Algorithms

Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Algorithms

Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Examples:

- computing with floating point numbers
- compressing data
- executing machine code

Algorithms

Algorithm: a well-ordered collection of unambiguous and effectively computable operations, that, when executed, produces a result in a finite amount of time.

Examples:

- computing with floating point numbers
- compressing data
- executing machine code

Program: representation of an algorithm
Pseudocode: representation of an algorithm
Process: execution of an algorithm

Algorithms

Art of problem solving
Polya's principles applied to algorithms:

1. Understand the problem
2. Get an idea for a possible algorithmic procedure (to solve it)
3. Formulate the algorithm and represent it as a program
4. Evaluate the program for correctness and its potential as a tool for solving other problems

Algorithms

Art of problem solving
Polya's principles applied to algorithms:

1. Understand the problem
2. Get an idea for a possible algorithmic procedure (to solve it)
3. Formulate the algorithm and represent it as a program
4. Evaluate the program for correctness and its potential as a tool for solving other problems

Not so easy as $1 \to 2 \to 3 \to 4$.

Algorithms

Examples:

- Magic trick — ideas,
  discover they don't work with some initial cards...
- 3 politicians (no names) A, B, C — know each other
  - 1 always tells the truth
  - 1 always lies
  - 1 does some of each
  - Ask 3 true/false questions
    - choose whichever politician you like for whichever question
    - determine which politician is which

Algorithm design techniques

Techniques:

- Brute force
- Stepwise refinement (top-down)
  - break into smaller and smaller problems
  - **if** modular (relatively independent) parts, can program in teams — software engineering

Algorithm design techniques

Cute problems in textbook.

Example: Step from pier into a boat
Hat falls into water.
River flows 2.5 miles/hour
Go upstream at 4.75 miles/hour
After 10 minutes discover hat missing.
Turn around to travel downstream.
How long before you get to the hat?

Cute problems in textbook.

Example: Step from pier into a boat
Hat falls into water.
River flows 2.5 miles/hour
Go upstream at 4.75 miles/hour
After 10 minutes discover hat missing.
Turn around to travel downstream.
How long before you get to the hat?
Answer: 10 minutes
— It pays to think.

Pseudocode

- easier to read than a program
- syntax less important
- constructs from many languages work the same

Pseudocode

Pseudocode

- easier to read than a program
- syntax less important
- constructs from many languages work the same
  - `if...then...else` — condition is Boolean
  - `while`
  - `repeat`
  - `for`
  - recursion

Pseudocode

Types — use consistently and clearly

Incorrect example: $\texttt{Card} \leftarrow \texttt{Card} + \texttt{n}$

Pseudocode

Types — use consistently and clearly

Incorrect example: `Card` $\leftarrow$ `Card` + `n`
Incorrect example: Suppose Card has the form (s_1, v_1) and $1 \leq n \leq 6$.

Must explain the general idea and what variables are used for if not obvious — not what it does, but why,
in `if...then...else` clause for example.

Sequential search

Sequential search problem:
Input: List of elements, TargetValue
Output: success if TargetValue is in List
failure if it is not in List

A brute force algorithm.

Sequential search


```
procedure Search(List, TargetValue):
{ Input: List is a list; TargetValue is a possible entry }
{ Output: success if TargetValue in List; failure otherwise }

    if (List empty)
        then Output failure

        else
            TestEntry ← 1st entry in List
            while (TargetValue ≠ TestEntry
                        and there are entries not considered)
                do (TestEntry ← next entry in List)
            if (TargetValue = TestEntry)
                then Output success
                else Output failure
```

Sequential search

Analysis:

- time
- fundamental operation
  - takes time
  - number of occurrences proportional to everything else that happens

Sequential search

Analysis:

$|\text{ List }| = n$

How many comparisons are necessary in the worst case?

A. 1

B. $n - 1$

C. n

D. $n + 1$

E. $2n$

Vote at m.socrative.com. Room number 415439.

Sequential search

Analysis:

$|\text{ List }| = n$

How many comparisons are necessary in the worst case?

D. $n + 1$

This is $\Theta(n)$.

Sequential search

Analysis:
What does $\Theta(n)$ meant?
Need to define $O(n)$ too.

$g \in O(f)$ means $\exists c, d$ s.t. $g(n) \leq c \cdot f(n) + d$
$g \in \Theta(f)$ means $g \in O(f)$ and $f \in O(g)$.

Sequential search

Analysis:
$g \in O(f)$ means $\exists c, d$ s.t. $g(n) \leq c \cdot f(n) + d$
$g \in \Theta(f)$ means $g \in O(f)$ and $f \in O(g)$.

Examples:

- $2n + 3 \in \Theta(n)$
- $3 \log n \in \Theta(\log n)$
- $2n + 7 \log n \in \Theta(n)$
- $4 \log n + m \in \Theta(\log n)$ if $m \leq \log n$
- Can write $\Theta(\log n + m)$ if unsure which term is larger.

Sequential search

Analysis:
What is $n \log n - 1.4n + 15$?

A. $O(n^2)$

B. $O(n \log n)$

C. $\Theta(n \log n)$

D. all of the above

E. none of the above

Vote at m.socrative.com. Room number 415439.

Sequential search — correctness

procedure Search(List, TargetValue):
  if (List empty)
    then Output failure
    else
      TestEntry $\leftarrow$ 1st entry in List
    { precondition: TestEntry is 1st entry in List }
      while (TargetValue $\neq$ TestEntry
          and there are entries not considered)
        do (TestEntry $\leftarrow$ next entry in List)
    { loop invariant: TargetValue $\neq$ any entry before TestEntry }
    { postcondition: either TargetValue $=$ TestEntry
      or all entries considered and TargetValue not in List }
      if (TargetValue $=$ TestEntry)
        then Output success
        else Output failure

Sequential search — correctness

Assertions

- statements which can be proven to hold (induction)
- at different points in program
- examples: precondition, postcondition, loop invariant

Proof by induction on number of times through the loop:

Proof verification: automated?

Sequential search — correctness