CS 281
Prof. Alex Pothen
Webbook:
http://www.cs.odu.edu/~webbook
http://www.cs.odu.edu/~pothen/Courses/CS281
Wed Sep 16, 1998
Lecture 5
Sec 2.1 Algorithms, App. 2
Next Lecture:
Sec 2.2, 1.8
Algorithms

- Given a sequence of $n$ integers, find the smallest element.

- Example:  
  \[ a_1 \ a_2 \ a_3 \ a_4 \ a_5 \ a_6 \ a_7 \]  
  \[ 10 \ 20 \ 12 \ 6 \ 30 \ 3 \ 15 \]

- Method:  
  set temporary minimum (min) to $a_1$;  
  compare next integer with min. If it is smaller than min, set min to the integer.  
  repeat step 2 until all integers have been compared with min.
Pseudocode

• An informal way to specify the steps in an algorithm

• intermediate between a description in English (imprecise) and a translation into a programming language (precise).

• Constructs:
  comments: { this is a comment}
  assignment statement
  if block statement for selection
  for loop and while loop for repetition
Selection Statement

• if (condition) then statement fi ;

• e.g. if (\(n > 0\)) then \(x := 1/n\); fi ;

• If the condition is true, then execute the block of statements; if not, continue after the fi (end if) statement.
Selection Statement II

- **if** (condition 1) **then** statement 1;
  **else if** (condition 2) **then** statement 2;
  **else** statement 3; **fi**;

- Meaning: if condition 1 is true, then execute statement 1, and continue after **fi**;
  if condition 1 is false, check if condition 2 is true;
  if condition 2 is true, then execute statement 2, and continue after **fi**;
  if condition 2 is also false, execute statement 3, and continue after **fi**.
Repetition: FOR loop

- **for** `index_variable := initial_value to final_value` **do**
  statement;
  **od**

- for each value of `index_variable` from `initial_value` to `final_value`, execute the statement.

- If `initial_value` is smaller than `final_value`, then the value of the `index_variable` increases by one after each iteration.

- More general:
  ```plaintext
  for index_variable := initial_value to final_value by increment do
  statement; od
  ```
Repetition: WHILE loop

- **while** (condition) **do**
  statement; **od**

- Check if the condition is true;
  if false, continue after the **od** (end while) statement.
  if true, execute the statement.
  Repeat: check the condition again, and execute the statement as long as the condition is true.
Pseudocode for min-finding algorithm

Min := proc(A: list)
    min := a_1;
    for i := 2 to n do
        if (a_i < min) then min := a_i;
    end for
end proc
**Sentinel Search**

- Given a sequence of integers $a_1, a_2, \ldots, a_n$, and an integer, item, search for item in the sequence. If item is found, then return the index of its first occurrence; else print ‘item not found’.

- Method:
  
  Set $a_{n+1} := \text{item (sentinel)}$;
  
  set $i := 1$;
  
  compare $a_i$ and item:
  
  if equal, return $i$ and stop
  
  if unequal, increase $i$ by one, and repeat comparison step.
  
  if $i < n + 1$, then item found; else, not found.
Sentinel Search algorithm

SentinelSearch := proc(A::list(integer),
    item::integer)
    a_{n+1} := item;
    i := 1;
    while (a_i \neq item) do
        i := i + 1;
    od
    if i < n + 1 then
        item is equal to a_i;
    else
        item is not found;
    fi
Linear Search algorithm

LinearSearch := proc\(A::\text{list}(\text{integer}),\item::\text{integer}\)
\begin{align*}
i &:= 1; \\
\textbf{while} \ (i \leq n \ \textbf{and} \ a_i \neq \item) \ \textbf{do} \\
\quad i &:= i + 1; \\
\textbf{od} \\
\textbf{if} \ i < n + 1 \ \textbf{then} \\
\quad \item \ \text{is equal to} \ a_i; \\
\textbf{else} \\
\quad \item \ \text{is not found}; \\
\textbf{fi}
\end{align*}
Properties of Algorithms

• An algorithm is a definite procedure for solving a computational problem using a finite number of steps.

• An algorithm has the following properties:
  – It accepts input values from a specified set.
  – It produces a set of output values.
  – Each step is defined precisely.
  – For any input in the specified set, it produces results in a finite number of steps.
  – It should be possible to do each step in finite time.
  – It should be applicable to all instances of the computational problem.
• Verify that each algorithm we looked at satisfies these properties.