Algorithmic patterns over intervals

1. Summation

Problem: Let *H* be an arbitrary set on which an associative operation is defined, with a left-hand neutral element denoted by *0*. Let us call the operation addition and suppose that its operator is denoted by +. A function f:[$m..n$]→H is given. Let us calculate the sum of the values of *f* over the interval [*m..n*].

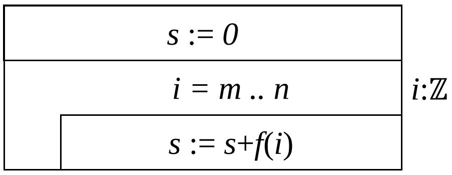
Specification:

$A = (m:\mathbb{Z}, n:\mathbb{Z}, s:H)$

$Pre = (m=m' \wedge n=n')$

$Post = (Pre \wedge s = \sum_{i=m..n} f(i))$

Algorithm:



2. Counting

Problem: Given a logical function *cond*:[*m..n*]→𝕃. Let us count the elements in the interval [*m..n*] for which *cond* condition holds.

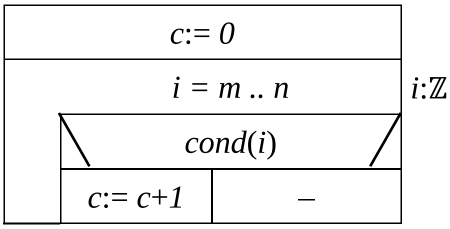
Specification:

$A = (m:\mathbb{Z}, n:\mathbb{Z}, c: \mathbb{N})$

$Pre = (m=m' \wedge n=n')$

$Post = (Pre \wedge c = \sum_{\substack{i=m..n \\ cond(i)}} 1)$

Algorithm:



3. Maximum search

Problem: Given a non-empty interval [*m..n*] and a function *f*:[*m..n*]→*H* where *H* is a totally ordered set. Let us search the maximal value of the function *f* over [*m..n*] and an argument where the function *f* gives back its maximal value.

Specification:

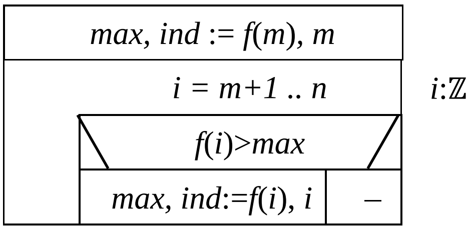
$A = (m:\mathbb{Z}, n:\mathbb{Z}, max:H, ind:\mathbb{Z})$

$Pre = (m=m' \wedge n=n' \wedge m\leq n)$

$Post = (Pre \wedge$

$(max,ind) = MAX_{i=m..n} f(i))$

Algorithm:



4. Conditional maximum search

Problem: Given a logical function $cond:[m..n]\rightarrow\mathbb{L}$ and an $f:[m..n]\rightarrow H$ function where H is a totally ordered set. Let us find the maximum value of the function among the outputs where a corresponding argument satisfies the condition *cond* and this argument is in interval $[m..n]$. The argument also has to be determined.

Specification:

$A = (m:\mathbb{Z},\ n:\mathbb{Z},\ l:\mathbb{L},\ ind:\mathbb{Z},\ max:H)$
$Pre = (\ m=m' \wedge n=n'\ )$
$Post = (\ Pre \wedge (l,max,ind) = \underset{cond(i)}{MAX_{i=m..n}} f(i)$

Algorithm:

```
$l := false$
$i = m .. n$                                          $i:\mathbb{Z}$
    $\neg cond(i)$ | $l \wedge cond(i)$ | $\neg l \wedge cond(i)$
    $\neg cond(i)$:
        SKIP
    $l \wedge cond(i)$:
        $f(i)>max$:
            $max, ind := f(i), i$
            | SKIP
    $\neg l \wedge cond(i)$:
        $l, max, ind := true, f(i), i$
```

Remark: Hereinafter, the flexibility of the above algorithmic patterns is presented.

1. Index *ind* can be left out if it is not needed in patterns
   - maximum search, linear search (decision)
2. Minimum search
   - In case of selection of the minimum value, instead of „>”, relational operator „<” has to be used. (In the specification: MIN function instead of MAX).
3. Searching for the last element
   - In case of maximum search: *f(i)≥max* instead of *f(i)>max*

5. Selection

Problem: Given a logical function $cond:[m..n]\rightarrow\mathbb{L}$ and an integer number *m*. Let us find the first integer number that is greater than or equal to *m* for which condition *cond* holds. It is assumed that there is such a kind of number.

Specification:

$A = (m:\mathbb{Z}, i:\mathbb{Z})$

$Pre = (m=m' \wedge \exists k\geq m: cond(k))$

$Post = (Pre \wedge i = SELECT_{i\geq m} cond(i))$

Algorithm:

```
$i := m$
$\neg cond(i)$
    $i := i+1$
```

6. Linear (or sequential) search

Problem: Given a logical function $cond:[m..n]\rightarrow\mathbb{L}$. Let us find the first argument in the interval $[m..n]$ for which the condition *cond* holds.

Specification:

$A = (m:\mathbb{Z}, n:\mathbb{Z}, l:\mathbb{L}, ind:\mathbb{Z})$

$Pre = (m=m' \wedge n=n')$

$Post = (Pre \wedge$

$(l, ind) = SEARCH_{i=m..n} cond(i))$

Algorithm:

```
$l, i := false, m$        $i:\mathbb{Z}$
$\neg l \wedge i\leq n$
    $l, ind := cond(i), i$
    $i := i+1$
```

The same algorithm can be applied in case of solving decision problems. In this case, you have to leave out variable *ind* and the related terms from the specification and from the algorithm, as well.

Typical problem for linear search: Is there any element in the interval for which a given condition holds?

The optimistic version of linear search examines, whether all elements satisfy the given condition, and determines the first index for which the condition does not hold.

The simpler version of optimistic linear search is optimistic decision, where variable *ind* and the related terms in the specification and in the algorithm are left out.

Specification:

$A = (m:\mathbb{Z}, n:\mathbb{Z}, l:\mathbb{L})$

$Pre = (m=m' \wedge n=n')$

$Post = (Pre \wedge$

$l = \forall SEARCH_{i=m..n} cond(i))$

Algorithm:

```
$l, i := true, m$        $i:\mathbb{Z}$
$l \wedge i\leq n$
    $l := cond(i)$
    $i := i+1$
```

Remark: Instead of searching for the first element satisfying a given condition, we can also search for the last one, too by replacing *i:=i+1*with *i:=i –1* in the algorithm.

7. Binary search

Problem: Given an f:[$m..n$]→H function monotonically increasing over the interval [$m..n$]. H is a totally ordered set. Let us decide whether f gives a certain value, and in case it does, an argument is also required to be given where the output of f equals to the given value.

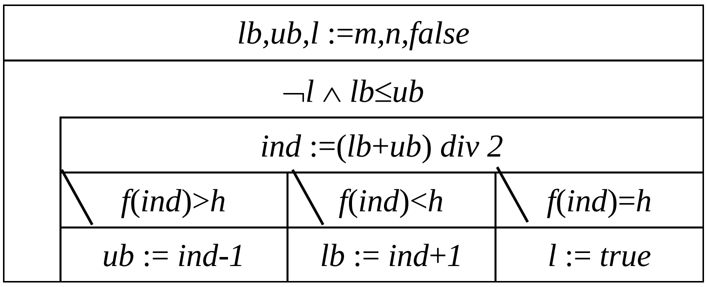
Specification:

$A = (m{:}\mathbb{Z},\ n{:}\mathbb{Z},\ h{:}H,\ l{:}\mathbb{L},\ ind{:}\mathbb{Z})$

$Pre = (\ m=m' \land n=n' \land h=h' \land \forall j\in[m..n\text{-}1]{:}\ f(j) \leq f(j+1)\ )$

$Post = (\ Pre \land l=(\ \exists j\in[m..n]{:}f(j)=h\ ) \land l\rightarrow(\ ind\in[m..n] \land f(ind)=h\ )\ )$

Algorithm:



```
lb,ub,l :=m,n,false
¬l ∧ lb≤ub
    ind :=(lb+ub) div 2
    \ f(ind)>h      | \ f(ind)<h      | \ f(ind)=h
      ub := ind-1   |   lb := ind+1   |   l := true
```

$lb,\ ub{:}\mathbb{Z}$

lb ~ lower-bound
ub ~ upper-bound