Lecture 7: Analysis of Algorithms

BT 3051 – Data Structures and Algorithms for Biology

Karthik Raman
Department of Biotechnology
Indian Institute of Technology Madras

August 19, 2014

Searching for an element in a list

- Given a list of integers, find if an input integer belongs in the list

Linear Search

```
def linear_search(array,value):
    ''' (list,object) --> int
    Return the index of the first occurrennce of an
                                    object in a list.
    Returns -1 if the number is absent in the list.
    >>> linear_search([1,2,3,4,5],1)
    0
    >>> linear_search([1,2,3,4,5],5)
    4
    >>> linear_search([1,2,3,4,5],6)
    -1
    '''
    n = len(array)

    for i in range(n):
        if array[i] == value:
            return i

    return -1
```

Binary Search

```
def binary_search(array,value):
    ''' (list,object) --> int
    Return the index of the first occurrence of an
    object in a sorted list. Returns -1 if the number
    is absent in the list.
    '''
    N = len(array)
    i = 0
    j = N  - 1
    while i != j+1:
        mid = (i+j)//2
        if array[mid] < value:
            i = mid+1
        else:
            j = mid-1

    if 0 <= i < N and array[i] == value:
        return i
    else:
        return -1
```

Comparison of Times

```
import time
from linear_search import *
from binary_search import *
def time_it(search_function, L, v):
    '''(function,object,value) --> number
    Time (in millseconds) how long it takes to run the
    search_function to find the value v in list L.
    '''
    t_begin = time.perf_counter()
    search_function(L,v)
    t_end = time.perf_counter()
    return int((t_end - t_begin) * 1000)

for prob_sz in [10**7]:
    test_list = list(range(prob_sz))
    for search_func in [linear_search,binary_search]:
        for test_value in [0,prob_sz//2, prob_sz]:
            t=time_it(search_func,test_list,test_value)
            print (search_func.__name__, prob_sz,
                                        test_value, t)
```

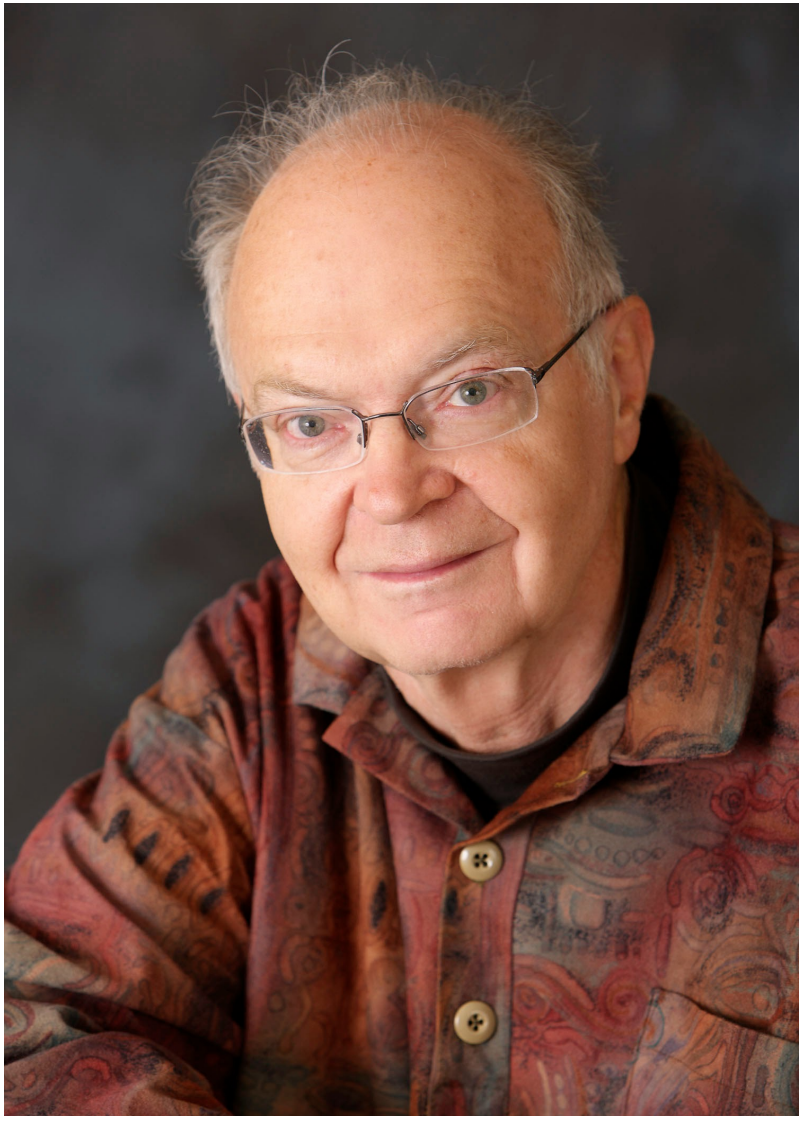
Analysis of Algorithms

- Empirical Analysis
- Mathematical Analysis

Empirical Analysis

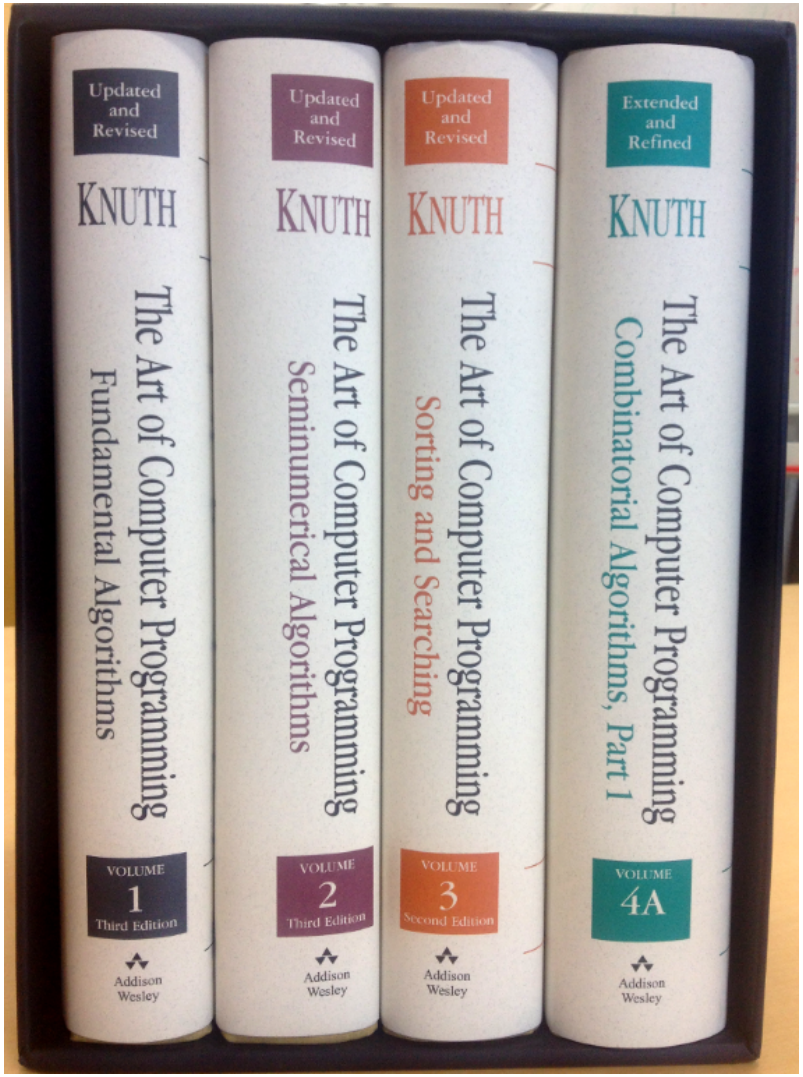
- Involves measuring (timing) program performance …
- Followed by 'curve fitting'!
- Measure things like $T(2n)/T(n)$
- Easy to conduct *experiments*
- Difficult to get precise measurements — many confounding factors e.g. hardware, software, state of the system etc.

Donald Ervin Knuth



The 1974 A.M. Turing Award was presented to Professor Donald E. Knuth of Stanford University for a number of major contributions to analysis of algorithms and the design of programming languages, and in particular for his most significant contributions to the "art of computer programming" through his series of well-known books. The collections of technique, algorithms, and relevant theory in these books have served as a focal point for developing curricula and as an organizing influence on computer science.

Donald Ervin Knuth



*TAOCP emphasized a mathematical approach for comparing algorithms to find out how good a method is. Knuth says that when he decided this approach, he suggested to his publishers renaming the book The Analysis of Algorithms, but they said "*We can't; it will never sell.*" Arguably, the books established analysis of algorithms as a computer science topic in its own right.* ***Knuth has stated that developing analysis of algorithms as an academic subject is his proudest achievement.***

http://amturing.acm.org/award_winners/knuth_1013846.cfm

Image credit: http://www.preining.info/

Mathematical Analysis

- $T_{\text{TOTAL}} = \sum_i \text{FREQUENCY}_i \cdot \text{COST}_i$
- Frequency depends on Algorithm/Program and the data input
- Cost depends on machine
- Involves careful analysis of algorithm performance
- Accurate mathematical models can be constructed in principle
- Important to break down a program into '*basic operations*'[a]

[a] http://en.wikipedia.org/wiki/MMIX