CSCI 3401 Algorithms

recitation 1: introduction to algorithm notations and proofs

University of Colorado, Boulder, CO

Jon Z. Cai
[jon.z.cai@colorado.edu]

1 Asymptotic Analysis

1.1 Definition

We use $\mathcal{O}(*)$ notation to bound a function from above, which is all called **asymptotic upper bound**.

Definition 1 (big $\mathcal{O}$) $\mathcal{O}(g(n)) = \{f(n) \mid \exists c > 0, n_0 > 0 \ s.t. \ \forall n \geq n_0, 0 \leq f(n) \leq cg(n)\}$

Similarly, we use Ω to denote **asymptotic lower bound**.

Definition 2 (big Ω) $\Omega(g(n)) = \{f(n) \mid \exists c > 0, n_0 > 0 \ s.t. \ \forall n \geq n_0, 0 \leq cg(n) \leq f(n)\}$

If we can bound a function from both above and below, we got our **asymptotic tight bound** Θ.

Definition 3 (big Θ) $\Theta(g(n)) = \{f(n) \mid \exists c_1 > 0, c_2 > 0, n_0 > 0 \ s.t. \ \forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$

1.2 Examples

1. Prove the upper bound of $f(n) = n!$ can be $g(n) = n^n$ by definition.

2. Prove the lower bound of $f(n) = 2^n$ can be $g(n) = n^2$ by definition.

2 Proof Techniques

2.1 Proof by contradiction

In logic, proof by contradiction is a form of proof that establishes the truth or validity of a proposition by first assuming that the opposite proposition is true, and then shows that such an assumption leads to a contradiction General steps:

1. P is assumed to be false.
2. It is shown that $\neg P$ implies P.
3. Since P and $\neg P$ cannot both be true, the assumption must be wrong and P must be true.

2.2 Loop invariant

1. **Initialization**: It is true prior to the first iteration of the loop.
2. **Maintenance**: If it is true before an iteration of the loop, it remains true before the
3. **Termination**: When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

2.3 Solve Recurrences

- Substitution method
  1. Guess the form of the solution.
  2. Use mathematical induction to find the constants and show that the solution works.
- Recursion Tree method
- Master theorem: $T(n) = aT(n/b) + f(n)$

2.4 Examples

1. Prove: $\forall a, b \in \mathbb{Z},\ a^2 - 4b \neq 2$.

2. Prove: There are no non-zero natural number solutions to the equation $x^2 - y^2 = 1$.

3. Prove the iterative implementation of the Fibonacci sequence with a loop invariant.

4. Prove linear search algorithm is correct with a loop invariant.
   Input: A sequence of n number $A = [a_1, a_2, \ldots, a_n]$ and a value v.
   Output: An index i such that $v = A[i]$ or the special value NIL if v does not appear in A.

5. Solve recurrence: $T(n) = 3T(n-2) + n$, where $T(1) = 3$, $T(0) = 3$.

6. provide upper bound of run time recursion: $T(n) = T(\lceil \frac{n}{2} \rceil) + 1$ to be $\mathcal{O}(\log(n))$