Searching

Markus Roggenbach

January 2005

Examples: where do we search?

- Google
- Telephone book
- Train connection

Making things precise: Searching as a Computational Problem

Searching Problem:

Input: array $A[1..n]$ of n numbers;
number d

Output: if d appears in $A[1..n]$:
smallest index $i \in \{1, \ldots, n\}$ with $A[i] = d$
otherwise: 0

Definition 1 (computational problem)

A computational problem specifies in general terms the desired input/output relationship.

We formulate such problems using the scheme:

<**ProblemName**>:

Input: <*list of inputs*>

Output: <*list of desired outputs*>

Algorithms which shall solve the Sorting Problem

SeqSearch$(A, p, q; d)$ (iterative version)

```
1 l ← 0
2 WHILE p ≤ q   DO
3                 IF A[p] = d THEN    l, q ← p
4                 p ← p + 1
5                 OD
6 output l
```

$\textsc{SeqSearch}(A, p, q; d)$ (recursive version)

```
1 IF p > q THEN   output 0
2          ELSE   DO
3                 IF A[p] = d
4                 THEN output p
5                 ELSE SeqSearch(A, p + 1, q; d)
6                 OD
```

Correctness

Question: Are the two variants of SeqSearch doing what we expect them to do?

Definition 2 (Correctness)

A algorithms is said to be correct if, for every input instance, it halts with the correct output. We say that a correct algorithms solves the given computational problem.

Remark 1 (Computational Problems and correct Algorithms)
For one computational problem
there are many different algorithms solving it.

Running time

Question: How many steps do the variants of SeqSearch need to produce an output?

Definition 3 (Running time)

The running time of an algorithms on a particular input is the number of primitive operations or "steps" executed.

Usually we concentrate on the worst case running time, that is, the longest running time for any input of size n.

General approach:

1. Formulate a computational problem P.
2. Write an algorithm A
3. Prove: A solves P, i.e.
   - A halts for every input
   - A produces the correct output
4. Analyse the running time of A.

Links to the Course Text:

- Algorithm, computational problem, correctness: section 1.1 of [CFRS03]
- Running time: section 2.2. of [CFRS03]
- Pseudo code conventions: pp.19-20 in [CFRS03]