Tri par insertion

```
for i = 2:n,
    for (k = i; k > 1 and a[k] < a[k-1]; k--)
        swap a[k,k-1]
    → invariant: a[1..i] is sorted
end
```

Tri par sélection

```
for i = 1:n,
    k = i
    for j = i+1:n, if a[j] < a[k], k = j
    → invariant: a[k] smallest of a[i..n]
    swap a[i,k]
    → invariant: a[1..i] in final position
end
```

Tri à bulles

```
for i = 1:n,
    swapped = false
    for j = n:i+1,
        if a[j] < a[j-1],
            swap a[j,j-1]
            swapped = true
    → invariant: a[1..i] in final position
    break if not swapped
end
```

Tri de Shell

```
h = 1
while h < n, h = 3*h + 1
while h > 0,
    h = h / 3
    for k = 1:h, insertion sort a[k:h:n]
    → invariant: each h-sub-array is sorted
end
```

Tri par fusion

```
# split in half
m = n / 2

# recursive sorts
sort a[1..m]
sort a[m+1..n]

# merge sorted sub-arrays using temp array
b = copy of a[1..m]
i = 1, j = m+1, k = 1
while i <= m and j <= n,
    a[k++] = (a[j] < b[i]) ? a[j++] : b[i++]
    → invariant: a[1..k] in final position
while i <= m,
    a[k++] = b[i++]
    → invariant: a[1..k] in final position
```

Tri par tas

```
# heapify
for i = n/2:1, sink(a,i,n)
→ invariant: a[1,n] in heap order

# sortdown
for i = 1:n,
    swap a[1,n-i+1]
    sink(a,1,n-i)
    → invariant: a[n-i+1,n] in final position
end

# sink from i in a[1..n]
function sink(a,i,n):
    # {lc,rc,mc} = {left,right,max} child index
    lc = 2*i
    if lc > n, return # no children
    rc = lc + 1
    mc = (rc > n) ? lc : (a[lc] > a[rc]) ? lc : rc
    if a[i] >= a[mc], return # heap ordered
    swap a[i,mc]
    sink(a,mc,n)
```

Tri rapide

```
_# choose pivot_
swap a[1,rand(1,n)]

_# 2-way partition_
k = 1
for i = 2:n, if a[i] < a[1], swap a[++k,i]
swap a[1,k]
_→ invariant: a[1..k-1] < a[k] <= a[k+1..n]_

_# recursive sorts_
sort a[1..k-1]
sort a[k+1,n]
```

Tri rapide (3 partitions)

```
_# choose pivot_
swap a[n,rand(1,n)]

_# 3-way partition_
i = 1, k = 1, p = n
while i < p,
  if a[i] < a[n], swap a[i++,k++]
  else if a[i] == a[n], swap a[i,--p]
  else i++
end
_→ invariant: a[p..n] all equal_
_→ invariant: a[1..k-1] < a[p..n] < a[k..p-1]_

_# move pivots to center_
m = min(p-k,n-p+1)
swap a[k..k+m-1,n-m+1..n]

_# recursive sorts_
sort a[1..k-1]
sort a[n-p+k+1,n]
```