Minimum i maximum listy jednocześnie:

```
fun minmax [x] = (x,x)
  | minmax (x :: reszta) =
       let val (min, max) = minmax reszta
       in  if x<min then (x, max)
           else if x>max then (min, x)
                else (min, max)
       end;
val minmax = fn : int list -> int * int

- minmax [1, 2, 5, 3, 6];
val it = (1,6) : int * int
```

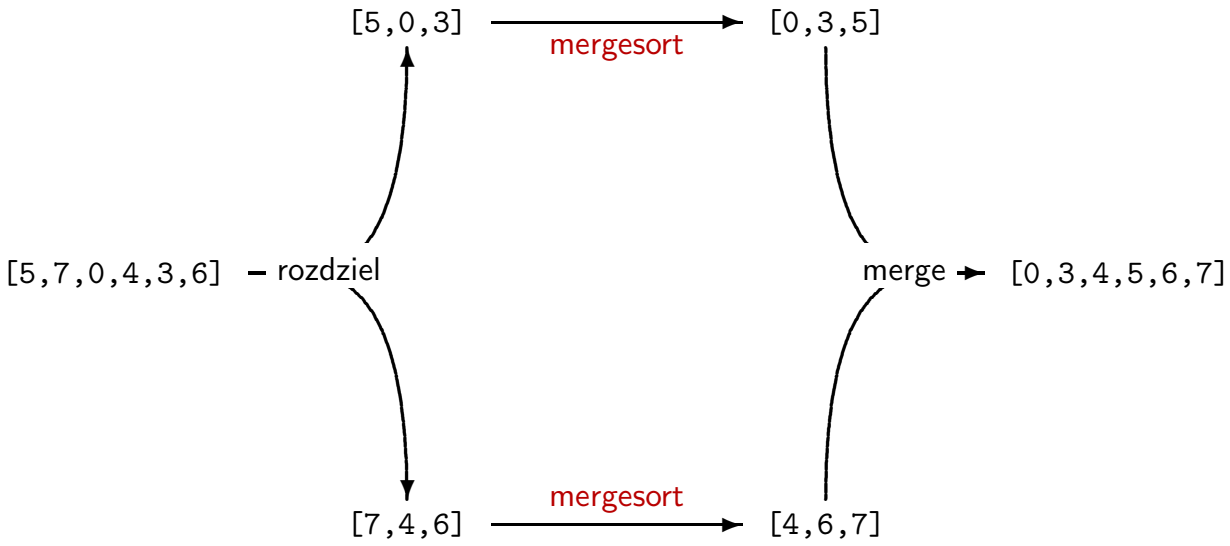

Konkatenacja list:

```
append [3, 4, 5] [1, 2]  =  [3, 4, 5, 1, 2]

fun append [ ] druga = druga
  | append (x::pierwsza) druga =
      x :: append pierwsza druga;
```

„Spłaszczanie" listy list:

```
flat [[1, 2], [ ], [5, 3, 6]]  =  [1, 2, 5, 3, 6]

fun flat [ ] = [ ]
  | flat (lis :: lislis) = append lis (flat lislis);
```

Sortowanie listy — Mergesort:



Języki programowania do sztucznej inteligencji

Sortowanie listy — Mergesort:

```
fun msort [ ] = [ ] | msort [x] = [x]
  | msort lista =
      let val (pierwsza, druga) = rozdziel lista
      in  merge (msort pierwsza, msort druga)  end;

fun rozdziel [ ] = ([ ], [ ])
  | rozdziel (x :: reszta) =
      let val (p,d) = rozdziel reszta
      in  (x::d, p)  end;

fun merge (pierwsza, [ ]) = pierwsza
  | merge ([ ], druga) = druga
  | merge (x :: pierwsza, y :: druga) =
      if  x < y  then  x :: merge (pierwsza, y::druga)
      else  y :: merge (x::pierwsza, druga);
```

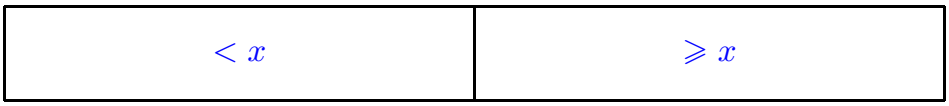
Sortowanie listy — Quicksort:

Mergesort:

1. rozdzielić ciąg
   byle jak
2. posortować rekursywnie obie części
3. scalić
   porównując elementy

Quicksort:

1. rozdzielić ciąg
   porównując elementy
2. posortować rekursywnie obie części
3. scalić
   byle jak



Języki programowania do sztucznej inteligencji

Sortowanie listy — Quicksort:

```
fun qsort lista = qsort1 lista [];

fun qsort1 [] ls = ls
  | qsort1 (x::lista) ls =
      let val (mniejsze, wkrown) = podziel x lista
      in  qsort1 mniejsze (x :: (qsort1 wkrown ls))
      end;

fun podziel x [] = ([], [])
  | podziel x (y::reszta) =
      let val (mniejsze, wkrown) = podziel x reszta
      in  if y<x then  (y::mniejsze, wkrown)
          else  (mniejsze, y::wkrown)
      end;

qsort [5,7,0,4,3,6];
```

Programowanie logiczne:

Język PROLOG (od lat 1980-tych) i jego dialekty.

W Prologu **nie ma funkcji**, są tylko *predykaty*.

W SML pytamy:	*Ile to jest* $2+1$*?*	`2+1;`	`3`
W Prologu pytamy:	*Czy* $2+1=4$*?*	`plus(2,1,4).`	`No`
	Czy $2+1=3$*?*	`plus(2,1,3).`	`Yes`
Synonimy:	*Czy* $2+1=4$*?*	`4 is 2+1.`	`No`
	Czy $2+1=3$*?*	`3 is 2+1.`	`Yes`
Zmienne:	*Czy* $2+1=X$*?*	`X is 2+1.`	`X = 3`
	Czy $2+X=1$*?*	`plus(2,X,1).`	`X = -1`