Tölvunarfræði II
Skiladæmi 12

Sesar Hersisson - seh32

Dæmi 1:

```
public class Degrees
{
  private Digraph G;

  public Degrees(Digraph G)
  {
    this.G = new Digraph(G);
  }

  public int indegree(int v)
  {
    return G.indegree(v);
  }

  public int outdegree(int v)
  {
    return G.outdegree(v);
  }

  public Iterable<Integer> sources()
  {
    Bag<Integer> s = new Bag<Integer>();
    for(int i = 0; i < G.V(); i++)
    {
      if(indegree(i) == 0) s.add(i);
    }
    return s;
  }

  public Iterable<Integer> sinks()
  {
    Bag<Integer> s = new Bag<Integer>();
    for(int i = 0; i < G.V(); i++)
    {
      if(outdegree(i) == 0) s.add(i);
    }
    return s;
  }

  public boolean isMap()
  {
    for(int i = 0; i < G.V(); i++)
    {
      if(outdegree(i) != 1) return false;
    }
    return true;
  }

  public static void main(String[] args){
    In I = new In();
    Digraph G = new Digraph(I);
    Degrees Dgs = new Degrees(G);
    for(int i : Dgs.sources())
    {
      System.out.println(i);
    }
    for(int i : Dgs.sinks())
    {
      System.out.println(i);
    }
  }
}
```

Dæmi 2:

```java
public class Topological
{
  private Iterable<Integer> order;
  public Topological(Digraph G)
  {
    // topological order
    DirectedCycle cyclefinder = new DirectedCycle(G);
    if (!cyclefinder.hasCycle())
    {
      DepthFirstOrder dfs = new DepthFirstOrder(G);
      order = dfs.reversePost();
    }
   }
   public Iterable<Integer> order()
   {
     return order;
   }

   public boolean isDAG()
   {
     return order == null;
   }

   public static void main(String[] args)
   {
     String filename = args[0];
     String separator = args[1];
     In I = new In();
     SymbolDigraph sg = new SymbolDigraph(filename, separator);
     Topological top = new Topological(sg.G());
     int[] perm = I.readAllInts();
     boolean topolog = true;
     int i = 0;
     for (int v : top.order())
     {
       if(perm[i] != v)
       {
         StdOut.println("Ekki topological");
         topolog = false;
         break;
       }
       i++;
     }
     if(topolog) StdOut.println("Topological");
   }
}
```

Dæmi 3:

```
public class ArithmaticDAG
{
  private SymbolDigraph SG;
  private Digraph G;

  public ArithmaticDAG(SymbolDigraph SG)
  {
    this.SG = SG;
    Digraph G = SG.digraph();
    DirectedCycle dc = new DirectedCycle(G);
    if (dc.hasCycle()) throw new IllegalArgumentException("Digraph
       must be a DAG");
  }

  public double value()
  {
    double[] vals = new double[G.V()];
    for (int i = G.V() - 1; i >= 0; i--)
    {
      String str = SG.nameOf(i);
      if(G.outdegree(i) == 0)
      {
        double val = Double.parseDouble(str);
        vals[i] = val;
      }
      else
      {
        switch(str)
        {
          case "+" :
          {
            vals[i] = 0;
            for(int v : G.adj(i))
            {
              vals[i] += vals[v];
            }
            break;
          }
          case "*" :
          {
            vals[i] = 1;
            for(int v : G.adj(i))
            {
              vals[i] *= vals[v];
            }
            break;
          }
          case "-" :
          {
            double[] temp = new double[2];
            int counter = 0;
            for(int v : G.adj(i))
            {
              temp[counter] = vals[v];
              counter++;
            }
            vals[i] = temp[0] - temp[1];
            break;
          }
          case "/" :
          {
            double[] temp = new double[2];
            int counter = 0;
            for(int v : G.adj(i))
            {
              temp[counter] = vals[v];
              counter++;
            }
            vals[i] = temp[0] / temp[1];
            break;
          }
        }
      }
    }
    return vals[0];
  }
}
```

Dæmi 4:

Ég bætti eftirfarandi tilviksbreytum við í DijkstraSP.java

```
    private double[] SecDistTo;        // SecDistTo[v] = distance  of
       second shortest s->v path
    private DirectedEdge[] SecEdgeTo;
    private DirectedEdge[] SecFinalEdgeTo; // SecFinalEdgeTo[v] = last
       edge on second shortest s->v path
```

Svo bætti ég við eftirfarandi föllum í DijkstraSP.java

```
public double SecondDP(EdgeWeightedDigraph G, int s, int t)
    {
      Double SecFinalDistTo = Double.POSITIVE_INFINITY;
      DijkstraSP dsp = new DijkstraSP(G, s);

      for(DirectedEdge i : pathTo(t))
      {
        SecDistTo = new double[G.V()];
        SecEdgeTo = new DirectedEdge[G.V()];

        validateVertex(s);

        for (int v = 0; v < G.V(); v++)
            SecDistTo[v] = Double.POSITIVE_INFINITY;
        SecDistTo[s] = 0.0;

        // relax vertices in order of distance from s
        pq = new IndexMinPQ<Double>(G.V());
        pq.insert(s, SecDistTo[s]);
        while (!pq.isEmpty()) {
            int v = pq.delMin();
            for (DirectedEdge e : G.adj(v))
            {
              if(!e.equals(i)) SecRelax(e);
            }
        }

        // check optimality conditions
        assert check(G, s);
        if(SecDistTo[t] == dsp.distTo(t))
        {
          SecFinalDistTo = SecDistTo[t];
          SecFinalEdgeTo = Arrays.copyOf(SecEdgeTo, SecEdgeTo.length);
        }

      }
      if(SecFinalDistTo == Double.POSITIVE_INFINITY) return 0;
      return SecFinalDistTo;
    }

    // relax edge e and update pq if changed
    private void SecRelax(DirectedEdge e) {
        int v = e.from(), w = e.to();
        if (SecDistTo[w] > SecDistTo[v] + e.weight()) {
            SecDistTo[w] = SecDistTo[v] + e.weight();
            SecEdgeTo[w] = e;
            if (pq.contains(w)) pq.decreaseKey(w, SecDistTo[w]);
            else                pq.insert(w, SecDistTo[w]);
        }
    }

    public Iterable<DirectedEdge> SecPathTo(int v) {
        validateVertex(v);
        if (!hasPathTo(v)) return null;
        Stack<DirectedEdge> path = new Stack<DirectedEdge>();
        for (DirectedEdge e = SecFinalEdgeTo[v]; e != null; e =
           SecFinalEdgeTo[e.from()]) {
            path.push(e);
        }
        return path;
    }
```

Og keyrði eftirfarandi main fall á skránna

```
8
13
5 4  0.35
4 7  0.37
5 7  0.28
5 1  0.32
4 0  0.10
0 2  0.26
3 7  0.39
1 3  0.29
7 2  0.34
6 2  0.40
3 6  0.52
6 0  0.58
6 4  0.48

public static void main(String[] args) {
        In in = new In(args[0]);
        EdgeWeightedDigraph G = new EdgeWeightedDigraph(in);
        int s = Integer.parseInt(args[1]);

        // compute shortest paths
        DijkstraSP sp = new DijkstraSP(G, s);

        // print shortest path
        System.out.println("Shortest paths: ");
        for (int t = 0; t < G.V(); t++) {
            if (sp.hasPathTo(t)) {
                StdOut.printf("%d to %d (%.2f)  ", s, t, sp.distTo(t));
                for (DirectedEdge e : sp.pathTo(t)) {
                    StdOut.print(e + "   ");
                }
                StdOut.println();
            }
            else {
                StdOut.printf("%d to %d         no path\n", s, t);
            }
        }
        // print second shortest path
        System.out.println("Second shortest paths: ");
        for (int t = 0; t < G.V(); t++) {
            if (sp.hasPathTo(t)) {
              if(sp.SecondDP(G, s, t) == 0) StdOut.printf("%d to %d %s
                  \n", s, t, "No second shortest path");
              else
              {
                StdOut.printf("%d to %d (%.2f)  ", s, t,
                   sp.SecondDP(G, s, t));
                for (DirectedEdge e : sp.SecPathTo(t)) {
                    StdOut.print(e + "   ");
                }
                StdOut.println();
              }
            }
            else {
                StdOut.printf("%d to %d         no path\n", s, t);
            }
        }
    }
```

Og þá fékkst efirfarandi:

```
Shortest paths:
1 to 0 (1.39)  1->3  0.29   3->6  0.52   6->0  0.58
1 to 1 (0.00)
1 to 2 (1.02)  1->3  0.29   3->7  0.39   7->2  0.34
1 to 3 (0.29)  1->3  0.29
1 to 4 (1.29)  1->3  0.29   3->6  0.52   6->4  0.48
1 to 5         no path
1 to 6 (0.81)  1->3  0.29   3->6  0.52
1 to 7 (0.68)  1->3  0.29   3->7  0.39
Second shortest paths:
1 to 0 (1.39)  1->3  0.29   3->6  0.52   6->4  0.48   4->0  0.10
1 to 1 No second shortest path
1 to 2 No second shortest path
1 to 3 No second shortest path
1 to 4 No second shortest path
1 to 5         no path
1 to 6 No second shortest path
1 to 7 No second shortest path
```

Dæmi 5

```
public class SourceSinkSP
{
  private EdgeWeightedDigraph G;

  public SourceSinkSP(EdgeWeightedDigraph G)
  {
    this.G = new EdgeWeightedDigraph(G);
  }

  //Gefur lengd milli s og t
  public double distBetween(int s, int t)
  {
    DijkstraSP dsp = new DijkstraSP(G, s);
    return dsp.distTo(t);
  }

  //Gefur leidina milli s og t
  public Iterable<DirectedEdge> pathBetween(int s, int t)
  {
    DijkstraSP dsp = new DijkstraSP(G, s);
    Stack<DirectedEdge> path = new Stack<DirectedEdge>();
    for(DirectedEdge i : dsp.pathTo(t))
    {
      path.push(i);
    }
    return path;
  }
}
```