



GREINING REIKNIRITA

TÖL403G

Skilaverkefni 2

Verkefnishöfundar:

Guðmundur MÁR GUNNARSSON

Skarphéðinn ÞÓRÐARSSON

Sigurður SKÚLI SIGURGEIRSSON

Kennari:

Páll MELSTED

7. mars 2014

```

1 port java.util.Scanner;
2
3 public class IntervalTree {
4
5     //nóðurnar í trénu
6     static class Node
7
8     int q; //miðgildið í bilinu
9     int lower; //neðri mörk
10    int higher; //efri mörk
11    Node left; //vinstra barn
12    Node right; //hægra barn
13    Node parent; //foreldri
14    Link intervals; //bilin sem skerast á við bilið í nóðunni
15
16    //Notkun: node.insertInterval(a,b);
17    //Fyrir: a og b eru heiltölur, a < b
18    //Eftir: búið er að setja bilið [a,b] á réttan stað í intervals
19    void insertInterval(int a, int b)
20    {
21        Link newLink = new Link();
22        newLink.lower = a;
23        newLink.higher = b;
24        if( intervals == null || intervals.compareTo(a,b)<0 )
25        {
26            newLink.next=intervals;
27            intervals=newLink;
28            return;
29        }
30        if(intervals.lower == a && intervals.higher == b)
31        {
32            return;
33        }
34        Link temp = intervals;
35        while( temp.next != null )
36        {
37            if( temp.next.compareTo(a,b) > 0 )
38            {
39                temp = temp.next;
40            }
41            else if( temp.next.compareTo(a,b) == 0 )
42                return;
43            else
44            {
45                newLink.next = temp.next;
46                temp.next = newLink;
47                return;
48            }
49        }
50        newLink.next = temp.next;
51        temp.next = newLink;
52    }

```

```

53
54 //Notkun: node.findIntersections(a,b);
55 //Fyrir: a og b eru heiltölur, a < b
56 //Eftir: búið er að finna öll bil sem skerast á við bilið [a,b]
57 int findIntersections(int a,int b)
58 {
59     Link chain = intervals;
60
61     int found = 0;
62
63     while(chain != null)
64     {
65         if((chain.lower <= b && chain.higher >= b) || (chain.lower <= a && chain.higher >= a))
66         {
67             System.out.print("[ "+chain.lower+", "+chain.higher+" ]");
68             found++;
69         }
70         else if((chain.lower >= a && chain.lower <= b) || (chain.higher >= a && chain.higher <=
71         {
72             System.out.print("[ "+chain.lower+", "+chain.higher+" ]");
73             found++;
74         }
75         chain = chain.next;
76     }
77
78     return found;
79 }
80
81 //Notkun: node.findContains(a,b);
82 //Fyrir: a og b eru heiltölur, a < b
83 //Eftir: búið er að finna öll bil sem innihalda [a,b]
84 boolean findContains(int a,int b)
85 {
86     Link chain = intervals;
87
88     boolean found = false;
89
90     while(chain != null)
91     {
92         if(chain.lower <= a && b <= chain.higher)
93         {
94             System.out.print("[ "+chain.lower+", "+chain.higher+" ]");
95             found = true;
96         }
97         chain = chain.next;
98     }
99
100     return found;
101 }
102
103 void deleteInterval(int a, int b)
104 {
105     if(intervals == null) return;

```

```

106
107 //athugar hvort fremsta stakið sé það sem verið er að leita af
108 if(intervals.lower == a && intervals.higher == b)
109 {
110     intervals = intervals.next;
111     return;
112 }
113
114 Link chain = intervals;
115
116 //fer í gegnum afganginn af listanum og leitar
117 while(chain.next != null)
118 {
119     if(chain.next.lower == a && chain.next.higher == b)
120     {
121         chain.next = chain.next.next;
122         return;
123     }
124
125     chain = chain.next;
126 }
127 }
128
129
130
131 tatic class Link {
132     Link next;
133     int lower;
134     int higher;
135
136     //Notkun: link.compareTo(a,b);
137     //Fyrir: a og b eru heiltölur, a < b
138     //Eftir: Skilar 1 ef [lower,higher] < [a,b], 0 ef þau eru jöfn og -1 annars
139     int compareTo(int a, int b)
140     {
141         if(lower < a)
142         {
143             return 1;
144         }
145         else if(lower > a)
146         {
147             return -1;
148         }
149         else
150         {
151             if(higher < b)
152             {
153                 return 1;
154             }
155             else if(higher > b)
156             {
157                 return -1;
158             }

```

```

159         else
160         {
161             return 0;
162         }
163     }
164 }
165
166
167 ode root;
168
169 public IntervalTree()
170
171 root = null;
172
173
174 /Notkun: tree.insert(a,b);
175 /Fyrir: a og b eru heiltölur, a < b
176 /Eftir: búið er að bæta bilinu [a,b] í tréð
177 public void insert(int a, int b)
178
179 if(b < a) return;
180
181 Node newNode = new Node();
182 newNode.q = a+b/2;
183 newNode.lower = a;
184 newNode.higher = b;
185 newNode.insertInterval(a,b);
186
187 if(root == null) {
188     root = newNode;
189     return;
190 }
191
192 Node tree = root;
193
194 while(tree != null)
195 {
196     if(b < tree.lower)
197     {
198         if(tree.left != null)
199         {
200             tree = tree.left;
201         }
202         else
203         {
204             newNode.parent = tree;
205             tree.left = newNode;
206             return;
207         }
208     }
209     else if(a > tree.higher)
210     {
211         if(tree.right != null)

```

```

212         {
213             tree = tree.right;
214         }
215         else
216         {
217             newNode.parent = tree;
218             tree.right = newNode;
219             return;
220         }
221     }
222     else
223     {
224         tree.insertInterval(a,b);
225         return;
226     }
227 }
228
229
230 /Notkun: tree.intersects(a,b,root);
231 /Fyrir: a og b eru heiltölur, a < b, root er nóða
232 /Eftir: búið er að finna öll bil sem skerast á við [a,b]
233 public int intersects(int a, int b, Node node)
234
235 if(b < a) return 0;
236
237
238 if(node == null)
239 {
240     return 0;
241 }
242
243 int instanceFound = 0;
244
245 Node tree = node;
246
247 if(a < tree.lower)
248 {
249     instanceFound = instanceFound + intersects(a,b, tree.left);
250 }
251
252 instanceFound = instanceFound + node.findIntersections(a,b);
253
254 if(b > tree.higher)
255 {
256     instanceFound = instanceFound + intersects(a,b, tree.right);
257 }
258 return instanceFound;
259
260
261 public void intersects(int a, int b)
262
263 int instance = intersects(a, b, root);
264 if(instance == 0)

```

```

265 {
266     System.out.print("[]");
267 }
268 System.out.println("");
269
270
271 /Notkun: tree.contains(a,b,root);
272 /Fyrir: a og b eru heiltölur, a <= b, root er nóða
273 /Eftir: bóðið er að finna öll bil sem innihalda[a,b]
274 public boolean contains(int a, int b, Node node)
275
276 if(b < a) return false;
277
278 if(node == null)
279 {
280     return false;
281 }
282
283 boolean instanceFound = false;
284
285 Node tree = node;
286
287 if(a < tree.lower)
288 {
289     boolean left = contains(a,b, tree.left);
290     instanceFound = instanceFound || left;
291 }
292
293 boolean center = tree.findContains(a,b);
294 instanceFound = instanceFound || center;
295
296 if(b > tree.higher)
297 {
298     boolean right = contains(a,b, tree.right);
299     instanceFound = instanceFound || right;
300 }
301
302 return instanceFound;
303
304
305 public void contains(int a, int b)
306
307 boolean instance = contains(a, b, root);
308 if(!instance)
309 {
310     System.out.print("[]");
311 }
312 System.out.println("");
313
314
315 /Notkun: tree.point(a);
316 /Fyrir: a er heiltala
317 /Eftir: bóðið er að finna öll bil sem innihalda a

```

```

318 public void point(int a)
319
320 boolean instance = contains(a, a, root);
321 if(!instance)
322 {
323     System.out.print("[]");
324 }
325 System.out.println("");
326
327
328 /Notkun: tree.delete(a,b);
329 /Fyrir: a og b eru heiltölur, a <= b
330 /Eftir: Ef [a,b] var í trénu þá er búið að eyða því
331 public void delete(int a, int b)
332
333
334 if(root == null || b < a) return;
335
336 Node tree = root;
337
338 while(tree != null)
339 {
340     if(b < tree.lower)
341     {
342         if(tree.left != null)
343         {
344             tree = tree.left;
345         }
346         else
347         {
348             return;
349         }
350     }
351     else if(a > tree.higher)
352     {
353         if(tree.right != null)
354         {
355             tree = tree.right;
356         }
357         else
358         {
359             return;
360         }
361     }
362     else
363     {
364         tree.deleteInterval(a,b);
365         if(tree.intervals == null)
366         {
367             deleteNode(tree);
368         }
369         return;
370     }

```



```

371 }
372
373
374 /Notkun: tree.deleteNode(node)
375 /Fyrir: node er nóða
376 /Eftir: Búið er að fjarlægja node úr trénu
377 public void deleteNode(Node node)
378
379
380
381 if(node == null) return;
382
383 if(node.left == null && node.right == null)
384 {
385     node = null;
386     return;
387 }
388 if(node.right == null)
389 {
390     node.left.parent = node.parent;
391     node = node.left;
392     return;
393 }
394
395 Node search = node.right;
396 while(search.left != null)
397 {
398     search = search.left;
399 }
400
401 Node copyOfSearch = search;
402
403 search = search.right;
404 copyOfSearch.parent = node.parent;
405 node = copyOfSearch;
406
407
408
409
410 /gengur í gegnum tréð, bara aðstodarfall ekki skila
411 public void traverse(Node root)
412
413 if(root == null)
414 {
415     return;
416 }
417 System.out.println(root.lower+" : "+root.higher);
418 traverse(root.left);
419 traverse(root.right);
420
421
422 public static void main(String[] args)
423

```

```

424 IntervalTree tree = new IntervalTree();
425 Scanner scanner = new Scanner(System.in);
426 while(scanner.hasNext())
427 {
428     String query = scanner.nextLine();
429     String[] splitQuery = query.split(" ");
430     int lower = Integer.parseInt(splitQuery[1]);
431
432     if(splitQuery[0].equals("?p"))
433     {
434         tree.point(lower);
435     }
436     else {
437
438         int higher = Integer.parseInt(splitQuery[2]);
439
440         if(splitQuery[0].contains("+"))
441         {
442             tree.insert(lower, higher);
443         }
444         else if(splitQuery[0].equals("?o"))
445         {
446             tree.intersects(lower,higher);
447         }
448         else if(splitQuery[0].equals("?i"))
449         {
450             tree.contains(lower,higher);
451         }
452         else if(splitQuery[0].equals("-"))
453         {
454             tree.delete(lower, higher);
455         }
456     }
457 }
458
459
460 }
461
462
463
464
465
466

```
