```
procedure Join(A,B)
if A=nil then return B
        else if B=nil then return A;
if key(B)<key(A) then Swap(A,B);
right(A):=Join(right(A),B);
if dist(right(A))>dist(left(A)) then Swap(right(A),left(A));
if right(A)=null then dist(A):=1
        else dist(A):=1+dist(right(A));
return A;
```

Złożoność operacji Join wynosi O(dist(A)+dist(B)), czyli O(log n).

Ini i Min tworzymy standardowo i działają w stałym czasie.

```
procedure Insert(r,x)
p:=Ini(x);
r:=Join(r,p);
```

Koszt: O(log n).

Build - po kolei wykonujemy Insert, albo szybciej - tworzymy zwykły kopiec, który też jest lewicowy. Koszt:O(n)

```
procedure DeleteMin(r)
min:=r.value;
r:=Join(left(r),right(r));
return min;
```

Koszt: O(log n).

procedure RemoveNode(v)

p:=v.parent;

zastąp v przez Join(left(v), right(v));

while p≠nil **do**

begin

if dist(left(p)) < dist(right(p)) **then** Swap(left(p), right(p));

if dist(p) = dist(right(p))+1 **then** break

else dist(p):= dist(right(p))+1; p:=p.parent;

end;

W pętli **while**:

- jeśli v było w left(p), to w przypadku wykonania Swap drzewo o korzeniu w p musi mieć co najmniej 2^k węzłów (gdzie k jest odległością miedzy v i p), bo k-krotnie zwiększaliśmy wartość dist dla kolejnych węzłów (inaczej procedura zatrzymałaby się).

- jeśli v było w right(p), to liczba wywołań pętli nie przekracza O(log n) (długość skrajnej ścieżki).

Zatem czas wykonania procedury jest logarytmiczny.

procedure DecreaseKey

RemoveNode(v);

Insert(r,v'); Koszt: O(log n).