```
# #load "bintree.cmo";;
# open Bintree;;
# let t = bst (<=) [13; 40; 20; 10; 5; 16; 8; 4; 2; 1];;
val t : int Mi_bintree.bintree =
  Node (... ...)
# in_order t;;
- : int list = [1; 2; 4; 5; 8; 10; 13; 16; 20; 40]
# let rlist = List.init 100_000 (fun _ -> Random.int 100_000);;
val rlist : int list =
  [9344; 26685; 80182; 31641; 80439; 7500; 44104; 7020; 19921; 78370; 55217; ...]
# qsort (<=) rlist = List.sort compare rlist;;
- : bool = true
# let crono f x =
      let t = Sys.time () in f x; Sys.time () -. t;;
val crono : ('a -> 'b) -> 'a -> float = <fun>
# crono (List.sort compare) rlist;;
- : float = 0.0943319999999999714
# crono (qsort (<=)) rlist;;
- : float = 0.133970999999999951

# #load "bintree_opt.cmo";;
# open Bintree_opt;;
# let rec f x = function
    0 -> Empty |   n -> Node (x, f (2*x) (n-1), f (2*x+1) (n-1));;
val f : int -> int -> int Bintree.bintree = <fun>
# bfs (f 1 10) = List.init 1023 succ;;
- : bool = true
# bfs' (f 1 10) = List.init 1023 succ;;
- : bool = true
# crono bfs (f 1 14);;
- : float = 3.41552400000000489
# crono bfs' (f 1 14);;
- : float = 0.00509399999999971
# let tree = bst (<=) rlist;;
val tree : int Mi_bintree.bintree =
  Node (... ...)
# is_bst (<=) tree;;
- : bool = true
# let tree2 = insert (<=) tree 10_000;;
val tree2 : int Mi_bintree.bintree = Node (... ...)
# is_bst (<=) tree2;;
- : bool = true
# let tree3 = insert (>=) tree2 10_001;;
val tree3 : int Mi_bintree.bintree = Node (... ...)
# is_bst (<=) tree3;;
- : bool = false
# crono (is_bst (<=)) tree2;;
- : float = 0.0138899999999999579
# crono (is_bst (<=)) tree3;;
- : float = 1.00000000000065512e-05
# perfecto tree;;
- : bool = false
# casi_completo tree;;
- : bool = false
```

```
# #load "act.cmo";;
# open Act;;
# let an_act n = act (List.init n abs);;
val an_act : int -> int Bintree.bintree = <fun>
# List.for_all casi_completo (List.map an_act (List.init 16 abs));;
- : bool = true
# List.map perfecto (List.map an_act (List.init 16 abs));;
- : bool list =
[true; true; false; true; false; false; false; true; false; false; false;
 false; false; false; false; true]
# let big = an_act (power2 20);;
val big2 : int Bintree.bintree =
  Node (0,
   ...)
# let big1 = an_act (power2 20 - 1);;
val big2 : int Bintree.bintree =
  Node (0,
   ...)
# let big2 = an_act (power2 20 - 2);;
val big2 : int Bintree.bintree =
  Node (0,
   ...)
# List.map perfecto [big; big1; big2];;
- : bool list = [false; true; false]
# List.map casi_completo [big; big1; big2];;
- : bool list = [true; true; true]
# List.map (crono casi_completo) [big; big1; big2];;
- : float list =
[0.548848000000000447; 0.523184999999999789; 0.52183800000000069]
# let swap = function
      Empty -> Empty | Node (r,i,d) -> Node (r,d,i);;
val swap : 'a Bintree.bintree -> 'a Bintree.bintree = <fun>
# List.map perfecto (List.map swap [big; big1; big2]);;
- : bool list = [false; true; false]
# List.map casi_completo (List.map swap [big; big1; big2]);;
- : bool list = [false; true; false]
# let prod l1 l2 =
    let sprod n l = List.map (fun i -> n,i) l in
    List.concat (List.map (fun i -> sprod i l2) l1);;
val prod : 'a list -> 'b list -> ('a * 'b) list = <fun>
# let p = prod [0; 1; 2; 3] [0; 1; 2; 3] in
  List.map perfecto (List.map (fun (i,j) -> Node (0, an_act i, an_act j)) p);;
- : bool list =
[true; false; false; false; false; true; false; false; false; false; false;
 false; false; false; false; true]
# let p = prod [0; 1; 2; 3; 4] [0; 1; 2; 3] in
  List.map casi_completo (List.map (fun (i,j) -> Node (0, an_act i, an_act j)) p);;
- : bool list =
[true; false; false; false; true; true; false; false; false; true; false;
 false; false; true; true; true; false; false; false; true]
```