Comparing Algorithms

\[ 5n + 6 \quad =? \quad 10n + 2 \quad =? \quad 2n + 50 \]

- Growth rates & Asymptotic bound \((n \to \infty)\)
- Alg. A vs. alg. B: which algorithm has the smaller asymptotic time bound?

Computational Complexity

- The term complexity of an algorithm refers to its asymptotic bound. That is, the order of that bound.

May be better for small \(n\), but as soon as \(n\) increases, the algorithm becomes inefficient.
“Big-O” and “Big-Θ” notation
L18: Complexity and Algorithm Analysis

Selection Sort

\[ a[ ] = 15, 6, 18, 3, 7, 17, 20, 1, 4, 13, 9 \]
L18: Complexity and Algorithm Analysis

**SelectionSort** \(a[], \text{lower}, \text{upper})\) {
    \text{for}(\text{low} = \text{lower}; \ \text{low}<\text{upper}; \ \text{low}++) {
        \text{smallest} = \text{findSmallest}(a, \text{low}, \text{upper})
        \text{tmp} = a[\text{low}]
        a[\text{low}] = a[\text{smallest}]
        a[\text{smallest}] = \text{tmp}
    }
    \text{return}
}

\text{findSmallest}(a[], \text{lower}, \text{upper})\) {
    \text{smallestIndex} = \text{lower}
    \text{for}(i = \text{lower}+1; i\leq\text{upper}; i++)
        \text{if}(a[i] < a[\text{smallestIndex}])
            \text{smallestIndex} = i
    \text{return \text{smallestIndex}}
}

\(T(n) = ?, \ O?\)

Neglect operations that are independent of the size of the input.
L18: Complexity and Algorithm Analysis

\[ T(n) = ?, \quad O? \]
Recursive version

SelectionSortR(a[], lower, upper) {
    if lower = upper       1
        return
    smallest = findSmallest(a, low, upper)   n
    tmp = a[low]           1
    a[low] = a[smallest]   1
    a[smallest] = tmp      1
    SelectionSortR(a[], lower+1, upper)    T(n-1)
}

T(1) = 1
T(n) = T(n-1) + n + 4  

Recurrence

O?
Recurrences

- **Iteration method**: iterate, express as $\sum$, solve $\sum$
Assignment

• Read Sections 14.6 – 14.8

• Additional readings posted on Blackboard (QuickSort-1 and QuickSort-2)

• Homework #3