



GREINING REIKNIRITA

TÖL403G

Skilaverkefni 2

Verkefnishöfundar:

Guðmundur MÁR GUNNARSSON

Skarphéðinn ÞÓRÐARSSON

Sigurður SKÚLI SIGURGEIRSSON

Kennari:

Páll MELSTED

9. mars 2014

1 Keyrsla og virkni

Öll forritun fór fram í Java og notast við java version 1.6.0_27 fyrir Windows við keyrslu og þýðingu þess. Til þess að þýða forritið þarf einungis að sækja skjalið "IntervalTree.java" og þýða það með "javac IntervalTree.java". Forritið er aðgengilegt á <https://github.com/gudmundurmar/SKSKG>. Þar er einnig að finna Stopwatch.java sem notast var við til þess að taka tímann á forritinu en einnig uppgæfin inntök inntök og úttök s1-s3 sem notast var við til þess að keyra forritið líkt og gefið var upp í verkefnislýsingunni.

Þýðing: `$javac IntervalTree.java`

Keyrsla: `$java IntervalTree < s1.in | diff -w s1.out - | wc -l`

2 Uppbygging

Tréð er byggt upp þannig að fyrsta bilið verður rótin. Ef næsta inntak skarast ekki við bilið í nóðunni og neðri mörk bilsins eru stærri en efri mörk rötarinnar fer það hægra megin, annars fer það vinstra megin í tréð. Ef bilið skarast aftur á móti við rótina fer það í tengdann lista hjá rótinni. Það sama á við um rótina og aðrar nóður í trénu. Ef bilið skarast bætist það við nóðuna í tengdum lista. Þessi lausn hefur þann vankant að ef við byrjum á því að bæta stórum bilum í tréð getum við lent í því að tréð okkar verði bara einn tengdur listi. Upprunalega vildum við notast við median bilsins til þess að raða upp í tréð en þar sem að við fengum það ekki til þess að virka rétt, vildum við frekar hafa rétta lausn sem tæki mögulega aðeins lengri tíma frekar en ranga lausn sem tæki minni tíma. Við hugsuðum líka um kosti og galla þess að notast við rauð-svört tré til þess að halda leitartíma í lágmarki en af því að við erum að við vitum ekki hvort við erum að leita oftar en við bætum í tréð og eyðum úr því, gátum við ekki sagt til um hvort við myndum græða meira á því eða hreinlega tapa á því að innleiða rautt-svart tré í lausnina okkar.

2.1 Yfirklassinn - IntervalTree

```
public class IntervalTree {  
  
    static ArrayList<int[]> closedIntervalList = new ArrayList<int[]>();  
    Node root; //rót trésins  
  
    //FG: Lokuðu bilin eru geymd í tré sem hefur rót í root.  
    //Öll bil sem prenta skal út eru geymd í closedIntervalList  
  
    public IntervalTree() { root = null; }  
  
    //Notkun: tree.insert(a,b);  
    //Fyrir: a og b eru heiltölur, a < b  
    //Eftir: búið er að bæta bilinu [a,b] í tréð  
    public void insert(int a, int b){}  
  
    //Notkun: tree.delete(a,b);  
    //Fyrir: a og b eru heiltölur, a <= b  
    //Eftir: Ef [a,b] var í trénu þá er búið að eyða því  
    public void delete(int a, int b){}  
  
    //Notkun: tree.intersects(a,b,root);  
    //Fyrir: a og b eru heiltölur, a <= b, root er nóða  
    //Eftir: búið er að finna öll bil sem skarast á við [a,b] í trénu með rótina root  
    public int intersects(int a, int b, Node node){}
```

```

//Notkun: tree.intersects(a,b);
//Fyrir: a og b eru heiltölur, a <= b
//Eftir: búið er að finna og prenta út öll bil sem skarast á við bilið [a,b]
public void intersects(int a, int b) {}

//Notkun: tree.contains(a,b,root);
//Fyrir: a og b eru heiltölur, a <= b, root er nóða
//Eftir: búið er að finna öll bil sem innihalda[a,b] í trénu með rótina root
public boolean contains(int a, int b, Node node){}

//Notkun: tree.contains(a,b);
//Fyrir: a og b eru heiltölur, a <= b, root er nóða
//Eftir: búið er að finna og prenta öll bil sem innihalda[a,b]
public void contains(int a, int b){}

//Notkun: tree.point(a);
//Fyrir: a er heiltala
//Eftir: búið er að finna og prenta öll bil sem innihalda a
public void point(int a){}

//Notkun: printFoundIntervals()
//Fyrir: Ekkert
//Eftir: Búið er að prenta öll þau bil sem leitað var að
public void printFoundIntervals(){}

//Notkun: tree.deleteNode(node)
//Fyrir: node er nóða
//Eftir: Búið er að fjarlægja node úr trénu
public void deleteNode(Node node){}

//Notkun: tree.query(q);
//Fyrir: q er strengur sem inniheldur fyrirspurn fyrir Interval tréð
//Eftir: Búið er að framkvæma fyrirspurnina í q.
public void query(String query) {}

//nóðurnar í trénu
static class Node
{ /*Sjá næstu földunarhæð hér að neðan*/ }

public static void main(String[] args)
{
    //Býr til nýtt tré sem það síðan
    //byggir upp með inntaki sem lesið er inn af standard input
}
}

```

2.2 Fyrsta földun - Node

```

//nóðurnar í trénu
static class Node
{
    int lower; //neðri mörk
    int higher; //efri mörk
}

```

```

Node left; //vinstra barn
Node right; //hægra barn
Node parent; //foreldri
Link intervals; //bilin sem skarast á við bilið í nódunni

//FG: Lokuðu bilin í nódunni eru geymd í tengda listanum intervals, hægra barn
//nóðurnar er í right og vinstra barn í left. Foreldri nóðunnar er parent.
//lower er lægri endi fyrsta lokaða bilsins sem var sett í nóduna og higher
//er hærri endi lokaðs bilsins.

static class Link
{
    //Sjá næstu földun í skjalinu
}

//Notkun: node.insertInterval(a,b);
//Fyrir: a og b eru heiltölur, a < b
//Eftir: búið er að setja bilið [a,b] á réttan stað í intervals
void insertInterval(int a, int b){}

//Notkun: node.findIntersections(a,b);
//Fyrir: a og b eru heiltölur, a < b
//Eftir: búið er að finna öll bil sem skarast á við bilið [a,b] í nódunni node
int findIntersections(int a,int b){}

//Notkun: node.findContains(a,b);
//Fyrir: a og b eru heiltölur, a < b
//Eftir: búið er að finna öll bil sem innihalda [a,b] í nódunni node
boolean findContains(int a,int b){}

//Notkun: node.deleteInterval(a,b);
//Fyrir: a og b eru heiltölur, a < b
//Eftir: Búið er að fjarlægja lokaða bilið [a,b] ef það var geymt í nódunni.
void deleteInterval(int a, int b){}
}

```

2.3 Önnur földun - Link

```

static class Link
{
    Link next;
    int lower;
    int higher;

    //FG: next bendir á næsta hlekk í tengda listanum, lower er lægri endi lokaðs bils
    //og higher er hærri endi lokaðs bils.

    //Notkun: link.compareTo(a,b);
    //Fyrir: a og b eru heiltölur, a < b
    //Eftir: Skilar 1 ef [lower,higher] < [a,b], 0 ef þau eru jöfn og -1 annars
    int compareTo(int a, int b){}
}

```

3 Forritið í heild

```
1 import java.util.Scanner;
2 import java.util.ArrayList;
3 import java.util.Collections;
4 import java.util.Comparator;
5 import java.util.Arrays;
6
7 public class IntervalTree {
8
9     static ArrayList<int[]> closedIntervallList = new ArrayList<int[]>();
10    Node root; //rót trésins
11
12    //FG: Lokuðu bilin eru geymd í tré sem hefur rót í root.
13    //Öll bil sem prenta skal út eru geymd í closedIntervallList
14
15    public IntervalTree()
16    {
17        root = null;
18    }
19
20    //Notkun: tree.insert(a,b);
21    //Fyrir: a og b eru heiltölur, a < b
22    //Eftir: búið er að bæta bilinu [a,b] í tréð
23    public void insert(int a, int b)
24    {
25        if(b < a) return;
26
27        Node newNode = new Node();
28        newNode.lower = a;
29        newNode.higher = b;
30        newNode.insertInterval(a,b);
31
32        if(root == null) {
33            root = newNode;
34            return;
35        }
36
37        Node tree = root;
38
39        while(tree != null)
40        {
41            if(b < tree.lower)
42            {
43                if(tree.left != null)
44                {
45                    tree = tree.left;
46                }
47            }
48            else
49            {
50                newNode.parent = tree;
51                tree.left = newNode;
```

```

51         return;
52     }
53 }
54 else if(a > tree.higher)
55 {
56     if(tree.right != null)
57     {
58         tree = tree.right;
59     }
60     else
61     {
62         newNode.parent = tree;
63         tree.right = newNode;
64         return;
65     }
66 }
67 else
68 {
69     tree.insertInterval(a,b);
70     return;
71 }
72 }
73 }
74
75 //Notkun: tree.delete(a,b);
76 //Fyrir: a og b eru heiltölur, a <= b
77 //Eftir: Ef [a,b] var í trénu þá er búið að eyða því
78 public void delete(int a, int b)
79 {
80     if(root == null || b < a) return;
81
82     Node tree = root;
83
84     while(tree != null)
85     {
86         if(b < tree.lower)
87         {
88             if(tree.left != null)
89             {
90                 tree = tree.left;
91             }
92             else
93             {
94                 return;
95             }
96         }
97         else if(a > tree.higher)
98         {
99             if(tree.right != null)
100             {
101                 tree = tree.right;
102             }
103             else

```

```

104         {
105             return;
106         }
107     }
108     else
109     {
110         tree.deleteInterval(a,b);
111         if(tree.intervals == null)
112         {
113             deleteNode(tree);
114         }
115         return;
116     }
117 }
118 }
119
120 //Notkun: tree.intersects(a,b,root);
121 //Fyrir: a og b eru heiltölur, a < b, root er nóða
122 //Eftir: búið er að finna öll bil sem skarast á við [a,b] í trénu með rótina root
123 public int intersects(int a, int b, Node node)
124 {
125     if(b < a) return 0;
126
127
128     if(node == null)
129     {
130         return 0;
131     }
132
133     int instanceFound = 0;
134
135     Node tree = node;
136
137     if(a < tree.lower)
138     {
139         instanceFound = instanceFound + intersects(a,b, tree.left);
140     }
141
142     instanceFound = instanceFound + node.findIntersections(a,b);
143
144     if(b > tree.higher)
145     {
146         instanceFound = instanceFound + intersects(a,b, tree.right);
147     }
148     return instanceFound;
149 }
150
151 //Notkun: tree.intersects(a,b,root);
152 //Fyrir: a og b eru heiltölur, a < b, root er nóða
153 //Eftir: búið er að finna og prenta öll bil sem skarast á við [a,b]
154 public void intersects(int a, int b)
155 {
156     int instance = intersects(a, b, root);

```

```

157     if(instance == 0)
158     {
159         System.out.print("[");
160     }else{
161         printFoundIntervals();
162     }
163     System.out.println("");
164 }
165
166 //Notkun: tree.contains(a,b,root);
167 //Fyrir: a og b eru heiltölur, a <= b, root er nóða
168 //Eftir: búið er að finna öll bil sem innihalda[a,b] í trénu með rótina root
169 public boolean contains(int a, int b, Node node)
170 {
171     if(b < a) return false;
172
173     if(node == null)
174     {
175         return false;
176     }
177
178     boolean instanceFound = false;
179
180     Node tree = node;
181
182     if(a < tree.lower)
183     {
184         boolean left = contains(a,b, tree.left);
185         instanceFound = instanceFound || left;
186     }
187
188     boolean center = tree.findContains(a,b);
189     instanceFound = instanceFound || center;
190
191     if(b > tree.higher)
192     {
193         boolean right = contains(a,b, tree.right);
194         instanceFound = instanceFound || right;
195     }
196
197     return instanceFound;
198 }
199
200 //Notkun: tree.contains(a,b);
201 //Fyrir: a og b eru heiltölur, a <= b, root er nóða
202 //Eftir: búið er að finna og prenta öll bil sem innihalda[a,b]
203 public void contains(int a, int b)
204 {
205     boolean instance = contains(a, b, root);
206     if(!instance)
207     {
208         System.out.print("[");
209     }else{

```



```

210     printFoundIntervals();
211 }
212 System.out.println("");
213 }
214
215 //Notkun: tree.point(a);
216 //Fyrir: a er heiltala
217 //Eftir: búið er að finna og prenta öll bil sem innihalda a
218 public void point(int a)
219 {
220     boolean instance = contains(a, a, root);
221     if(!instance)
222     {
223         System.out.print("[");
224     }else{
225         printFoundIntervals();
226     }
227     System.out.println("");
228 }
229
230 //Notkun: printFoundIntervals()
231 //Fyrir: Ekkert
232 //Eftir: Búið er að prenta öll þau bil sem leitað var að
233 public void printFoundIntervals(){
234
235     // Comparator sem sér um að röðun
236     Comparator<int[]> sort = new Comparator<int[]>() {
237         public int compare(int[] a, int[] b) {
238             if(a[0] < b[0]) return -1;
239             else if(a[0] > b[0]) return 1;
240             else return 0;
241         }
242     };
243
244     Collections.sort(closedIntervallList, sort);
245
246     for (int[] arr : closedIntervallList) {
247         System.out.print(Arrays.toString(arr)+" ");
248     }
249     closedIntervallList.clear();
250 }
251
252 //Notkun: tree.deleteNode(node)
253 //Fyrir: node er nóða
254 //Eftir: Búið er að fjarlægja node úr trénu
255 public void deleteNode(Node node)
256 {
257
258     if(node == null) return;
259
260     if(node.left == null && node.right == null)
261     {
262         node = null;

```

```

263     return;
264 }
265 if(node.right == null)
266 {
267     node.left.parent = node.parent;
268     node = node.left;
269     return;
270 }
271
272 Node search = node.right;
273 while(search.left != null)
274 {
275     search = search.left;
276 }
277
278 Node copyOfSearch = search;
279
280 search = search.right;
281 copyOfSearch.parent = node.parent;
282 node = copyOfSearch;
283 }
284
285 //Notkun: tree.query(q);
286 //Fyrir: q er strengur sem inniheldur fyrirspurn í Interval tréð
287 //Eftir: Búið er að framkvæma fyrirspurnina í q.
288 public void query(String query) {
289     String[] splitQuery = query.split(" ");
290     int lower = Integer.parseInt(splitQuery[1]);
291
292     if(splitQuery[0].equals("?p"))
293     {
294         point(lower);
295     }
296     else
297     {
298
299         int higher = Integer.parseInt(splitQuery[2]);
300
301         if(splitQuery[0].contains("+"))
302         {
303             insert(lower, higher);
304         }
305         else if(splitQuery[0].equals("-"))
306         {
307             delete(lower, higher);
308         }
309         else if(splitQuery[0].equals("?o"))
310         {
311             intersects(lower,higher);
312         }
313         else if(splitQuery[0].equals("?i"))
314         {
315             contains(lower,higher);

```

```

316     }
317 }
318
319 }
320
321 //nóðurnar í trénu
322 static class Node
323 {
324     int lower; //neðri mörk
325     int higher; //efri mörk
326     Node left; //vinstra barn
327     Node right; //hægra barn
328     Node parent; //foreldri
329     Link intervals; //bilin sem skarast á við bilið í nóðunni
330
331     //FG: Lokuðu bilin í nóðunni eru geymd í tengda listanum intervals, hægra barn
332     //nóðurnar er í right og vinstra barn í left. Foreldri nóðunnar er parent.
333     //lower er lægri endi fyrsta lokaðs bilsins sem var sett í nóðuna og higher
334     //er hærri endi lokaðs bilsins.
335
336     static class Link
337     {
338         Link next;
339         int lower;
340         int higher;
341
342         //FG: next bendir á næsta hlekk í tengda listanum, lower er lægri endi lokaðs bils
343         //og higher er hærri endi lokaðs bils.
344
345         //Notkun: link.compareTo(a,b);
346         //Fyrir: a og b eru heiltölur, a < b
347         //Eftir: Skilar 1 ef [lower,higher] < [a,b], 0 ef þau eru jöfn og -1 annars
348         int compareTo(int a, int b)
349         {
350             if(lower < a)
351             {
352                 return 1;
353             }
354             else if(lower > a)
355             {
356                 return -1;
357             }
358             else
359             {
360                 if(higher < b)
361                 {
362                     return 1;
363                 }
364                 else if(higher > b)
365                 {
366                     return -1;
367                 }
368                 else

```

```

369         {
370             return 0;
371         }
372     }
373 }
374 }
375
376 //Notkun: node.insertInterval(a,b);
377 //Fyrir: a og b eru heiltölur, a < b
378 //Eftir: búið er að setja bilið [a,b] á réttan stað í intervals
379 void insertInterval(int a, int b)
380 {
381     Link newLink = new Link();
382     newLink.lower = a;
383     newLink.higher = b;
384     if( intervals == null || intervals.compareTo(a,b)<0 )
385     {
386         newLink.next=intervals;
387         intervals=newLink;
388         return;
389     }
390     if(intervals.lower == a && intervals.higher == b)
391     {
392         return;
393     }
394     Link temp = intervals;
395     while( temp.next != null )
396     {
397         if( temp.next.compareTo(a,b) > 0 )
398         {
399             temp = temp.next;
400         }
401         else if( temp.next.compareTo(a,b) == 0 )
402             return;
403         else
404         {
405             newLink.next = temp.next;
406             temp.next = newLink;
407             return;
408         }
409     }
410     newLink.next = temp.next;
411     temp.next = newLink;
412 }
413
414 //Notkun: node.findIntersections(a,b);
415 //Fyrir: a og b eru heiltölur, a < b
416 //Eftir: búið er að finna öll bil sem skarast á við bilið [a,b] í nódunni node
417 int findIntersections(int a,int b)
418 {
419     Link chain = intervals;
420
421     int found = 0;

```

```

422 while(chain != null)
423 {
424     if(chain.lower > b) break;
425
426     if((chain.lower <= b && chain.higher >= b) || (chain.lower <= a && chain.higher >= a))
427     {
428         int[] closedInterval = {chain.lower, chain.higher};
429         closedIntervalList.add(closedInterval);
430         found++;
431     }
432     else if((chain.lower >= a && chain.lower <= b) || (chain.higher >= a && chain.higher <= b))
433     {
434         int[] closedInterval = {chain.lower, chain.higher};
435         closedIntervalList.add(closedInterval);
436         found++;
437     }
438
439     chain = chain.next;
440 }
441
442 return found;
443 }
444
445 //Notkun: node.findContains(a,b);
446 //Fyrir: a og b eru heiltölur, a < b
447 //Eftir: búið er að finna öll bil sem innihalda [a,b] í nóðunni node
448 boolean findContains(int a,int b)
449 {
450     Link chain = intervals;
451
452     boolean found = false;
453
454     while(chain != null)
455     {
456         if(chain.lower > a) break;
457
458         if(chain.lower <= a && b <= chain.higher)
459         {
460             int[] closedInterval = {chain.lower, chain.higher};
461             closedIntervalList.add(closedInterval);
462             found = true;
463         }
464         chain = chain.next;
465     }
466
467     return found;
468 }
469
470 //Notkun: node.deleteInterval(a,b);
471 //Fyrir: a og b eru heiltölur, a < b
472 //Eftir: Búið er að fjarlægja lokaða bilið [a,b] ef það var geymt í nóðunni.
473 void deleteInterval(int a, int b)

```

```

475     {
476         if(intervals == null) return;
477
478         //athugar hvort fremsta stakið sé það sem verið er að leita af
479         if(intervals.lower == a && intervals.higher == b)
480         {
481             intervals = intervals.next;
482             return;
483         }
484
485         Link chain = intervals;
486
487         //fer í gegnum afganginn af listanum og leitar
488         while(chain.next != null)
489         {
490             if(chain.next.lower == a && chain.next.higher == b)
491             {
492                 chain.next = chain.next.next;
493                 return;
494             }
495
496             chain = chain.next;
497         }
498     }
499
500 }
501
502 public static void main(String[] args)
503 {
504     IntervalTree tree = new IntervalTree();
505     Scanner scanner = new Scanner(System.in);
506     while(scanner.hasNext())
507     {
508         String query = scanner.nextLine();
509         tree.query(query);
510     }
511 }
512 }

```
