CSE 373 Optional Section

Led by Yuanwei, Luyi
Apr 10 2014
Today

- Proof by Induction
- Big-Oh
- Algorithm Analysis
Proof by Induction

Base Case:
1. Prove $P(0)$ (sometimes $P(1)$)

Inductive Hypothesis:
2. Let $k$ be an arbitrary integer $\geq 0$
3. Assume that $P(k)$ is true

Inductive Step
4. have $P(k)$ is true, Prove $P(k+1)$ is true

Conclusion:
5. $P(n)$ is true for $n \geq 0$ (or 1...)
Examples

\[ \sum_{i=1}^{N} i^2 = 1 + 2^2 + 3^2 + 4^2 + \ldots + n^2 = \frac{N(N+1)(2N+1)}{6} \quad \text{for all } n \geq 1 \]

\[ \sum_{i=0}^{N} 2^i = 2^0 + 2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 1 \]

Extra

\[ \sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1} \quad \text{where } n \in \mathbb{Z}^+ \]
Logarithms

• log in CS means log base of 2
• log grows very slowly
• logAB=logA+logB; log(A/B)=logA-logB
• log(N^k)= k log N
  – Eg. Log(A^2) = log(A*A) = log A + log A = 2log A
• distinguish log(log x) and log^2x   --(log x)(log x)
Big-Oh

- We only look at worst case
- Big input
- Ignore constant factor and lower order terms
  - Why?
- Definition:
  
  \[ g(n) \text{ is in } O(f(n)) \text{ if there exist constants } c \text{ and } n_0 \]
  
  \[ \text{such that } g(n) \leq c \cdot f(n) \text{ for all } n \geq n_0 \]

- Also lower bound and tight bound

We use \( O \) on a function \( f(n) \) (for example \( n^2 \)) to mean the set of functions with asymptotic behavior less than or equal to \( f(n) \)
Big-Oh Practice

• Prove that $5n^2 + 3n$ is $O(n^2)$
  – Key point
    Find constant $c$ and $n_0$
Big-Oh Practice

• Prove that $5n^2 + 3n$ is $O(n^2)$
  – Key point
    Find constant $c$ and $n_0$

Possible $c$ and $n0$:

- $c = 8$ and $n0 = 1$
- $c = 6$ and $n0 = 3$
- ...

Math Related

- Series

\[ \sum_{i=1}^{N} A^i = A + A^2 + A^3 + A^4 + \cdots = \frac{A^{N+1} - 1}{A - 1} \]

\[ \sum_{i=1}^{N} i = 1 + 2 + 3 + 4 + \cdots = \frac{N(N+1)}{2} \approx \frac{N^2}{2} \]

\[ \sum_{i=1}^{N} i^2 = 1 + 2^2 + 3^2 + 4^2 + \cdots = \frac{N(N+1)(2N+1)}{6} \approx \frac{N^3}{3} \]

- Very useful for runtime analysis
- On your textbook, p4
How to analyze the code?

Consecutive statements: Sum of times
Conditionals: Time of test plus slower branch
Loops: Sum of iterations
Calls: Time of call’s body
Recursion: Solve recurrence equation
Examples

1. int sunny (int n) {
    if (n < 10)
        return n - 1;
    else {
        return sunny (n / 2);
    }
}

2. int funny (int n, int sum) {
    for (int k = 0; k < n * n; +
        +k)
        for (int j = 0; j < k; j+)
            sum++;
    return sum;
}

3. int happy (int n, int sum) {
    for (int k = n; k > 0; k = k - 1) {
        for (int i = 0; i < k; i++)
            sum++;
        for (int j = n; j > 0; j--)
            sum++;
    }
    return sum;
}
Examples

1. int sunny (int n) {
    if (n < 10)
        return n - 1;
    else {
        return sunny (n / 2);
    }
}

2. int funny (int n, int sum) {
    for (int k = 0; k < n * n; ++k)
        for (int j = 0; j < k; j++)
            sum++;
    return sum;
}

3. int happy (int n, int sum) {
    for (int k = n; k > 0; k = k - 1) {
        for (int i = 0; i < k; i++)
            sum++;
        for (int j = n; j > 0; j--)
            sum++;
    }
    return sum;
}

Answer:

1. O(logn)
2. O(n^4)
3. O(n^2)