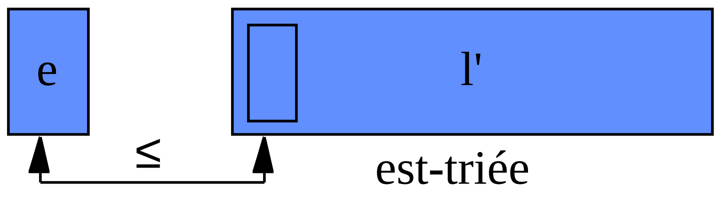
Généralités sur le tri et méthodes simples

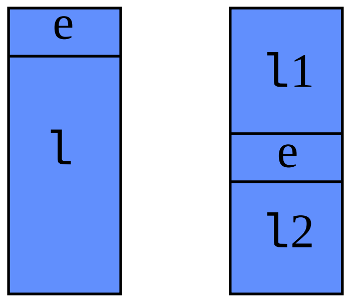
Spécification du tri (1)

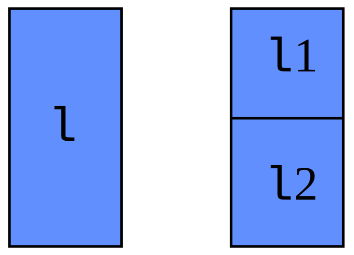
extension type liste
avec
    _ ≤ _ : élément × élément → booléen
opérations
    est-triée : liste → booléen
sémantique
    est-triée(l) = **si** l = listevide **alors** vrai
                sinon soit l = e::l'; e ≤ premier(l')   est-triée(l') **fsi**



Spécification du tri (2)

```
extension type liste
opérations
    _   _          : élément × liste → booléen
    est-permut     : liste × liste → booléen
sémantique
    e    l = si estvide (l) alors faux
             sinon e = premier(l)    e    fin(l) fsi
    est-permut(l , l ') = si estvide (l) alors estvide(l ')
                          sinsi ¬ premier(l)    l ' alors faux
                          sinon soit l ' = l 1 & [premier(l)] & l 2;
                                est-permut (fin(l), l 1 & l 2) fsi
```





Spécification du tri (3)

- opération de tri:

  opérations

  tri : liste → liste

  axiomes

  l: liste

  est-permut(l, tri(l)) est-triée(tri(l))

- rien ne dit comment l'obtenir
- résultat peut différer d'une implantation à l'autre:
  - a ≤ b et b ≤ a avec a ≠ b

quelle relation d'ordre

- spécification Ada:

```
generic
  with function "<" (U, V : in Element) return Boolean is <>;
procedure Listes_Contigues.Tri_Liste (La_Liste: in out Liste);
```

modification sur place

Tri par sélection (1)

extension type liste

opérations

min	: liste /→ élément
supprimer-min	: liste /→ liste
tri-sélect	: liste → liste

préconditions

min(l): ¬ estvide(l)

supprimer-min(l): ¬ estvide(l)

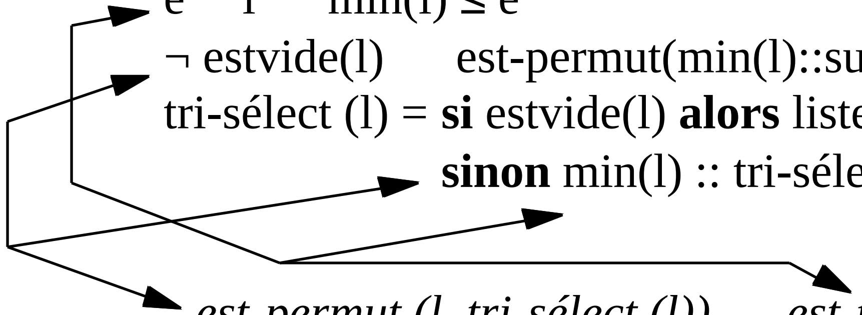
axiomes

e: élément, l: liste

e l min(l) ≤ e

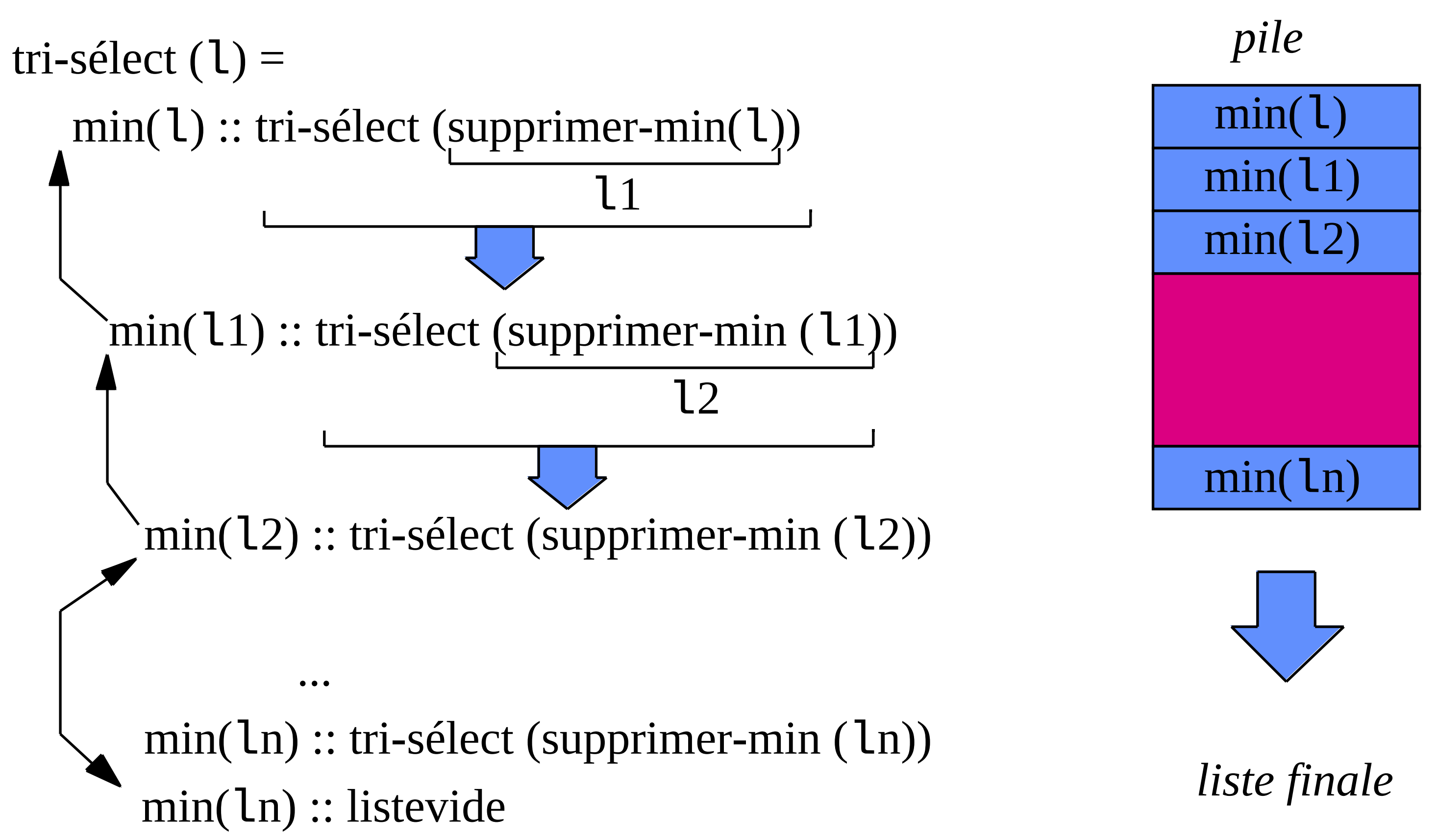
¬ estvide(l) est-permut(min(l)::supprimer-min(l), l)

tri-sélect (l) = **si** estvide(l) **alors** listevide
sinon min(l) :: tri-sélect(supprimer-min(l)) **fsi**



est-permut (l, tri-sélect (l)) *est-triée (tri-sélect (l))*

Tri par sélection (2)



```
tri-sélect (l) =
    min(l) :: tri-sélect (supprimer-min(l))
      min(l1) :: tri-sélect (supprimer-min (l1))
        min(l2) :: tri-sélect (supprimer-min (l2))
              ...
        min(ln) :: tri-sélect (supprimer-min (ln))
        min(ln) :: listevide
```

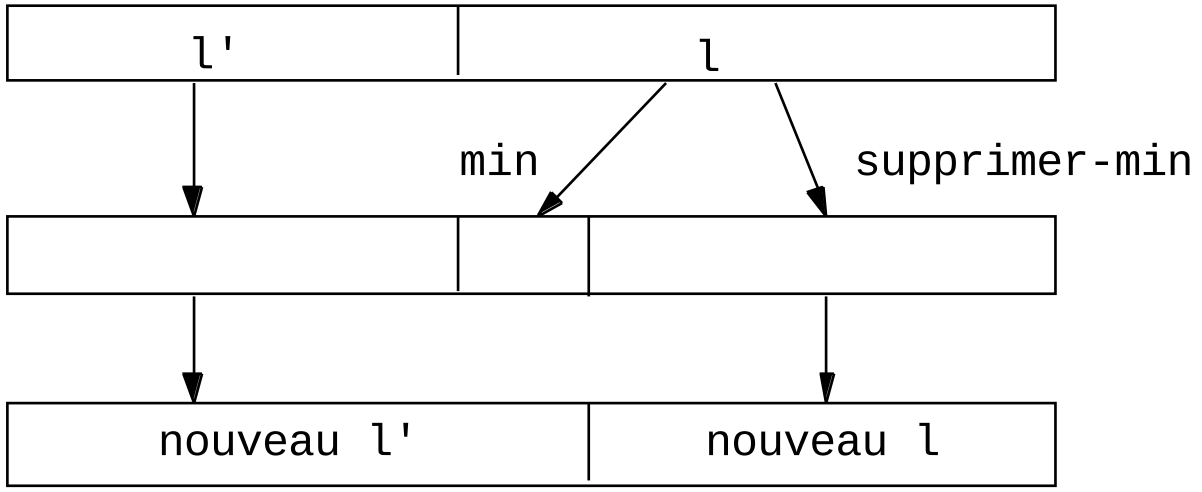
Tri par sélection (3)

pile dans l'algorithme récursif

```
extension type liste
opérations
    tri-sélect-iter : liste × liste → liste
axiomes
    tri-sélect-iter(l', l) = si l = listevide alors l'
                             sinon tri-sélect-iter(l'&[min(l)], supprimer-min(l)) fsi
```

tri-sélect (l) = tri-sélect-iter (listevide, l)

implantation sur place



Sélection ordinaire (1)

```
for K in 1 .. La_Liste.Longueur - 1 loop
    J := K;                    -- premier candidat possible
    for L in K + 1 .. La_Liste.Longueur loop
        if La_Liste.Ieme (L) < La_Liste.Ieme (J) then
            J := L;                     -- candidat meilleur
        end if;
    end loop;
    Echanger (J, K);
end loop;
```

- comparaisons: $\sum_{p=2}^{n}(p-1) = \sum_{p=1}^{n-1} p = \frac{n*(n-1)}{2}$ $\Theta\,(n^2)$
- transferts: 3 (n - 1) $\Theta\,(n)$
- place mémoire supplémentaire: $\Theta(1)$

Sélection ordinaire (2)

[]	$\underline{55}$, 40, 15, 30, 10, $\boxed{5}$, 25, 35, 60, 20, 45, 50
[5]	$\underline{40}$, 15, 30, $\boxed{10}$, 55, 25, 35, 60, 20, 45, 50
[5, 10]	$\boxed{\underline{15}}$, 30, 40, 55, 25, 35, 60, 20, 45, 50
[5, 10, 15]	$\underline{30}$, 40, 55, 25, 35, 60, $\boxed{20}$, 45, 50
[5, 10, 15, 20]	$\underline{40}$, 55, $\boxed{25}$, 35, 60, 30, 45, 50
[5, 10, 15, 20, 25]	$\underline{55}$, 40, 35, 60, $\boxed{30}$, 45, 50
[5, 10, 15, 20, 25, 30]	$\underline{40}$, $\boxed{35}$, 60, 55, 45, 50
[5, 10, 15, 20, 25, 30, 35]	$\boxed{\underline{40}}$, 60, 55, 45, 50
[5, 10, 15, 20, 25, 30, 35, 40]	$\underline{60}$, 55, $\boxed{45}$, 50
[5, 10, 15, 20, 25, 30, 35, 40, 45]	$\underline{55}$, 60, $\boxed{50}$
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50]	$\underline{60}$, $\boxed{55}$
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55]	60

Méthode de la bulle (1)

```
for K in 1 .. La_Liste.Longueur - 1 loop
   for L in reverse K+1 .. La_Liste.Longueur loop
      if La_Liste.Ieme (L) < La_Liste.Ieme (L - 1) then
          Echanger(L - 1, L); -- L est meilleur candidat que L-1
      end if;
    end loop;
end loop;
```

- comparaisons: $\sum_{p=2}^{n}(p-1) = \sum_{p=1}^{n-1} p = \frac{n*(n-1)}{2}$ $\Theta(n^2)$
- transferts: au pire 3*comparaisons $\Theta(n^2)$

  en moyenne 3 n (n-1)/4 $\Theta(n^2)$
- place mémoire supplémentaire: $\Theta(1)$

Méthode de la bulle(2)

[] 55, 40, 15, 30, 10, 5, 25, 35, $\boxed{60, 20,}$ 45, 50

[] 55, 40, 15, 30, 10, 5, 25, $\boxed{35, 20,}$ 60, 45, 50

[] 55, 40, 15, 30, 10, 5, $\boxed{25, 20,}$ 35, 60, 45, 50

[] 55, 40, 15, 30, $\boxed{10, 5,}$ 20, 25, 35, 60, 45, 50

[] 55, 40, 15, $\boxed{30, 5,}$ 10, 20, 25, 35, 60, 45, 50

[] 55, 40, $\boxed{15, 5,}$ 30, 10, 20, 25, 35, 60, 45, 50

[] 55, $\boxed{40, 5,}$ 15, 30, 10, 20, 25, 35, 60, 45, 50

[] $\boxed{55, 5,}$ 40, 15, 30, 10, 20, 25, 35, 60, 45, 50

[5] 55, 40, 15, 30, 10, 20, 25, 35, $\boxed{60, 45,}$ 50

[5] 55, 40, 15, $\boxed{30, 10,}$ 20, 25, 35, 45, 60, 50

[5] 55, 40, $\boxed{15, 10,}$ 30, 20, 25, 35, 45, 60, 50

[5] 55, 40, 10, 15, 30, 20, 25, 35, 45, 60, 50

Méthode de la bulle(3)

[5] 55, $\boxed{40, 10}$, 15, 30, 20, 25, 35, 45, 60, 50

[5] $\boxed{55, 10}$, 40, 15, 30, 20, 25, 35, 45, 60, 50

[5, 10] 55, 40, 15, 30, 20, 25, 35, 45, $\boxed{60, 50}$

[5, 10] 55, 40, 15, $\boxed{30, 20}$, 25, 35, 45, 50, 60

[5, 10] 55, $\boxed{40, 15}$, 20, 30, 25, 35, 45, 50, 60

[5, 10] $\boxed{55, 15}$, 40, 20, 30, 25, 35, 45, 50, 60

[5, 10, 15] 55, 40, 20, $\boxed{30, 25}$, 35, 45, 50, 60

[5, 10, 15] 55, $\boxed{40, 20}$, 25, 30, 35, 45, 50, 60

[5, 10, 15] $\boxed{55, 20}$, 40, 25, 30, 35, 45, 50, 60

[5, 10, 15, 20] 55, $\boxed{40, 25}$, 30, 35, 45, 50, 60

[5, 10, 15, 20] $\boxed{55, 25}$, 40, 30, 35, 45, 50, 60

[5, 10, 15, 20, 25] 55, 40, 30, 35, 45, 50, 60

Méthode de la bulle(4)

[5, 10, 15, 20, 25] 55, $\boxed{40, 30}$, 35, 45, 50, 60

[5, 10, 15, 20, 25] $\boxed{55, 30}$, 40, 35, 45, 50, 60

[5, 10, 15, 20, 25, 30] 55, $\boxed{40, 35}$, 45, 50, 60

[5, 10, 15, 20, 25, 30] $\boxed{55, 35}$, 40, 45, 50, 60

[5, 10, 15, 20, 25, 30, 35] $\boxed{55, 40}$, 45, 50, 60

[5, 10, 15, 20, 25, 30, 35, 40] $\boxed{55, 45}$, 50, 60

[5, 10, 15, 20, 25, 30, 35, 40, 45] $\boxed{55, 50}$, 60

[5, 10, 15, 20, 25, 30, 35, 40, 45, 50] 55, 60

[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55] 60

Tri par insertion

extension type liste
opérations

placer	: élément × liste /→ liste
tri-insert-iter	: liste × liste /→ liste
tri-insert	: liste → liste

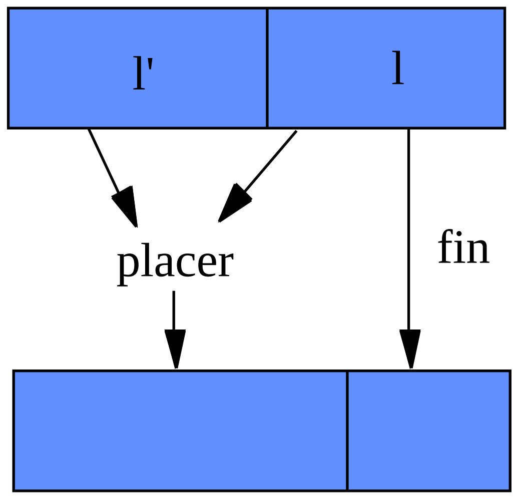
préconditions

placer(e, l): est-triée(l)
tri-insert-iter(l', l): est-triée(l')

sémantique

e: élément, l: liste
est-permut(e::l, placer(e, l)) est-triée(placer(e, l))
tri-insert-iter(l', l) = **si** estvide (l) **alors** l'
sinon tri-insert-iter(placer(premier(l), l'), fin(l)) **fsi**
tri-insert(l) = tri-insert-iter(listevide, l)

est-permut (l, tri-insert (l)) est-triée (tri-insert (l))



Insertion séquentielle (1)

```
for K in 2 .. La_Liste.Longueur loop
    U := La_Liste.Ieme (K);            -- placement du Kième
    L := K;                            -- première place possible
    while L /= 1 and then U < La_Liste.Ieme (L - 1) loop
        La_Liste.Ieme (L) := La_Liste.Ieme (L - 1);   L := L - 1;
    end loop;
    La_Liste.Ieme (L) := U;
end loop;
```

- comparaisons: au mieux, n-1 $\Theta(n)$

  au pire, n (n-1)/2 $\Theta(n^2)$

  en moyenne, n (n+3)/4 $\Theta(n^2)$
- transferts: nombre de comparaisons + n-1
- place mémoire supplémentaire: $\Theta(1)$

Insertion séquentielle (2)

[] 55, 40, 15, 30, 10, 5, 25, 35, 60, 20, 45, 50

[55, --] 15, 30, 10, 5, 25, 35, 60, 20, 45, 50

[40, 55, --] 30, 10, 5, 25, 35, 60, 20, 45, 50

[40, --, 55] 30, 10, 5, 25, 35, 60, 20, 45, 50

[15, 40, 55, --] 10, 5, 25, 35, 60, 20, 45, 50

[15, 40, --, 55] 10, 5, 25, 35, 60, 20, 45, 50

[15, 30, 40, 55, --] 5, 25, 35, 60, 20, 45, 50

[15, 30, 40, --, 55] 5, 25, 35, 60, 20, 45, 50

[15, 30, --, 40, 55] 5, 25, 35, 60, 20, 45, 50

[15, --, 30, 40, 55] 5, 25, 35, 60, 20, 45, 50

[10, 15, 30, 40, 55] 5, 25, 35, 60, 20, 45, 50

Insertion séquentielle (3)

```
[10, 15, 30, 40, 55]     5, 25, 35, 60, 20, 45, 50
[10, 15, 30, 40, 55,  --]   25, 35, 60, 20, 45, 50
[10, 15, 30, 40,  --, 55]   25, 35, 60, 20, 45, 50
[10, 15, 30,  --, 40, 55]   25, 35, 60, 20, 45, 50
[10, 15,  --, 30, 40, 55]   25, 35, 60, 20, 45, 50
[10,  --, 15, 30, 40, 55]   25, 35, 60, 20, 45, 50
[  5, 10, 15, 30, 40, 55,  --]   35, 60, 20, 45, 50
[  5, 10, 15, 30, 40,  --, 55]   35, 60, 20, 45, 50
[  5, 10, 15, 30,  --, 40, 55]   35, 60, 20, 45, 50
[  5, 10, 15, 25, 30, 40, 55,  --]   60, 20, 45, 50
[  5, 10, 15, 25, 30, 40,  --, 55]   60, 20, 45, 50
[  5, 10, 15, 25, 30, 35, 40, 55]   60, 20, 45, 50
```

Insertion séquentielle (4)

[5, 10, 15, 25, 30, 35, 40, 55] 60, 20, 45, 50

[5, 10, 15, 25, 30, 35, 40, 55, --] 20, 45, 50

[5, 10, 15, 25, 30, 35, 40, 55, $\underline{60}$, --] 45, 50

[5, 10, 15, 25, 30, 35, 40, $\underline{55}$, --, 60] 45, 50

[5, 10, 15, 25, 30, 35, $\underline{40}$, --, 55, 60] 45, 50

[5, 10, 15, 25, 30, $\underline{35}$, --, 40, 55, 60] 45, 50

[5, 10, 15, 25, $\underline{30}$, --, 35, 40, 55, 60] 45, 50

[5, 10, 15, $\underline{25}$, --, 30, 35, 40, 55, 60] 45, 50

[5, 10, 15, 20, 25, 30, 35, 40, 55, $\underline{60}$, --] 50

[5, 10, 15, 20, 25, 30, 35, 40, $\underline{55}$, --, 60] 50

[5, 10, 15, 20, 25, 30, 35, 40, 45, 55, $\underline{60}$, --]

[5, 10, 15, 20, 25, 30, 35, 40, 45, $\underline{55}$, --, 60]

[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]

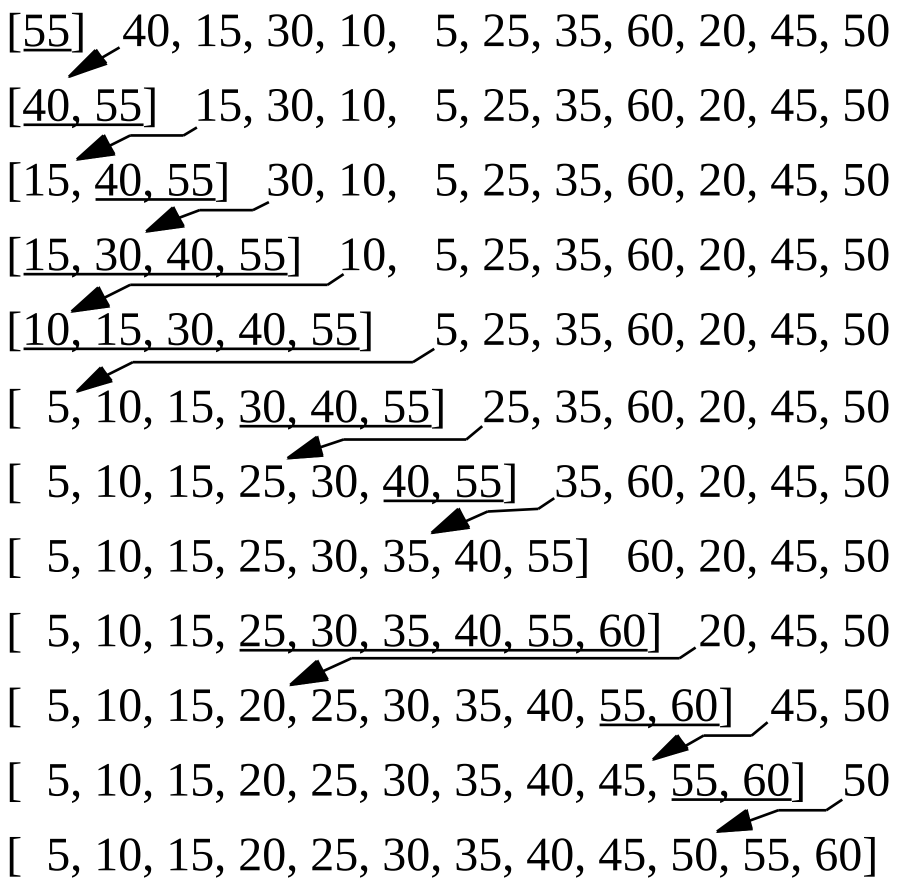
Insertion dichotomique (1)

```
for K in 2 .. La_Liste.Longueur loop
    if La_Liste.Ieme (K) < La_Liste.Ieme (K - 1) then
        U := La_Liste.Ieme (K);                          -- il n'est pas à sa place
        L := Recherche_Dicho (K - 1);                    -- recherche de sa place
        La_Liste.Ieme(L+1..K) := La_Liste.Ieme(L..K-1);     -- déplacements
        La_Liste.Ieme (L) := U;                              -- placement
    end if;
end loop;
```

- comparaisons: au mieux, n-1 $\Theta(n)$

  au pire et en moyenne , n $\log_2 n$ $\Theta(n \log n)$
- transferts: idem insertion séquentielle
  - au mieux $\Theta(n)$, en moyenne et au pire $\Theta(n^2)$
- place mémoire supplémentaire: $\Theta(1)$

Insertion dichotomique (2)



[55] 40, 15, 30, 10, 5, 25, 35, 60, 20, 45, 50

[40, 55] 15, 30, 10, 5, 25, 35, 60, 20, 45, 50

[15, 40, 55] 30, 10, 5, 25, 35, 60, 20, 45, 50

[15, 30, 40, 55] 10, 5, 25, 35, 60, 20, 45, 50

[10, 15, 30, 40, 55] 5, 25, 35, 60, 20, 45, 50

[5, 10, 15, 30, 40, 55] 25, 35, 60, 20, 45, 50

[5, 10, 15, 25, 30, 40, 55] 35, 60, 20, 45, 50

[5, 10, 15, 25, 30, 35, 40, 55] 60, 20, 45, 50

[5, 10, 15, 25, 30, 35, 40, 55, 60] 20, 45, 50

[5, 10, 15, 20, 25, 30, 35, 40, 55, 60] 45, 50

[5, 10, 15, 20, 25, 30, 35, 40, 45, 55, 60] 50

[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]