CS 5633 Analysis of Algorithms – Spring 05

3/31/05

6. Homework

Due **4/12/05** before class

1. **Minimum edge (3 points)**
   Let G be a connected undirected graph with weight function $w : E \to \mathbb{R}_0^+$ (i.e., all edge weights are ≥ 0). Assume edge weights are distinct. Let e^* be the cheapest edge, i.e., $w(e^*) < w(e)$ for all $e \in E$ with $e \neq e^*$.

   Is it true that there is a minimum spanning tree T of G that contains the edge e^*? If yes, justify your answer. If no, give a counterexample.

2. **Adding an edge (5 points)**
   Let $G = (V, E)$ be a connected undirected graph with weight function $w : E \to \mathbb{R}_0^+$ (i.e., all edge weights are ≥ 0). Assume edge weights are distinct. Further, let a minimum spanning tree T on G be given.

   Now, assume that one new edge (u, v), with $u, v \in V$, with weight $w(u, v)$ is added to G. (This weight is different from all other edge weights.)

   Give an efficient algorithm to test if T remains the minimum spanning tree for this new graph. Your algorithm should run in $O(|E|)$ time. Can you make it run in $O(|V|)$ time?

3. **Ackermann (2 points)**
   What is the value of $\alpha(10^8)$? Justify your answer.

4. **Union-Find (4 points)**

   ```
   for i:=1 to 16 do MAKE-SET(x[i])
   for i:=1 to 13 by 3 do UNION(x[i], x[i+1])
   for i:=1 to 11 by 5 do UNION(x[i], x[i+4])
   UNION(x[1],x[6])
   UNION(x[11],x[13])
   UNION(x[7],x[13])
   UNION(x[1],x[7])
   FIND-SET(x[2])
   FIND-SET(x[14])
   ```

   Assume an implementation of the Union-Find data structure with a disjoint-set forest with union-by-weight and path compression.

   Show the data structure after every for-loop, as well as after the last union operation, and the final data structure. What are the answers to the `FIND-SET` operations?

5. **LEDA (12 points)**
   Implement one of the two algorithms:

(a) Prim's MST algorithm using LEDA priority queues

(b) your algorithm from problem 2)

In order to test your implementation use the demo program

```
gw_min_spanning_tree.c
```

You will find this on any linux machine in

```
/usr/local/LEDA-4.5/demo/graph_alg/
```

When setting the environment variable $LEDAROOT to

```
/usr/local/LEDA-4.5
```

you should be able to run the precompiled executable

```
/usr/local/LEDA-4.5/demo/graph_alg/gw_min_spanning_tree
```

on any linux machine. This program provides a graphical user interface (GUI) which allows to create and load graphs, manipulate them, etc. And it displays the minimum spanning tree computed with the built-in LEDA implementation.

The program compiles on any linux machine using -I and -L to point to $LEDAROOT and using the libraries -lL -lG -lP -lD3 -lGeoW -lW -lX11 -lm . (In order to find the proper libX11.so I needed to create a symbolic link to /usr/X11R6/lib/libX11.so.6 but maybe there is a better way.)

Your implementation should use

```
gw_min_spanning_tree.c
```

If you implement Prim's algorithm then you should replace the call to the minimum spanning tree algorithm with a call to your algorithm. If you implement the algorithm from problem 2) you will have to modify the function

```
void new_edge_handler(GraphWin& gw, edge e)
```

in order to call your algorithm after a new edge has been inserted.

Please email me the source code and a screenshot of an example run: If you implement Prim's algorithm please provide screenshots of an MST computed with the built-in LEDA algorithm and the MST computed with your algorithm, for the same graph. If you implement the algorithm from problem 2) please provide a screenshot of the MST before and after the edge insertion.