Algorithm Benchmarking Analysis

Vasilisa Bashlovkina, David Kraemer, Bo Wang

May 9 2016

Introduction

- An algorithm's "worst-case" run-time is a useful object of study in computer science.
- Denote $f : \mathbb{N} \to \mathbb{N}$ where for a given input n, $f(n)$ is the steps required for the algorithm to terminate.

Definition

We say a function f has **order** g and write $f(n) \in O(g(n))$ if there exist $c \in \mathbb{R}$ and $N \in \mathbb{N}$ such that

$$f(n) \leq c \cdot g(n)$$

for all $n \geq N$.

- This is called **Big-Oh** analysis.

Sorting

Problem
Given an array A, rearrange elements of A such that
$A[k] \leq A[k+1]$ for all k.

- We consider two algorithms: **insertion sort** and **merge sort**.
- Insertion sort has worst-case complexity $O(n^2)$.
- Merge sort has worst-case complexity $O(n \log_2 n)$.
- In theory, merge sort is **much** faster than insertion sort.

Search

Problem
Given a sorted array A and entry v, determine the index k such that $A[k] = v$.

- We consider two algorithms: **linear search** and **binary search**.
- Linear search has worst-case complexity $O(n)$.
- Binary search has worst-case complexity $O(\log_2 n)$.
- In theory, binary search is **much** faster than linear search.

Iteration and recursion

- Algorithms can usually have both iterative and recursive implementations.
- The dominant view is that iteration is faster than recursion.
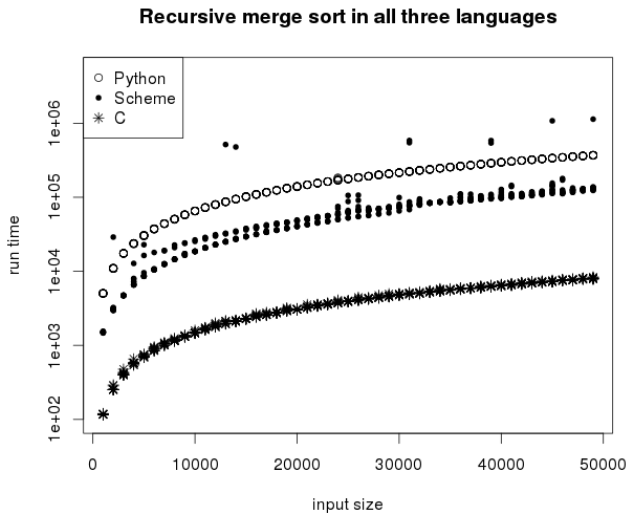- This is due to language-implementation details.

Experiments

- Each algorithm is implemented recursively and iteratively in C, recursively in Scheme, and iteratively in Python.
- It is a common perception that C is faster than both Scheme and Python.
- We generate arrays of length $n = 1000, \ldots 50000$.
- For sorting, the elements are iid Uniform$(0, n)$
- For searching, the differences between adjacent elements are iid Uniform$(0, 10)$.
- We measure program execution time, a rough proxy for steps to termination.
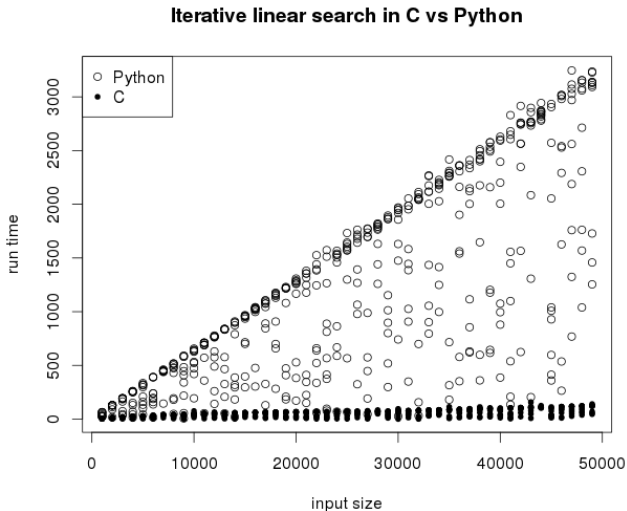
Research Questions

- Does the theorized relationship between input and run-time manifest in practice?
- Is merge sort empirically faster than insertion sort?
- Is binary search empirically faster than linear search?
- Is iteration faster than recursion?
- Is C faster than Scheme or Python?
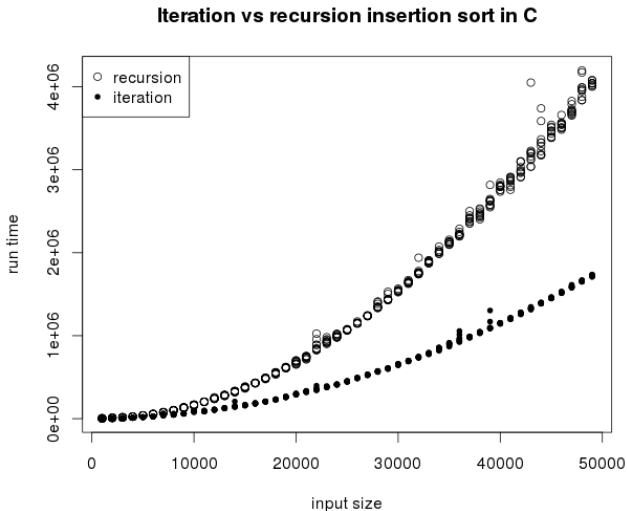- What is the relationship between Scheme and Python?
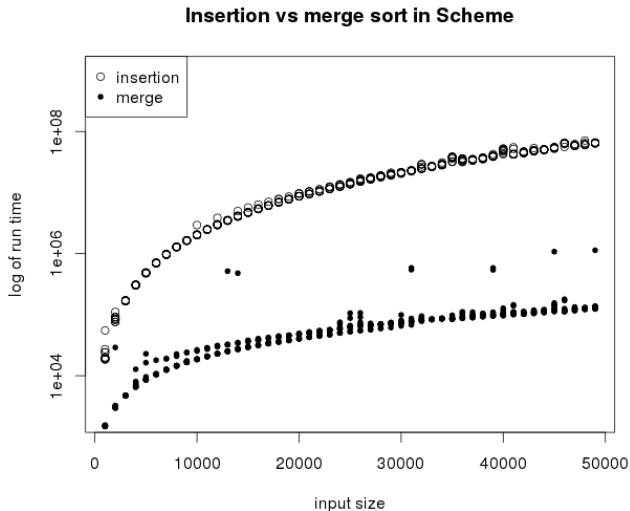
Comparing languages

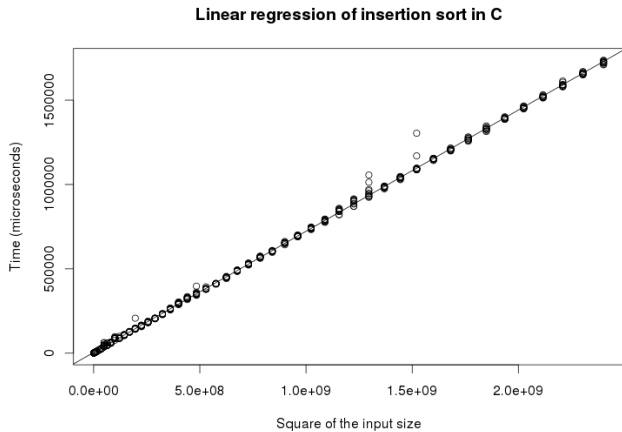**Recursive merge sort in all three languages**

Legend: ○ Python, ● Scheme, ✳ C

x-axis: input size (0, 10000, 20000, 30000, 40000, 50000)

y-axis: run time (1e+02, 1e+03, 1e+04, 1e+05, 1e+06)

Comparing languages

Iterative linear search in C vs Python

Comparing iteration and recursion

Iteration vs recursion insertion sort in C

Comparing sorting algorithms

Insertion vs merge sort in Scheme

Fitted theoretical model

Linear regression of insertion sort in C

Fitted theoretical model
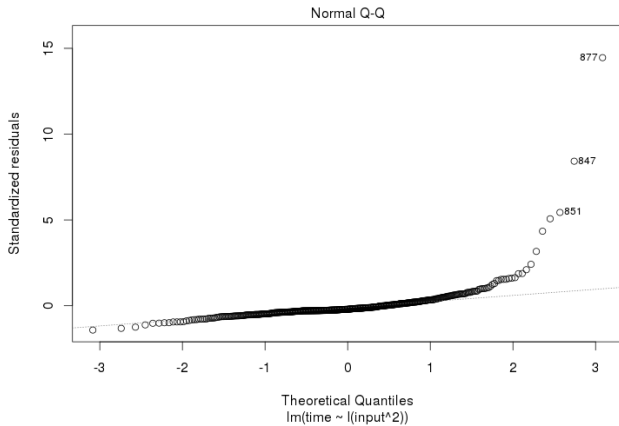
Methodology and Results

- We used paired *t*-tests to identify significant speed differences between
    - recursion vs iteration
    - C, Scheme, and Python,
    - linear vs binary search; insertion vs merge sort
- We found significant ($p < 0.001$) speed differences in almost every comparison.
- We found no significant speed difference between recursive and iterative binary search in C ($p = 0.24$).

Methodology and Results

- We used transformed linear models to determine if empirical data supports theoretical upper bounds of performance
- We found that the theoretical models better fit the data in almost every case.
- We did not find this for binary search in C.

Conclusion

- The overwhelming evidence is that C is the fastest language, iterative is the fastest paradigm, and merge sort and binary search are the fastest algorithms
- We found no significant results for binary search in C because its run time is sub-microsecond.
- Should every program be written in C?
- Should every algorithm be implemented iteratively?