, B, C > 0, A \neq 1$$

$$\log AB = \log A + \log B; \quad A, B > 0$$

$$\log \frac{A}{B} = \log A - \log B$$

$$\log A^B = B \log A$$

$$\log X < X \quad \text{for all } X > 0$$

$$\lg A = \log_2 A$$

$$\ln A = \log_e A; \quad e = 2.7182... \quad \text{"natural logarithm"}$$

In Weiss book,
 $\log A \rightarrow \log_2 A$



Example

- How many times to halve an array of length n until its length is 1?

```
Halve (n)
  i = 0
  while n ≠ 1
    i = i + 1
    n = floor(n/2)
  return i
```



Factorials

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n * (n-1)! & \text{if } n > 0 \end{cases}$$

$$n! < n^n$$

$$n! = \sqrt{2\pi n} (n/e)^n (1 + \theta(1/n)) \quad \text{Stirling's approximation}$$

```
PermutationSort (A,n)
// A is an integer array of length n
while A is not in order
    Permute (A)
```



Series

- **General** $\sum_{i=0}^N f(i) = f(0) + f(1) + \dots + f(N)$

- **Linearity** $\sum_{i=0}^N (cf(i) + g(i)) = c \sum_{i=0}^N f(i) + \sum_{i=0}^N g(i)$

- **Arithmetic series** $\sum_{i=1}^N i = \frac{N(N+1)}{2}$

Series

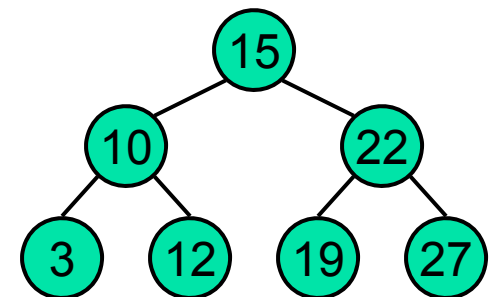
- Geometric series

$$\sum_{i=0}^N A^i = \frac{A^{N+1} - 1}{A - 1}$$

$$\sum_{i=0}^N A^i \leq \sum_{i=0}^{\infty} A^i = \frac{1}{1 - A}; \text{ if } 0 < A < 1$$

- Example

- How many nodes in a complete binary tree of depth D?





Modular Arithmetic

$A \bmod N = A - N * \lfloor A / N \rfloor$ "remainder"

$(A \bmod N) = (B \bmod N) \Rightarrow A \equiv B \pmod{N}$

"A is congruent to B modulo N"

E.g., $81 \equiv 61 \equiv 1 \pmod{10}$

If $A \equiv B \pmod{N}$

Then $A + C \equiv B + C \pmod{N}$

and $AD \equiv BD \pmod{N}$

Basis of most
encryption schemes:
(Message mod Key)



Proofs

- What do we want to prove?
 - Properties of a data structure always hold for all operations
 - Algorithm running time/memory will never exceed some limit
 - Algorithm will always be correct
 - Algorithm will always terminate
- Techniques
 - Proof by induction
 - Proof by counterexample
 - Proof by contradiction



Proof by Induction

- Goal: Prove some hypothesis is true
- Three-step process
 1. Base case: Show hypothesis is true for some initial conditions
 2. Inductive hypothesis: Assume hypothesis is true for all values $\leq k$
 3. Show hypothesis is true for next larger value (typically $k+1$)



Example

- Prove arithmetic series

$$\sum_{i=1}^N i = \frac{N(N+1)}{2}$$

- (Step 1) Base case: Show true for $N=1$

$$\sum_{i=1}^1 i = 1 = \frac{1(1+1)}{2}$$



Example (cont.)

- (Step 2) Assume true for $N=k$
- (Step 3) Show true for $N=k+1$

$$\begin{aligned}\sum_{i=1}^{k+1} i &= (k+1) + \sum_{i=1}^k i \\ &= (k+1) + \frac{k(k+1)}{2} \\ &= \frac{2(k+1) + k(k+1)}{2} \\ &= \frac{(k+1)(k+2)}{2}\end{aligned}$$



More Examples

- Prove the geometric series

$$\sum_{i=0}^N A^i = \frac{A^{N+1} - 1}{A - 1}$$

- Prove that the number of nodes N in a complete binary tree of depth D is $2^{D+1} - 1$

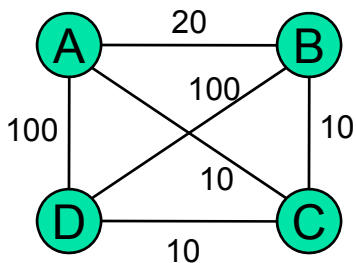


Proof by Counterexample

- Prove hypothesis is not true by giving an example that doesn't work
 - Example: $2^N > N^2$?
- Proof by example?
- Proof by lots of examples?
- Proof by all possible examples?
 - Empirical proof
 - Hard when input size and contents can vary arbitrarily

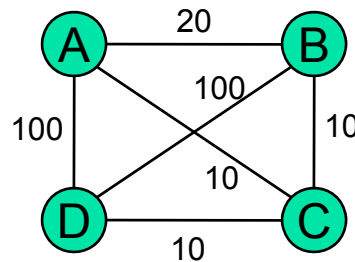
Another Example

- Traveling salesman problem
 - Given N cities and costs for traveling between each pair of cities, find the least-cost tour to visit each city exactly once
- Hypothesis
 - Given a least-cost tour for N cities, the same tour will be least-cost for $(N-1)$ cities
 - E.g., if $A \rightarrow B \rightarrow C \rightarrow D$ is the least-cost tour for cities $\{A, B, C, D\}$, then $A \rightarrow B \rightarrow C$ will be the least-cost tour for cities $\{A, B, C\}$



Another Example (cont.)

- Counterexample
 - Cost ($A \rightarrow B \rightarrow C \rightarrow D$) = 40 (optimal)
 - Cost ($A \rightarrow B \rightarrow C$) = 30
 - Cost ($A \rightarrow C \rightarrow B$) = 20





Proof by Contradiction

- Assume hypothesis is false
- Show this assumption leads to a contradiction (i.e., some known property is violated)
 - Can't use special cases or specific examples
- Therefore, hypothesis must be true

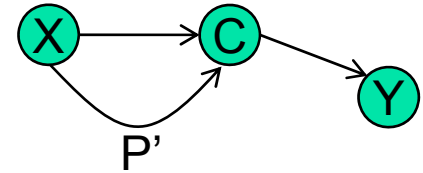
Example

- Variant of traveling salesman problem
 - Given N cities and costs for traveling between each pair of cities, find the least-cost path to go from city X to city Y
- Hypothesis
 - A least-cost path from X to Y contains least-cost paths from X to every city on the path
 - E.g., if $X \rightarrow C1 \rightarrow C2 \rightarrow C3 \rightarrow Y$ is the least-cost path from X to Y , then



- $X \rightarrow C1 \rightarrow C2 \rightarrow C3$ is the least-cost path from X to $C3$
- $X \rightarrow C1 \rightarrow C2$ is the least-cost path from X to $C2$
- $X \rightarrow C1$ is the least-cost path from X to $C1$

Example (cont.)



- Assume hypothesis is false
 - I.e., Given a least-cost path P from X to Y that goes through C , there is a better path P' from X to C than the one in P
- Show a contradiction
 - But we could replace the subpath from X to C in P with this lesser-cost path P'
 - The path cost from C to Y is the same
 - Thus we now have a better path from X to Y
 - But this violates the assumption that P is the least-cost path from X to Y
- Therefore, the original hypothesis must be true



Recursion

- A recursive function is defined in terms of itself
- Example: $n! = \begin{cases} 1 & \text{if } n = 0 \\ n * (n - 1)! & \text{if } n > 0 \end{cases}$

```
Factorial (n)
  if n = 0
  then return 1
  else return (n * Factorial (n-1))
```



Basic Rules of Recursion

- Base cases
 - Must always have some base cases, which can be solved without recursion
- Making progress
 - Recursive calls must always make progress toward a base case
- Design rule
 - Assume all recursive calls work
- Compound interest rule
 - Never duplicate work by solving the same instance of a problem in separate recursive calls

Example

■ Fibonacci numbers

- $F(0) = 1$
- $F(1) = 1$
- $F(n) = F(n-1) + F(n-2)$

```
Fibonacci (n)
  if (n ≤ 1)
    then return 1
  else return (Fibonacci (n-1) + Fibonacci (n-2))
```

Did you know?

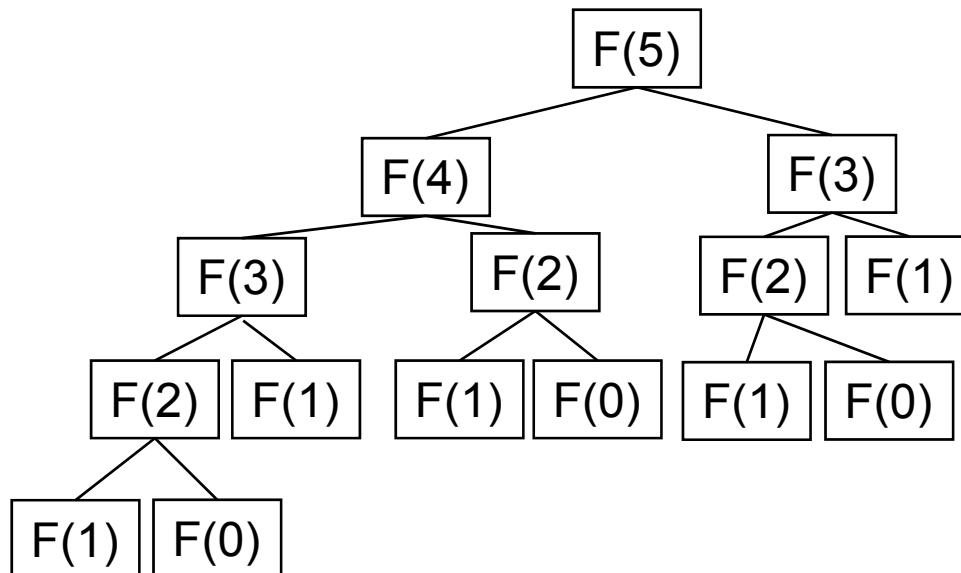


The number of petals on a flower is usually a Fibonacci number?
This daisy has $F(8)=34$ petals.



Example (cont.)

- Fibonacci (5)





Example (cont.)

- Show that the running time $T(n)$ of $\text{Fibonacci}(n)$ is exponential in n
- Use mathematical induction
 - Show $T(n) \leq c2^n$ for some positive constant c
 - Base case: $T(1) \leq c2^1 = 2c$ (true for some constant $c > 0$)
 - Inductive hypothesis: $T(k) \leq c2^k$
 - Show $T(k+1) \leq c2^{k+1}$
- Actually, only proved that $T(n)$ is no more than exponential
 - Need to also prove $T(n)$ is at least exponential



Summary

- Floors, ceilings, exponents, logarithms, series, and modular arithmetic
- Proofs by mathematical induction, counterexample and contradiction
- Recursion
- Tools to help us analyze the performance of our data structures and algorithms