ELTE | FACULTY OF INFORMATICS

PROGRAMMING

Lecture 4

Zsuzsa Pluhár

pluharzs@inf.elte.hu

Patterns of Algorithms

What is PoA? It is the general solution of a typical programming task.

- sequence → single value
- sequence → sequence
- sequence → sequences
- sequences → sequence

7. Copy

Tasks (examples):

1. Let's define the absolute values of **all elements** in an array!
2. Let's convert **all letters** of a text to lowercase!
3. Let's calculate the sum of two vectors!
4. Let's calculate sin(x) values for a given x series!
5. We know the ordinal number of N months, let's give their names!

7. Copy

What is common?

We have a sequence of „somethings”, and we have to assign another sequence to these. The type of the assigned values can be different from the original type of the elements, but the count remains the same, as well as the order of the elements in the sequence (mostly).

7. Copy

Specification

Input: $X[1..]\in\mathbb{S}_1^*$, $f: \mathbb{S}_1\to\mathbb{S}_2$

Output: $Y[1..]\in\mathbb{S}_2^*$

Precondition: –

Postcondition: $\forall i(1\leq i\leq length(X)):Y[i]=f(X[i])$

Short: $Y[1..length(X)]=f(X[1..length(X)])$

7. Copy

Specification

Input: $X[1..]\in\mathbb{S}_1^*$, $f: \mathbb{S}_1\to\mathbb{S}_2$

Output: $Y[1..]\in\mathbb{S}_2^*$

Precondition: –

Postcondition: $\forall i(1\leq i\leq \mathrm{length}(X)):Y[i]=f(X[i])$

Short: $Y[1..\mathrm{length}(X)]=f(X[1..\mathrm{length}(X)])$

7. Copy

Algorithm

```
i=1..length(X)
  Y[i]:=f(X[i])
```

Note: It's not a must to use the same i index for both arrays. Eg.

Postcondition:

∀i(1≤i≤length(X)):Y[p(i)]=f(X[i])

```
i=1..length(X)
  Y[p(i)]:=f(X[i])
```

`p(i)` could be `2*i` or `length(X)-i+1` (defined with the needed array size and index-interval)

7. Copy

f is often a function with a conditional statement

Specification (a frequent special case)₁

Input: $X[1..]\in\mathbb{S}^*$, $\boxed{g:\ \mathbb{S}\to\mathbb{S},\ A:\mathbb{S}\to\mathbb{L}}$

Output: $Y[1..]\in\mathbb{S}^*$

Precondition: –

Postcondition: $\forall i(1\le i\le length(X)):Y[i]=f(X[i])$

Definition: $f(x)=\begin{cases} g(x), & if\ A(x) \\ x, & otherwise \end{cases}$

7. Copy

Specification (a frequent special case)$_1$

Input: $X[1..] \in \mathbb{S}^*$, $g: \mathbb{S} \to \mathbb{S}$, $A: \mathbb{S} \to \mathbb{L}$

Output: $Y[1..] \in \mathbb{S}^*$

Precondition: –

Postcondition: $\forall i(1 \le i \le \text{length}(X)): (A(X[i]) \to Y[i]=g(X[i])$ $\text{and not } A(X[i]) \to Y[i]=X[i])$

7. Copy

Algorithm (a frequent special case)$_1$

Specification (a frequent special case)$_1$

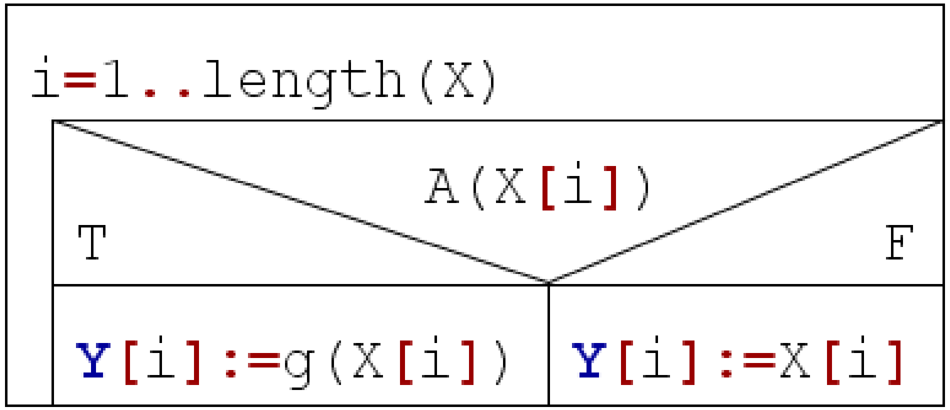
Input: X[1..]∈$\mathbb{S}^*$, g: $\mathbb{S}\to\mathbb{S}$, A:$\mathbb{S}\to\mathbb{L}$

Output: Y[1..]∈$\mathbb{S}^*$

Precondition: –

Postcondition:

∀i(1≤i≤length(X)):**(A(X[i]) → Y[i]=g(X[i]) and not A(X[i])→ Y[i]=X[i])**



7. Copy

Specification (another special case)$_2$

Input: $X[1..] \in \mathbb{S}^*$

Output: $Y[1..] \in \mathbb{S}^*$

Precondition: –

Postcondition: $\forall i(1 \le i \le \text{length}(X)): \mathbf{Y[i]=X[i]}$

Note: there is no `f` function, or more precisely it is identical (`f(x):=x`)

7. Copy

Algorithm (another special case)$_2$

```
i=1..length(X)
  Y[i]:=X[i]
```

Note: It can be replaced by the `Y:=X` value assignment, if the two arrays have the same size. Except when the indexes are different.

```
i=1..length(X)
  Y[p(i)]:=X[i]
```

7. Copy - example

```
i=1..length(X)
  Y[i]:=f(X[i])
```

Task: Let's calculate the sum of two vectors!

Specification

Input: $P[1..n]\in\mathbb{R}^n$, $Q[1..n]\in\mathbb{R}^n$

$f:\mathbb{R}\times\mathbb{R}\to\mathbb{R}$, $f((p_i,q_i))=\mathbf{p_i+q_i}$

Output: $R[1..n]\in\mathbb{R}^n$

Precondition: –

Postcondition:

$\forall i(1\le i\le n)):\mathbf{R[i]=P[i]+Q[i]}$

length(X)	→	n
X[ ]	→	P[ ], Q[ ]
Y[ ]	→	R[ ]
f(i)	→	P[i]+Q[i]

Algorithm:

```
i=1..n
  R[i]:=P[i]+Q[i]
```

8. Multiple item selection

Tasks (examples):

1. Let's **define all** excellent students in a class!
2. Let's calculate the divisor**s** of an integer!
3. Let's **list all** the vowels from an English word!
4. Let's **collect all** the people who are taller than 180cm, from a set of people!
5. Let's **list** those days of the year when it didn't freeze at noon!

8. Multiple item selection

What is common?

We have to list **all** elements from a sequence of "somethings" which have a common attribute A.

8. Multiple item selection

Specification (selecting indexes)

Input: `X[1..]`$\in\mathbb{S}^*$, `A:` $\mathbb{S}\to\mathbb{L}$

Output: `cnt`$\in\mathbb{N}$, `Y[1..]`$\in\mathbb{N}^*$

using only the first `cnt` element

`Y[1..cnt]`$\in\mathbb{N}^{\texttt{cnt}}$

Precondition: –

Postcondition: $$cnt = \sum_{\substack{i=1 \\ A(X[i])}}^{length(X)} 1$$

see the COUNTING PoA!

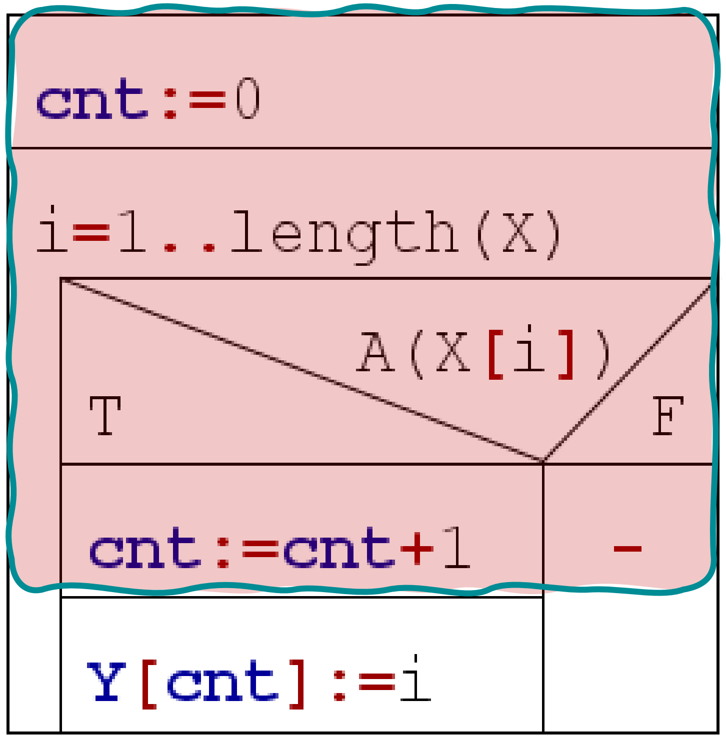
`and ∀i(1≤i≤cnt):A(X[Y[i]])and Y`**⊆(1,2,…,length(X))**

Short : $$(cnt, Y) = \underset{\substack{i=1 \\ A(X[i])}}{\overset{length(X)}{\boldsymbol{MULTISELECT}}}\ (i)$$

8. Multiple item selection

Algorithm (selecting indexes)

see the COUNTING PoA!



Comment: Collecting indexes is more general than collecting elements. If we needed elements then we would write: `Y[Cnt]:=X[i]` (and accordingly modified specification, as well – see later)

8. Multiple item selection

Specification (selecting values)

Input: X[1..]∈$\mathbb{S}^*$, A:$\mathbb{S}\to\mathbb{L}$

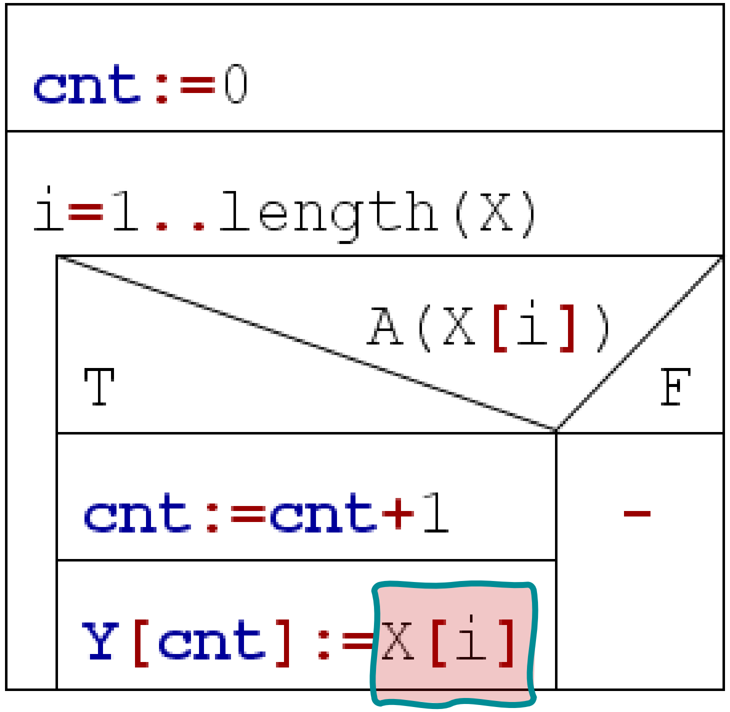
Output: cnt∈$\mathbb{N}$, Y[1..]∈$\mathbb{S}^*$

Precondition: –

Postcondition: $$cnt = \sum_{\substack{i=1 \\ A(X[i])}}^{length(X)} 1$$

and ∀i(1≤i≤cnt):A(Y[i])and Y ⊆ **X**

Algorithm:



Short : $$(cnt, Y) = \underset{\substack{i\,=\,1 \\ A(X[i])}}{\overset{length(X)}{\boldsymbol{MULTISELECT}}}\,(X[i])$$

8. Multiple item selection - example

Task: let's list those days of the year when it didn't freeze at noon!

Specification (selecting indexes)

Input: $T[1..n] \in \mathbb{R}^n$, $Pos: \mathbb{R} \to \mathbb{L}$, $Pos(x) = (x > 0)$

Output: $cnt \in \mathbb{N}$, $NF[1..n] \in \mathbb{N}^n$

using only the first cnt element

$NF[1..cnt] \in \mathbb{N}^{cnt}$

Precondition: –

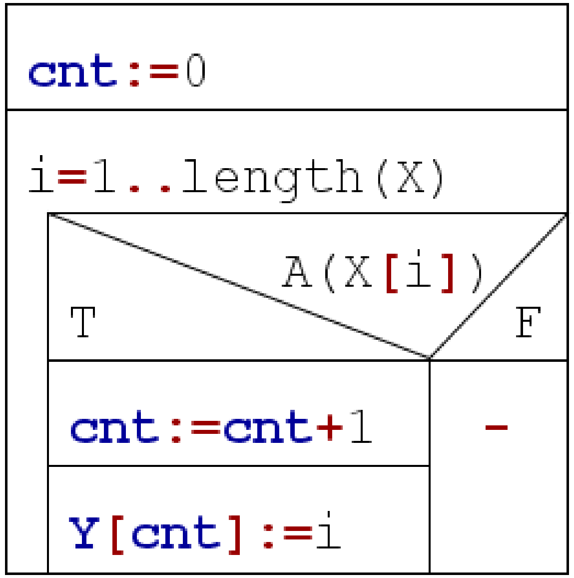
Postcondition: $cnt = \sum_{\substack{i=1 \\ T[i]>0}}^{n} 1$

and $\forall i(1 \leq i \leq cnt): T[NF[i]] > 0$ and $NF \subseteq (1, 2, \ldots, n)$

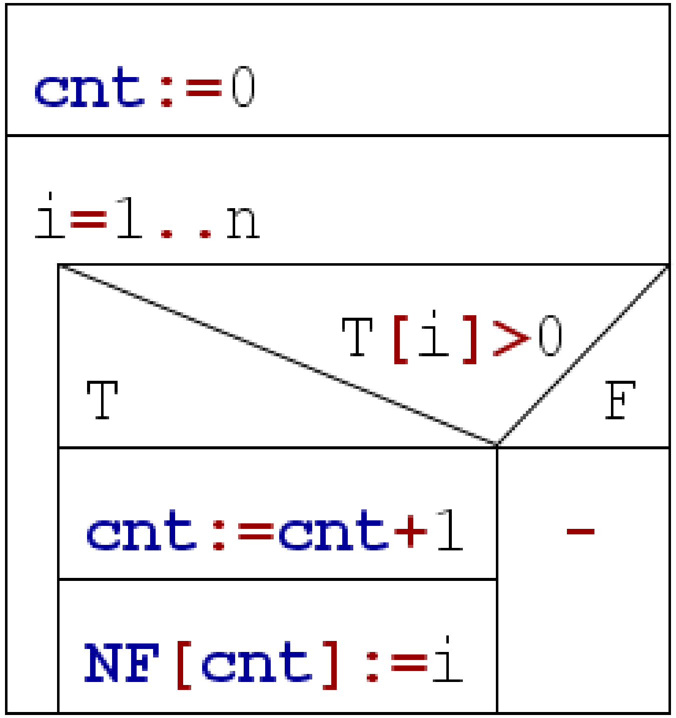
8. Multiple item selection - example

Task: let's list those days of the year when it didn't freeze at noon!

Algorithm (selecting indexes):



length(X)	→	n
X[ ]	→	T[ ]
Y[ ]	→	NF[ ]
cnt	→	cnt
A(X[i])	→	T[i]>0



8. Local MI selection

Specification (selecting indexes)

Input: X[1..]∈$\mathbb{S}^*$, A: $\mathbb{S}\rightarrow\mathbb{L}$

Output: cnt∈$\mathbb{N}$, Y[1..]$\boxed{\in\mathbb{S}^*}$

Precondition: –

Postcondition:

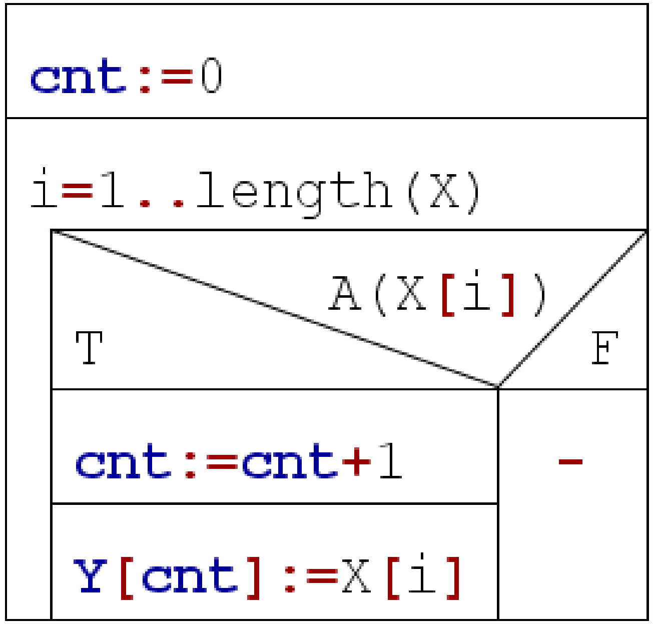
$$cnt = \sum_{\substack{i=1 \\ A(X[i])}}^{length(X)} 1$$

and ∀i(1≤i≤cnt):A(X'[i])and X'[1..cnt]⊆ X)

8. Local MI selection

Main idea: we put the selected item to a place which we don’t use later

Algorithm



9. Partitioning

Tasks (examples):

1. Let's **list all even** and all **odd** numbers from a series!
2. Let's **list** those days of the year when it was **freezing** and when it **wasn’t** at noon!
3. Let's **list** all the **vowels** and **consonants** of an English word!
4. Let's **collect** all the people who are **shorter** than 140cm, who are **between** 140-180 cm, and those who are **taller** than 180cm, from a set of people!
5. From a group of people, **list** the people who were born in **summer**, in **autumn**, in **winter** and in **spring**.

9. Partitioning

What is common?

We have to list **all** elements from a sequence of "somethings" which have a common attribute A, and then also list those ones not having attribute A. So, we “assign” all the elements of the input to one of the output sequences.

9. Partitioning

$Y[1..cnt] \in \mathbb{N}^{cnt}$

$Z[1..length(X)-cnt] \in \mathbb{N}^{length(X)-cnt}$

Specification (selecting indexes)

Input: $X[1..] \in \mathbb{S}^*$, $A: \mathbb{S} \to \mathbb{L}$

Output: $cnt \in \mathbb{N}$, $Y[1..] \in \mathbb{N}^*$, $Z[1..] \in \mathbb{N}^*$

Precondition: –

Postcondition:
$$cnt = \sum_{\substack{i=1 \\ A(X[i])}}^{length(X)} 1$$

see the COUNTING PoA!

and $\forall i(1 \leq i \leq cnt): A(X[Y[i]])$

and $\forall i(1 \leq i \leq length(X)-cnt):$ not $A(X[Z[i]])$

and $Y \subseteq \mathbf{(1,2,\ldots,length(X))}$ and $Z \subseteq \mathbf{(1,2,\ldots,length(X))}$

9. Partitioning

Specification

Input: `X[1..]`$\in\mathbb{S}^*$, `A:` $\mathbb{S}\to\mathbb{L}$

Output: `cnt`$\in\mathbb{N}$, `Y[1..]`$\in\mathbb{N}^*$, `Z[1..]`$\in\mathbb{N}^*$

Precondition: –

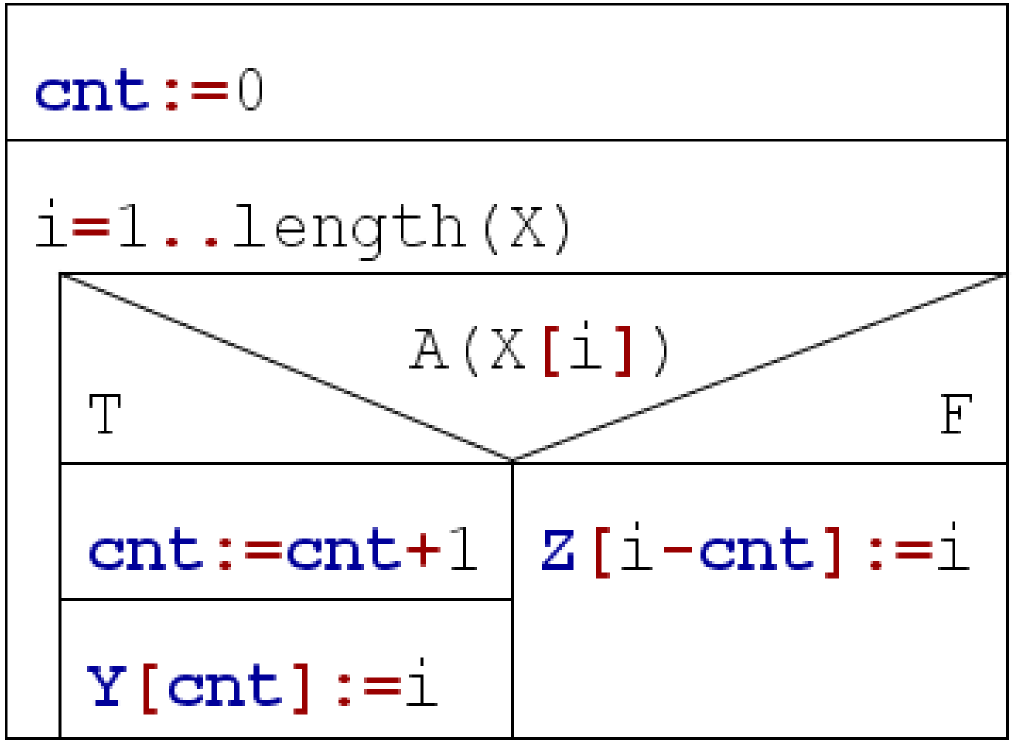
Postcondition: $$(cnt, Y, Z) = \underset{\substack{i=1\\ A(X[i])}}{\overset{length(X)}{\boldsymbol{PARTITION}}}\ (i)$$

For values: $$(cnt, Y, Z) = \underset{\substack{i=1\\ A(X[i])}}{\overset{length(X)}{\boldsymbol{PARTITION}}}\ (X[i])$$

Modify the original specification for „collecting values”

9. Partitioning

Algorithm



If we needed the values, then we would write `:=X[i]` instead of `:=i` (and accordingly specification should be modified).

9. Partitioning – in the same sequence

Specification (selecting indexes)

Input: `X[1..]`$\in\mathbb{S}^*$, `A:` $\mathbb{S}\to\mathbb{L}$

Output: `cnt`$\in\mathbb{N}$, `Y[1..]`$\in\mathbb{N}^*$

We will collect the A attributed elements at the beginning of the output array Y.

Precondition: –

Postcondition: $cnt = \sum_{\substack{i=1 \\ A(X[i])}}^{length(X)} 1$

and ∀i(1≤i≤cnt):A(X[Y[i]])

and ∀i(cnt+1≤i≤length(X)): not A(X[Y[i]])

and Y∈**Permutation(1,2,…,length(X))**

9. Partitioning – in the same sequence

Algorithm

```
cnt:=0
ind2:=length(X)+1
i=1..length(X)
    A(X[i])
    T                     F
    cnt:=cnt+1            ind2:=ind2-1
    Y[cnt]:=i             Y[ind2]:=i
```

Comment:
We can use an extra variable (ind2) to know where we are from back in Y.

9. Local Partitioning

Specification:

Input: X[1..]∈$\mathbb{S}^*$, A: $\mathbb{S}\to\mathbb{L}$

Output: cnt∈$\mathbb{N}$, X′[1..]∈$\mathbb{N}^*$

Precondition: –

Postcondition: $$cnt = \sum_{\substack{i=1 \\ A(X[i])}}^{length(X)} 1$$

and ∀i(1≤i≤cnt):A(X′[i])

and ∀i(cnt+1≤i≤length(X)): not A(X′[i])

and Y∈**Permutation (X)**

9. Local Partitioning

Main idea for the algorithm:

1. Pick out the **first** item of the sequence

O x x x x x x x x x x x

2. Let’s search an item from **back** which needs to be in the **front-sequence** (attribute A is true for it)

O x x x x x x **x** x x x x

3. Move the found item into the empty (first) place.

⊗ x x x x x x O x x x x

The first item and the items after the new empty place are at the right place.

9. Local Partitioning

4. Now we have an empty place at the end-sequence. From the **beginning** (starting after the first item) let’s search an item which needs to be in the end-sequence (attribute A is not true for it)

⊗ x x **x** x x x O x x x x

5. Move the founded item into the empty place. And we can search from back away.

⊗ x x O x x ⊗ x x x x x

The items before the new empty place and after the new place of the moved item are at right place.

9. Local Partitioning

6. … and so on …

7. We can finish searching if we reached the empty place from one direction.

   x x x x x O x x x x x x

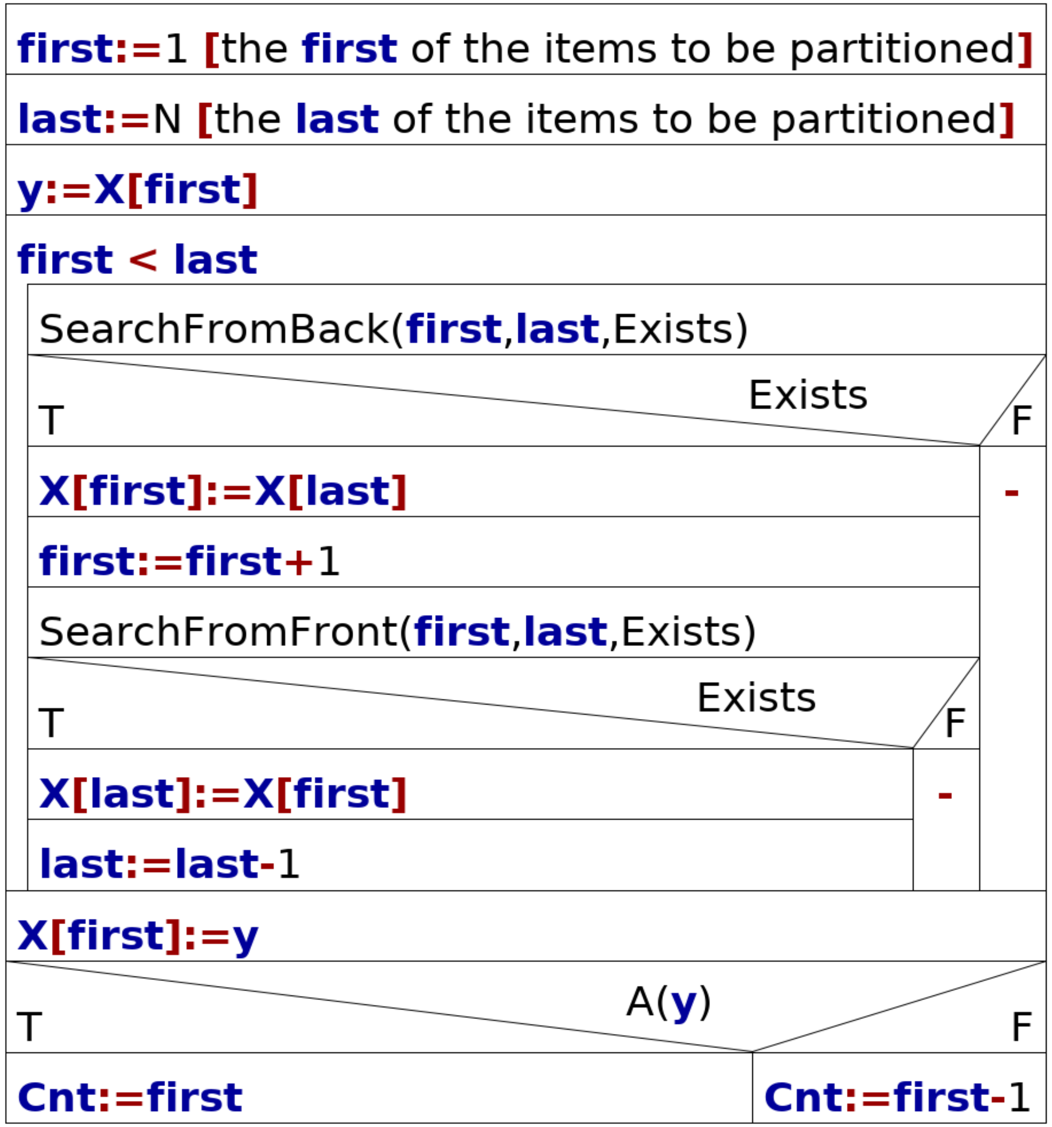
8. We put our first item back into this empty place.

9. Local Partitioning

Algorithm

What do we know about variable X?

- At the beginning: it is the input sequence
- At the end: it is the permutation of the original (input) sequence
- While „running”: to „first” are items with attribute A, from „last” items which are not of attribute A.



9. Local Partitioning

```
SearchFromFront(first,last:Integer,Exists:Boolean)
  first<last and A(X[first])
    first:=first+1
  Exists:=first<last
```

```
SearchFromBack(first,last:Integer,Exists:Boolean)
  first<last and not A(X[last])
    last:=last-1
  Exists:=first<last
```

10. Intersection

Tasks (examples):

1. Let's list all **common** divisors of **two** integers!
2. We investigate birds in winter **and** summer. Let's **collect** the birds which are not migratory!
3. Based on the free hours from the calendars of **two people**, let's **determine** when they can meet.
4. Let’s lists the animals which could be seen **both** in the zoos of Budapest and Veszprém.

10. Intersection

What is common?

We have two sets as input (with elements of the same type), and we have to list all elements that are part of both sets.

10. Intersection

Specification:

Input: X[1..]∈$\mathbb{S}^*$, Y[1..]∈$\mathbb{S}^*$,

Output: cnt∈$\mathbb{N}$, Z[1..]∈$\mathbb{N}^*$

Precondition: IsSet(X), IsSet(Y)

Postcondition: $cnt = \sum_{\substack{i=1 \\ X[i]\in Y}}^{length(X)} 1$

and ∀i(1≤i≤cnt):Z[i]∈X **and** Z[i]∈Y and IsSet(Z)

*: the minimum of the lengths of the input sequences:
MIN(length(X),length(Y))

Definition of IsSet function:
IsSet(x)=not ∃i(1≤i≤n):X[i]∈X[1..i-1]
IsSet(x)=i≠j → $x_i \neq x_j$

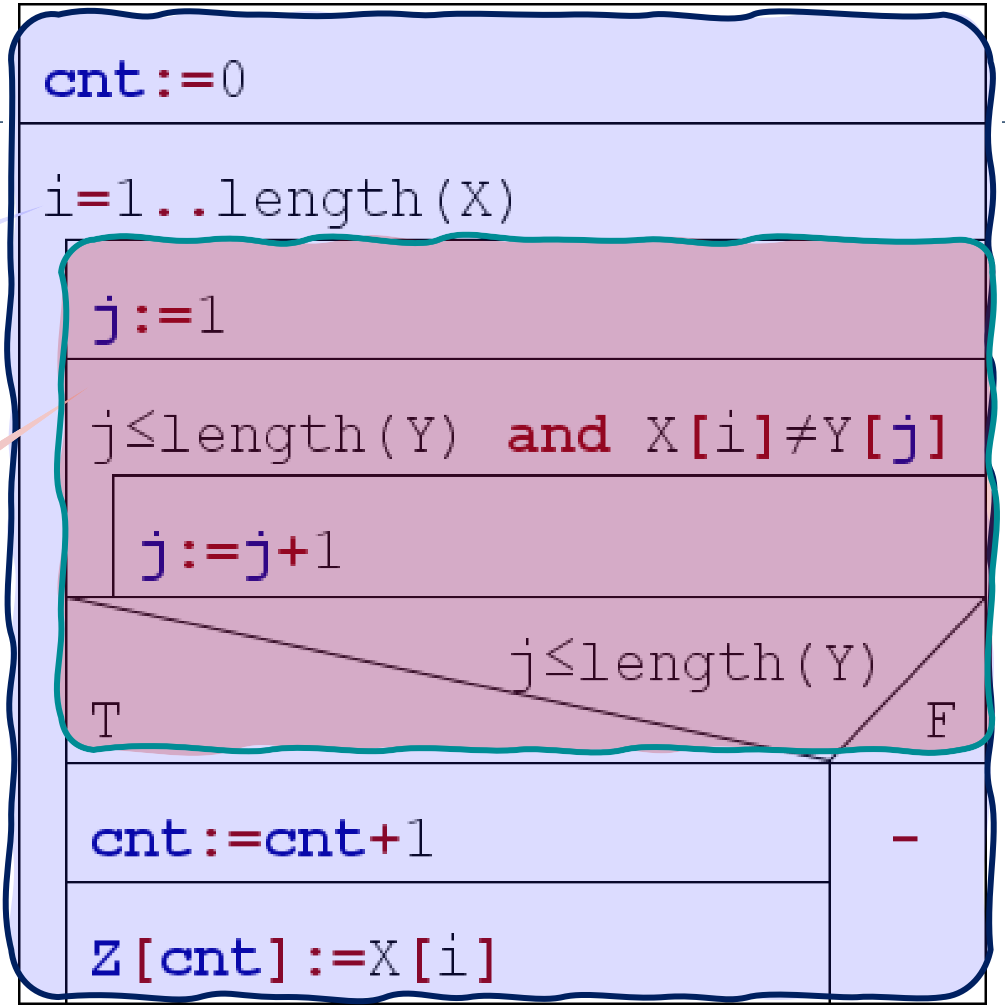
Comment:
SPECIAL Multiple item selection: A(X[i]) → X[i]∈Y

10. Intersection

Algorithm:

MULTIPLE ITEM SELECTION

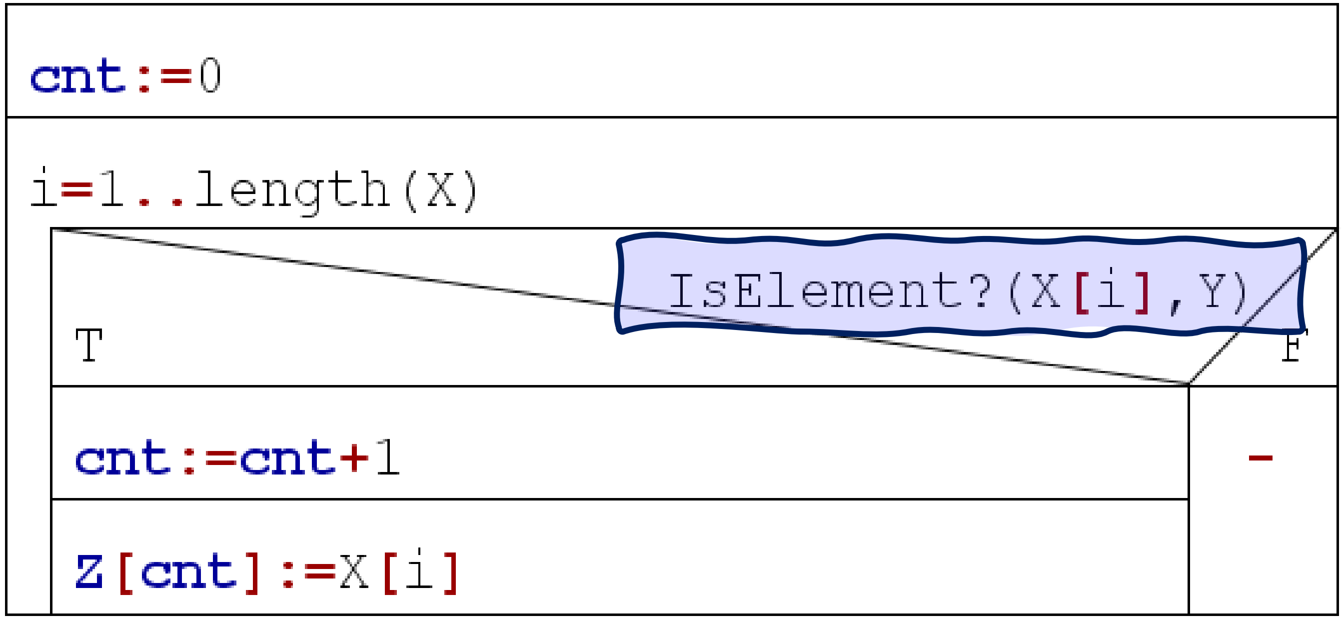
DECISION



SPECIAL Multiple item selection: A(X[i]) → X[i]∈Y

10. Intersection

Algorithm:



SPECIAL Multiple item selection: A(X[i]) → X[i]∈Y

10. Intersection

Variations for intersection:

- We have two sets, we have to determine the **count of common elements**.
- We have two sets, we have to determine if they have **at least one common element**.
- We have two sets, we have to give **one common element**.

Modify

the algorithm

11. Union

Tasks (examples):

1. We have two courses, let's **list** the students visiting **both** courses!
2. Let's **collect** all the birds we can observe in winter **as well as** in summer!
3. Based on the free hours from the calendars of two people, let's **determine** when we can reach **at least one of them**!
4. Let’s lists the animals which could be seen **either** in the zoos of Budapest **or** Veszprém.

11. Union

What is common?

We have two sets as input (with elements of the same type), and we have to list all elements that are included at least in one of the sets.

11. Union

*: the sum of the lengths of the input sequences: `length(X)+length(Y)`

Specification:

Input: `X[1..]`$\in\mathbb{S}^*$, `Y[1..]`$\in\mathbb{S}^*$,

Output: `cnt`$\in\mathbb{N}$, `Z[1..]`$\in\mathbb{N}^*$

Precondition: `IsSet(X), IsSet(Y)`

Postcondition:

$$cnt = length(X) + \sum_{\substack{i=1 \\ Y[i]\notin Y}}^{length(Y)} 1$$

`and ∀i(1≤i≤cnt):Z[i]∈X` **`or`** `Z[i]∈Y and IsSet(Z)`

Comment:

SPECIAL Multiple item selection: `A(Y[i])` $\rightarrow$ `Y[i]∉X`

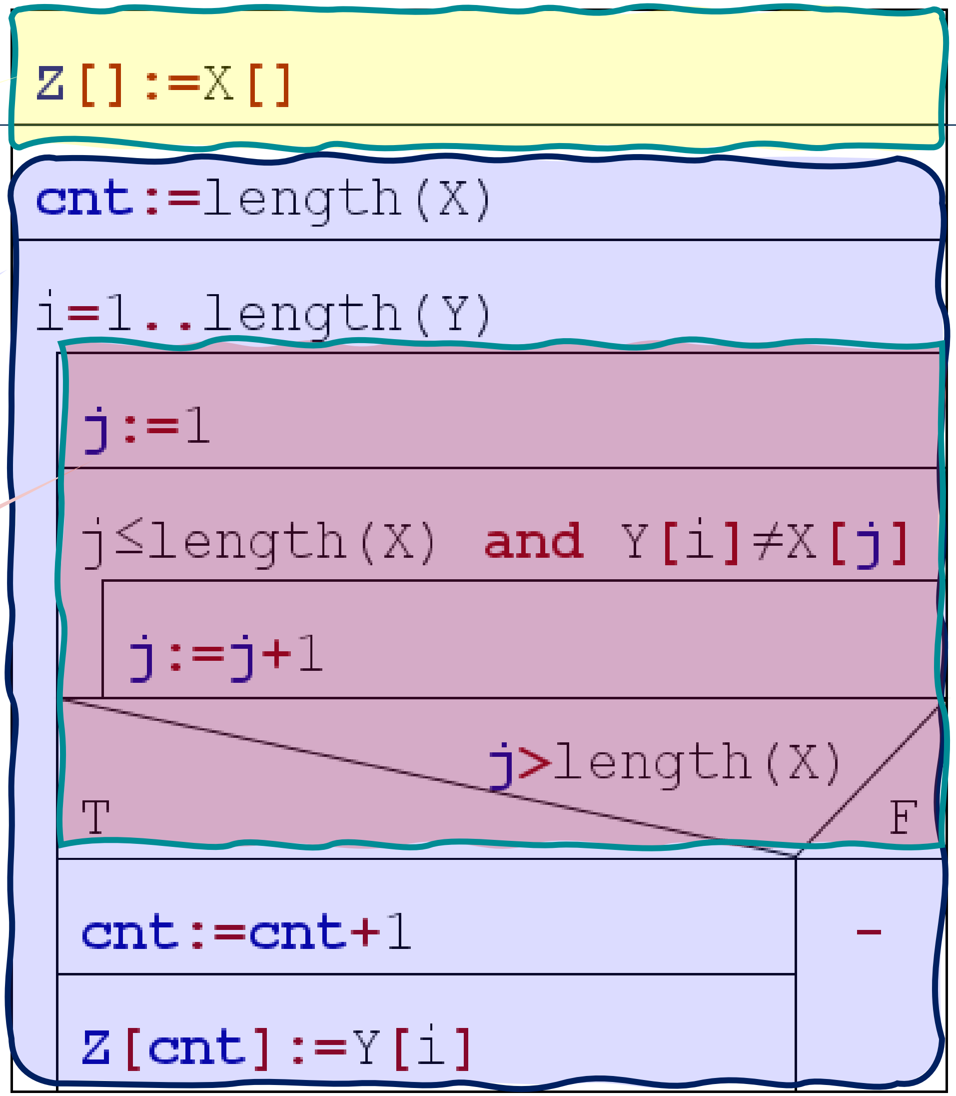
11. Union

Algorithm:

COPY

MULTIPLE ITEM SELECTION

DECISION



SPECIAL Multiple item selection: A(Y[i]) → Y[i]∉X

11. Union

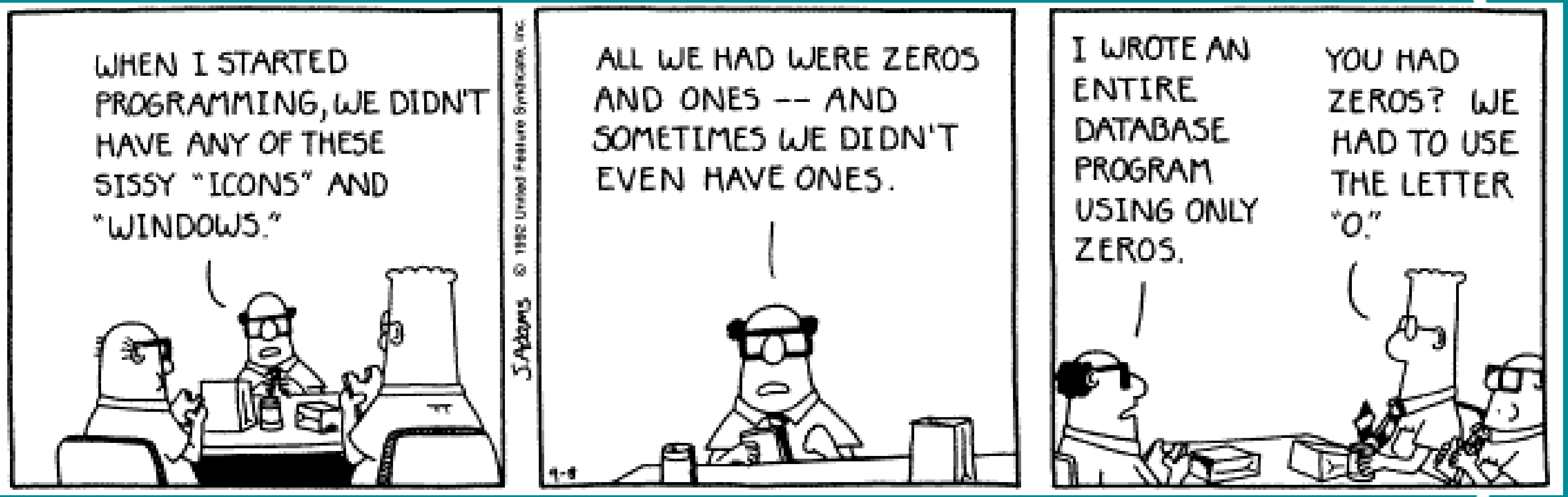
Variations for union:

- We have two sets, we have to determine the count of all the elements.
- We have two sets, we have to determine their difference ($X \setminus Y$).
- We have two sets, we have to give those elements which are only in one of the sets ($X \setminus Y \cup Y \setminus X$).

Modify

the algorithm

ELTE | FACULTY OF INFORMATICS



Thank you for your attention!