Advanced Algorithms

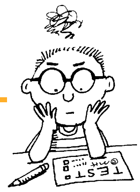
Floriano Zini

Free University of Bozen-Bolzano
Faculty of Computer Science

Academic Year 2013-2014

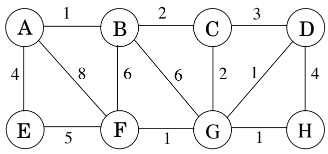
Lab 4 – Solutions of assignment

Assignment 03



Exercise 5.2.b page 148 DPV

Suppose we want to find the minimum spanning tree of the following graph



Run Kruskal's algorithm on the graph. Show how the disjoint-sets data structure looks at every intermediate stage (including the structure of the directed trees), assuming path compression is used

Assignment 03

Exercise 5.2.b page 148 DPV

```
procedure kruskal(G, w)
Input:    A connected undirected graph G = (V, E) with edge weights w_e
Output:   A minimum spanning tree defined by the edges X

for all u ∈ V:
   makeset(u)

X = {}
Sort the edges E by weight
for all edges {u, v} ∈ E, in increasing order of weight:
   if find(u) ≠ find(v):
      add edge {u, v} to X
      union(u, v)
```

Solution

Let's run the Kruskal's algorithm

1. Call makeset(u) for all u ∈ V →

   data structure: A^0 B^0 C^0 D^0 E^0 F^0 G^0 H^0

2. Initialize the solution →

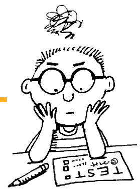
   X = {}

3. Sort edges in E by increasing weight →

   {(A,B):1,(D,G):1,(F,G):1,(H,G):1,(B,C):2,(C,G):2,
   (C,D):3,(A,E):4,(D,H):4,(E,F):5,(B,F):6,(B,G):6,(A,F):8}

NB: *(x,y):w* means that the edge *(x,y)* has weight *w*

Assignment 03

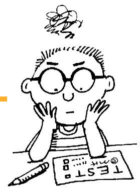


Exercise 5.2.b page 148 DPV

Solution (cont.)

4. Loop over ordered edges
   - (A,B)
     find(A)≠find(B); no compression; X={(A,B)};
     union(A,B)=({A^0,B^1},{A→B})
     data structure: ({A^0,B^1},{A→B})
   - (D,G)
     find(D)≠find(G); no compression; X={(A,B),(D,G)};
     union(D,G)=({D^0,G^1},{D→G})
     data structure: ({A^0,B^1,D^0,G^1},{A→B,D→G})
   - (F,G)
     find(F)≠find(G); no compression; X={(A,B),(D,G),(F,G)};
     union(F,G)=({D^0,F^0,G^1}, {D→G,F→G})
     data structure: ({A^0,B^1,D^0,F^0,G^1},{A→B,D→G,F→G})
   - (H,G)
     find(H)≠find(G); no compression; X={(A,B),(D,G),(F,G),(H,G)};
     union(H,G)=({D^0,F^0,H^0,G^1}, {D→G,F→G,H→G})
     data structure: ({A^0,B^1,D^0,F^0,H^0,G^1},{A→B,D→G,F→G,H→G})
   - (B,C)
     find(B) ≠find(C); no compression; X={(A,B),(D,G),(F,G),(H,G),(B,C)}
     union(B,C)=({A^0,C^0,B^1}, {A→B,C→B})
     data structure: ({A^0,C^0,B^1,D^0,F^0,H^0,G^1},{A→B,C→B,D→G,F→G,H→G})

Assignment 03

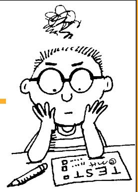


Exercise 5.2.b page 148 DPV

Solution (cont.)

4. Loop over ordered edges
   - (C,G)
     find(C)≠find(G); no compression;
     X={(A,B),(D,G),(F,G),(H,G),(B,C),(C,G)};
     union(C,G)=({A^0,C^0,D^0,F^0,H^0,B^1,G^2},{A→B,C→B,D→G,F→G,H→G,B→G})
     data structure: ({A^0,C^0,B^1,D^0,F^0,H^0,G^2},{A→B,C→B,D→G,F→G,H→G,B→G})
   - (C,D)
     find(C)=find(D); compression:({A^0,C^0,D^0,F^0,H^0,B^1,G^2},{A→B,C→G,D→G,F→G,H→G,B→G})
     X={(A,B),(D,G),(F,G),(H,G),(B,C),(C,G)};
     data structure: ({A^0,C^0,D^0,F^0,H^0,B^1,G^2},{A→B,C→G,D→G,F→G,H→G,B→G})
   - (A,E)
     find(A)≠find(E); compression:({A^0,C^0,D^0,F^0,H^0,B^1,G^2},{A→G,C→G,D→G,F→G,H→G,B→G})
     X={(A,B),(D,G),(F,G),(H,G),(B,C),(C,G),(A,E)}
     union(A,E) = ({A^0,C^0,D^0,F^0,H^0,E^0,B^1,G^2},{A→G,C→G,D→G,F→G,H→G,B→G,E→G})
     data structure: ({A^0,C^0,D^0,F^0,H^0,E^0,B^1,G^2},{A→B,C→G,D→G,F→G,H→G,B→G,E→G})
   - (D,H),(E,F),(B,F),(B,G),(A,F)
     find(x)=find(y); no compression;
     X={(A,B),(D,G),(F,G),(H,G),(B,C),(C,G),(A,E)}
     data structure: ({A^0,C^0,D^0,F^0,H^0,E^0,B^1,G^2},{A→G,C→G,D→G,F→G,H→G,B→G,E→G})
5. MST = ({A,B,C,D,E,F,G,H},{(A,B),(D,G),(F,G),(H,G),(B,C),(C,G),(A,E)})
   cost(MST)=1+1+1+1+2+2+4=12

Assignment 03

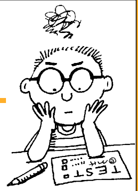


Exercise 5.9 page 149 DPV (partially)

The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph *G = (V, E)* is undirected. Do not assume that edge weights are distinct unless this is specifically stated.

1. If graph *G* has more than |*V*|−*1* edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree
2. If *G* has a cycle with a unique heaviest edge *e*, then *e* cannot be part of any MST
3. Let *e* be any edge of minimum weight in *G*. Then *e* must be part of some MST
4. If the lightest edge in a graph is unique, then it must be part of every MST.
5. If *e* is part of some MST of *G*, then it must be a lightest edge across some cut of G.

Assignment 03



Exercise 5.9 page 149 DPV (partially)

Solution

1. **False**, consider the case where the heaviest edge is a bridge (is the only edge connecting two connected components of *G*).
2. **True**, consider removing e from the MST and adding another edge belonging to the same cycle. Then we get *e* new tree with less total weight.
3. **True**, *e* will belong to the MST produced by Kruskal's algorithm.
4. **True**, if not there exists a cycle connecting the two endpoints of *e*, so adding *e* and removing another edge of the cycle, produces a lightest tree
5. **True**, consider the cut that has *u* in one side and *v* in the other, where *e=(u, v)*.