Analytic Methods in Algorithm Research

Overview

This lecture

Computer Science as research field

Central topics and results of algorithm research

Analytic-deductive methods

Next lecture

Experimental methods in algorithmics

Analytic Methods in Algorithm Research

Analytic-deductive methods

Central topics and results of algorithm research

Computer Science as research field

Overview
Algorithmic correctness

**Algorithm:**

- **Exact (and hopefully clear) formalizing:**
  
  - Researcher shall bring order among knowledge!

**Analytic methods of algorithmics** – p. 9/26

**QuickSort**

- **Procedure**

  ```
  procedure QuickSort(S):
  if |S| ≤ 1 then return S;
  else choose a pivot item s ∈ S;
  partition S into S< and S> = s
  return QuickSort(S<) ◦ s ◦ QuickSort(S>);
  endif;
  ```

**Correctness**

**Theorem**

**Procedure QuickSort works correctly.**

**Proof**

Induction on the number of items

**Method**

**1**: Exact (and hopefully clear) formalizing.

**2**: Modeling; just the essential features

**3**: Mathematical induction:

1◦ \( P(k) \)

2◦ \( P(k), P(k+1), \ldots, P(n-1) \) \(⇒\) \( P(n) \)

∴ \( P(n) \) for all \( n \geq k \)

**Algorithmic Efficiency**

- **Amount of required resources (time, memory, ...) with inputs of given size?**
- **As dependency from input size \((n)\)**
- **Simplifications:**
  - concentrate on “large enough” inputs \((n → ∞; n ≥ n_0)\)
  - ignore variable effects of programmer capability and execution environment, by analyzing how number of “basic operations” increase → focus on the scalability of algorithms

**Asymptotic Complexity Class estimates**

**Upper bound**

\( T(n) = O(f(n)) \), if \( T(n) ≤ cf(n) \) for all (sufficiently large) \( n \) and some \( c > 0 \)

**Lower bound**

\( T(n) = Ω(f(n)) \) and exact \( T(n) = Θ(f(n)) \) estimates similarly

**Immediate simplifications**

- \( c \times f(n) = O(f(n)) \), if \( \lim_{n → ∞} g(n)/f(n) = 0 \), then \( f(n) ± g(n) = Θ(f(n)) \) (ignore lower-order terms)

**Algorithmic correctness**

**Correspondence**

**Induction for QuickSort correctness**

**Theorem**: Procedure QuickSort works correctly.

**Proof**

Induction on the number of items

**Example**: Corollary.

The sorting problem arises from an instance of the problem in linear time.

**Algorithmic correctness**

- The sorting problem arises from an instance of the problem in linear time.
Approximating sum by an integral

### AVG complexity of QuickSort (1)

\[
\sum_{k=1}^{n} \frac{k}{n^2} = \frac{1}{2} \left( \frac{1}{n} \right) + \frac{1}{2} \left( \frac{1}{n^2} \right)
\]

### AVG complexity of QuickSort (2)

\[
\sum_{k=1}^{n} \frac{k}{n^2} = \frac{1}{2} \left( \frac{1}{n} \right) + \frac{1}{2} \left( \frac{1}{n^2} \right)
\]

### Worst case of QuickSort

On the average, QuickSort works much better (See next)

**Method 4: Combinatorial Observation**

\[
(\forall n)\big( (n) \big)_0 = (n) \big( (n) \big)_0 \leftarrow (n) \big( (n) \big)_0 \leftarrow \left( (n) \big( (n) \big)_0 \right)
\]

### Worst case of QuickSort (3)

- Good / bad of worst-case analysis:
  - \( (n) \big( (n) \big)_0 = (n) \big( (n) \big)_0 \)

### Method 3: Algebraic Manipulation

\[
(\forall n) 0 = (0) \big( (0) \big)_0 \leftarrow (n) \big( (n) \big)_0 \leftarrow \left( (n) \big( (n) \big)_0 \right)
\]

### Theorem for Direct

\[
(\forall n) \big( (n) \big)_0 = (n) \big( (n) \big)_0 \leftarrow (n) \big( (n) \big)_0 \leftarrow \left( (n) \big( (n) \big)_0 \right)
\]
QuickSort in practice

Logarithms grow slowly ⇝ an algorithm with complexity $O(n \log n)$ scales almost as well as a linear one.

QuickSort is one of the most popular sorting methods of practice, for example for its friendliness wrt virtual memory.

Other points of interest: implementability, usability, applicability, validity of analytic results, actual performance, etc.

Methods are analytic-deductive, mathematical, or empirical.
- efficiency
- the correctness
- the complexity
- algorithms

Results of theoretical algorithm research -- Experimental Research of Algorithms
- Summary

Analytic methods of algorithmics -- p. 25/26

Experimental methods of algorithmics -- p. 26/26