```
(* Applichiamo Programmazione Strutturata allo Sviluppo del QuickSort.
   Vincoli:
   + Una sequenza c;
   + di valori ordinabili t (generico)
   + Quindi, c e' un valore di tipo Seq(t), sequenze di tipo t.
   + Ordinamento totale op:
     perogni x,y: (1) (op x y)or(op y x);  (2) (op x y)and(op y x) iff x=y;
   Algoritmo
   1) Sia cc la sequenza corrente da ordinare;
   2) Sia op ordinamento totale sui valori di tipo t da utilizzare;
   3) Sia pv di tipo t, un elemento di separazione per cc;
   4) Dividiamo cc in TRE sottosequenze cLT, cEQ, cGT separate da a;
   5) Ripetere 1)-5) su cLT e cGT, se non singoletta, ottenute da 4);
   6) Terminiamo presentando la sequenza ottenuta
   Osservazioni:
   o1: pv elemento di cc, arbitrariamente scelto;
   o2: separazione:
            + se x in cLT allora (op x a)
            + se x in cEQ allora (x=a)
            + se x in cGT allora (op a x)
            + cLT seguito da cEQ, seguito da cGT, e' una permutazione di cc
   o3: op e` un valore funzione (predicato) trasmesso come parametro.
*)

# use "SeqT1.ml";;  (* carica il codice del tipo di dato:'t seq *)
                    (* per sequenze polimorfe in 't             *)

let rec quickSort (cc:'t seq) (op:'t ->'t ->bool) =
        (* Out: copia ordinata con op degli elementi di cc *)
        (* effect: nessuna modifica                        *)
    if size cc <= 1 then cc
    else let pv = pivot cc in
         let (cLT,cEQ,cGT) = triPartition cc op pv in
         concatena (quickSort cLT op)(concatena cEQ (quickSort cGT op))
;;


(*
# #use "QuickSortT.ml";;
type 't seq = 't list
val size : 't seq -> int = <fun>
val pivot : 't seq -> 't = <fun>
val triPartition :
  't seq -> ('t -> 't -> bool) -> 't -> 't seq * 't seq * 't seq = <fun>
val concatena : 't seq -> 't seq -> 't seq = <fun>
val cons : 't -> 't seq -> 't seq = <fun>
val quickSort : 'a seq -> ('a -> 'a -> bool) -> 'a seq = <fun>
# quickSort [2;5;0;-3;0;21;0;2;0-2;0;-3] (<);;
- : int seq = [-3; -3; -2; 0; 0; 0; 0; 2; 2; 5; 21]
*)
```