TÖL212M Rökstudd Forritun - Hópverkefni 2

Andri Fannar Kristjánsson

24. janúar 2025

Hópverkefni 2

Leysið, í Dafny, afbrigði af helmingunarleit. Þið munuð þurfa viðeigandi `requires` klausur og `ensures` klausur til að skilgreina virkni fallanna. Þið munuð einnig þurfa viðeigandi `invariant` klausur til að skilgreina fastayrðingu lykkju. Notið beinagrind af lausninni sem finna má á Canvas í skránni `H2-skeleton.dfy`. Sú beinagrind inniheldur prófunarfall sem sannreynir, að hluta, rökfræðilega eiginleika lausnarinnar. Skilið PDF lausn í Gradescope og sýnið þar einnig permalink á lausnina í tio.run eða sendið lausnina í tölvupósti til Snorra.

Svar:

Hér fyrir neðan má sjá leystu útgáfuna sem Dafny samþykkir. Hægt er einnig að sjá kóðann hér: https://shorturl.at/510wF.

```
// Author of question: Snorri Agnarsson
// Permalink of question: https://tinyurl.com/3xz4kd2p

// Authors of solution:   Andri Fannar Kristjánsson
// Permalink of solution: https://shorturl.at/510wF
// Use the command
//   dafny build H2-skeleton.dfy
// to compile the file.
// Or use the web page tio.run/#dafny.

// When you have solved the problem put
// the solution on the Dafny web page,
// generate a permalink and put it in
// this file or email the solution to me.

method SearchRecursive( a: seq<real>, i: int, j: int, x: real )
    returns ( k: int )
  decreases j-i
  requires 0 <= i <= j <= |a|
  requires forall p,q | i <= p < q < j :: a[p] >= a[q]
  ensures i <= k <= j
  ensures forall r | i <= r < k :: a[r] >= x
  ensures forall r | k <= r < j :: a[r] < x

  // a: |   | >= x | ??? | < x   |   |
  //      ^             ^      ^           ^
  //      0             i      j          |a|
{
  // Search over.
  if i == j
  {
    return i;
  }
```

```
    // Find middle.
    var m := i+(j-i)/2;

    // If element at middle is greater or equal to x
    // then search in the right half, excluding m.
    if a[m] >= x
    {
      k := SearchRecursive(a, m+1, j, x);
    }
    else
    {
      // Else search left half, including m
      // (as it will be excluded).
      k := SearchRecursive(a, i, m, x);
    }

    return k;
  }

  method SearchLoop( a: seq<real>, i: int, j: int, x: real )
      returns ( k: int )
    requires 0 <= i <= j <= |a|
    requires forall p,q | i <= p < q < j :: a[p] >= a[q]
    ensures i <= k <= j
    ensures forall r | i <= r < k :: a[r] >= x
    ensures forall r | k <= r < j :: a[r] < x
  {
    // Initialize p and q.
    var p, q := i, j;

    while p < q
      decreases q-p
      invariant i <= p <= q <= j
      invariant forall r | i <= r < p :: a[r] >= x
      invariant forall r | q <= r < j :: a[r] < x
      // a: |   | >= x | ??? | < x   |   |
      //       ^   ^        ^       ^        ^   ^
      //       0   i        p       q        j  |a|
    {
      // Find middle.
      var m := p+(q-p)/2;

      // If element at middle is greater or equal to x
      // then increase p to m+1 to search in the right half.
      if a[m] >= x
      {
        p := m+1;
      }
      else
      {
        // Else decrease q to m to search in the left half.
        q := m;
      }
    }
    return p;
  }
```

```
// Ef eftirfarandi fall er ekki samþykkt þá eru
// föllin ekki að haga sér rétt að mati Dafny.

// If the following method is not accepted then
// the functions are not behaving correctly in
// Dafny's opinion.

method Test( a: seq<real>, x: real )
  requires forall p,q | 0 <= p < q < |a| :: a[p] >= a[q]
{
  var k1 := SearchLoop(a,0,|a|,x);
  assert forall r | 0 <= r < k1 :: a[r] >= x;
  assert forall r | k1 <= r < |a| :: a[r] < x;
  var k2 := SearchRecursive(a,0,|a|,x);
  assert forall r | 0 <= r < k2 :: a[r] >= x;
  assert forall r | k2 <= r < |a| :: a[r] < x;
}
```