Algorithms

COMP 102, lecture 7

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** $x_1x_2x_3$ ... x_n

for i:=1 **to** n-1 **do**
  j:=i-1+FindMin(x_ix_{i+1} ... x_n)
  temp:=x_i; x_i:=x_j; x_j:=temp

output $x_1x_2x_3$ ... x_n

Find Minimum

given $x_1x_2...x_n$ find i such that $x_i \le x_j$, $1 \le j \le n$

- **Procedure FindMin**($x_1x_2...x_n$)
  mini:=1; min:=x_1
  for i:=2 **to** n **do**
    if x_i<min **then** min:=x_i; mini:=i
  output mini

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)

  for i:=1 **to** n-1 **do**

    j:=i-1+FindMin($x_i x_{i+1}$ ... x_n)

    temp:=x_i; x_i:=x_j; x_j:=temp

  output $x_1x_2x_3$ ... x_n

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)
  for i:=1 **to** n-1 **do**

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)
  for i:=1 **to** 3 **do**

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)
  for i:=1 **to** 3 **do**
  i=1: j:=i-1+FindMin($x_i x_{i+1}$ ... x_n) ⊢

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)
  for i:=1 **to** 3 **do**
  i=1: j:=i-1+FindMin($x_i x_{i+1}$ ... x_n) ⊢
  j:=1-1+FindMin(x_1(=3)x_2(=2)x_3(=6)x_4(=1)) ⊢

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)

  for i:=1 **to** 3 **do**

  i=1: j:=i-1+FindMin($x_i x_{i+1}$ ... x_n) $\vdash$

  j:=1-1+FindMin(x_1(=3)x_2(=2)x_3(=6)x_4(=1)) $\vdash$

  j:=1-1+4 $\vdash$ j:=4

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)

  for i:=1 **to** 3 **do**

  i=1: j:=4

  temp:=x_i ⊢ temp:=x_1 ⊢ temp:=3

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)

  for i:=1 **to** 3 **do**

  i=1: j:=4

  temp:=3

  $x_i := x_j \vdash x_1 := x_4 \vdash x_1 := 1$

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)

  for i:=1 **to** 3 **do**

  i=1: j:=4

  temp:=3

  x_1:=1

  x_j:=temp ⊢ x_4:=3

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)

  for i:=1 **to** 3 **do**

  i=1: j:=4 $(x_1(=1)x_2(=2)x_3(=6)x_4(=3))$

  i=2: j:=i-1+FindMin($x_i x_{i+1}$ ... x_n) $\vdash$

  j:=2-1+FindMin($x_2(=2)x_3(=6)x_4(=3)$) $\vdash$

  j:=1+1 $\vdash$ j:=2

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)

  for i:=1 **to** 3 **do**

  i=1: j:=4 (x_1(=1)x_2(=2)x_3(=6)x_4(=3))

  i=2: j:=2

  temp:=2

  x_2:=2

  x_2:=2

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)

  for i:=1 **to** 3 **do**

  i=1: j:=4 $(x_1(=1)x_2(=2)x_3(=6)x_4(=3))$

  i=2: j:=2 $(x_1(=1)x_2(=2)x_3(=6)x_4(=3))$

  i=3: j:=i-1+FindMin(x_ix_{i+1} ... x_n) $\vdash$

  j:=3-1+FindMin($x_3(=6)x_4(=3)$) $\vdash$

  j:=3-1+2 $\vdash$ j:=4

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)

  for i:=1 **to** 3 **do**

  i=1: j:=4 (x_1(=1)x_2(=2)x_3(=6)x_4(=3))

  i=2: j:=2 (x_1(=1)x_2(=2)x_3(=6)x_4(=3))

  i=3: j:=4

  temp:=6

  x_3:=1

  x_4:=6

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)
  for i:=1 **to** 3 **do**

i=1: j:=4	$(x_1(=1)x_2(=2)x_3(=6)x_4(=3))$
i=2: j:=2	$(x_1(=1)x_2(=2)x_3(=6)x_4(=3))$
i=3: j:=4	$(x_1(=1)x_2(=2)x_3(=3)x_4(=6))$

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **input** x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1)
  for i:=1 **to** 3 **do** (x_1(=1)x_2(=2)x_3(=3)x_4(=6))

output x_1 (=1) x_2 (=2) x_3 (=3) x_4 (=6)

Recursion

a different way of thinking

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**($x_1x_2...x_n$)
  if n=1 **then output** x_1
  else j:=FindMin(x_1x_2 ... x_n)
  temp:=x_1; x_1:=x_j; x_j:=temp
  output x_1, **Sort**(x_2x_3 ... x_n)

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  if n=1 **then output** x_1
  else j:=FindMin($x_1x_2x_3x_4$)
  temp:=x_1; x_1:=x_j; x_j:=temp
  output x_1, **Sort**($x_2x_3x_4$)

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  if n=1 **then output** x_1 **else** ...
  ⊢ **if** 4=1 **then output** x_1 **else** ...
  ⊢ j:=FindMin($x_1x_2x_3x_4$)
  temp:=x_1; x_1:=x_j; x_j:=temp
  output x_1, **Sort**($x_2x_3x_4$)

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  j:=FindMin($x_1x_2x_3x_4$) $\vdash$ j:=4
  temp:=x_1; x_1:=x_4; x_4:=temp
  (x_1 (=1) x_2 (=2) x_3 (=6) x_4 (=3))

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  (x_1 (=1) x_2 (=2) x_3 (=6) x_4 (=3))

  output x_1 (=1), **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \le x_{i+1}$, $1 \le i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  if n=1 **then output** x_1
  else j:=FindMin($x_1x_2x_3$)
  temp:=x_1; x_1:=x_j; x_j:=temp
  output x_1, **Sort**(x_2x_3)

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  j:=FindMin($x_1x_2x_3$) $\vdash$ j:=1
  temp:=x_1; x_1:=x_1; x_1:=temp
  (x_1 (=2) x_2 (=6) x_3 (=3))
  output x_1, **Sort**(x_2x_3)

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=6) x_2 (=3))
  if n=1 **then output** x_1 **else** ...

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=6) x_2 (=3))
  else j:=FindMin(x_1x_2)
  temp:=x_1; x_1:=x_j; x_j:=temp
  output x_1, **Sort**(x_2)

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=6) x_2 (=3))
  else j:=FindMin(x_1x_2) $\vdash$ j:=2
  temp:=6; x_1:=3; x_2:=6; **output** x_1, **Sort**(x_2)

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=6) x_2 (=3))
  output 3, **Sort**(x_2 (=6))

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=6) x_2 (=3))
  output 3, **Sort**(x_2 (=6))

- $\vdash$ **Procedure Sort**(x_1 (=6))

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=6) x_2 (=3))
  output 3, **Sort**(x_2 (=6))

- $\vdash$ **Procedure Sort**(x_1 (=6))
  if n=1 **then output** x_1 **else** ... $\vdash$ **output** x_1 (=6)

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=6) x_2 (=3))
  output 3, **Sort**(x_2 (=6))

- $\vdash$ **Procedure Sort**(x_1 (=6))
  output 6

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=6) x_2 (=3))
  output 3, **Sort**(x_2 (=6))
  $\vdash$ **output** 3, 6

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=6) x_2 (=3))
  output 3, 6

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))
- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, **Sort**(x_2 (=6) x_3 (=3))
  $\vdash$ **output** 2, 3, 6

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

- $\vdash$ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
  output 2, 3, 6

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))
  $\vdash$ **output** 1, 2, 3, 6

Sort

given $x_1x_2...x_n$ rearrange such that $x_i \leq x_{i+1}$, $1 \leq i < n$

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, 2, 3, 6

Simulating iteration via Recursion

- **while** <boolean-expression> **do** <instructions>

  * is the same as *

- **Procedure** Whiledo()

  if <boolean-expression>

  then <instructions>; Whiledo()

- Simulating Recursion via Iteration is possible but is rather complicated.

Global and local variables

- A variable is designated as **local** if its name appears explicitly as a parameter in the header of the Procedure in which it is used. Otherwise it is considered **global** and refers to a variable outside of the Procedure.

- Any assignment to a local variable only affects the value of this variable within the Procedure. If other variables outside the Procedure exist with the same name, they remain unaffected by the assignment.

- Any reference to a local variable will return its value as stored locally. Other variables with the same name may exist outside the Procedure, but there is no mechanism to access them directly.

Example

- **Procedure Sort**(x_1 (=3) x_2 (=2) x_3 (=6) x_4 (=1))
  output 1, **Sort**(x_2 (=2) x_3 (=6) x_4 (=3))

  - ⊢ **Procedure Sort**(x_1 (=2) x_2 (=6) x_3 (=3))
    output 2, **Sort**(x_2 (=6) x_3 (=3))

  - ⊢ **Procedure Sort**(x_1 (=6) x_2 (=3))
    Swap(x_1,x_2) (x_1 (=3) x_2 (=6))
    output x_1 (=3), **Sort**(x_2 (=6))

local x_1 is affected