§ Graduate Algorithms §

Homework 2

Elisabeth Crawford, Virginia Vassilevska

January 27, 2004

1 Question 1

1.1 Part a

To show that M is a matroid we need to demonstrate the following:

1. $S \in I$ and $S^{'} \subseteq S$ implies $S^{'} \in I$
2. $S, S^{'} \in I$ and $|S^{'}| \leq |S|$ implies $\exists x \in S - S^{'}$ s.t $S^{'} \cup x \in I$

Let $S \in I$, then:

1. Suppose $|S| = 3$, then the size of any subset of S, $S^{'}$ is either less than 3 or $S = S^{'}$. If $S = S^{'}$ then clearly $S^{'} \in I$, else $|S^{'}| < 3$ and $S^{'} \in I$ by definition.

   Further, suppose $|S| < 3$ then for any subset $S^{'}$, $|S^{'}| < 3$ and $S^{'} \in I$ by definition.

2. Suppose $|S| = 3$, we consider $S^{'} \in I$ s.t $|S^{'}| < 3$.

   (a) Suppose $|S^{'}| = 2$, then the two points in $S^{'}$ uniquely define a line. Now since $|S| = 3$ and $S \in I$, no line passes through all three points of S by definition. Hence at least 1 point in S does not lie on the unique line passing through the points in $S^{'}$. Hence $\exists x \in S$ s.t $S^{'} \cup \{x\} \in I$.

   (b) Suppose $|S^{'}| = 1$ then clearly $\exists x \in S$ s.t $x \notin S^{'}$ and $\forall x \in S - S^{'}$ $|S^{'} \cup \{x\}| = 2 < 3$ and thus $S^{'} \cup \{x\} \in I$.

   (c) Suppose $|S^{'}| = 0$, then $\forall x \in S$ $|S^{'} \cup \{x\}| = 1 < 3$ and thus $S^{'} \cup \{x\} \in I$.

   Suppose $|S| = 2$, $|S^{'}| < |S|$ implies that $|S^{'}| = 0$ or 1. As such, by our argument above $\exists x \in S$ s.t $S^{'} \cup \{x\} \in I$.

   Suppose $|S| = 1$ then $|S^{'}| = 0$ and by the same argument as before, $\exists x \in S$ s.t $S^{'} \cup \{x\} \in I$.

The circuits of M are: $\{a, b, f\}, \{a, c, e\}, \{a, d, g\}, \{c, f, g\}, \{b, e, g\}, \{b, c, d\}, \{d, e, f\}$ (the lines in the Fano plane).

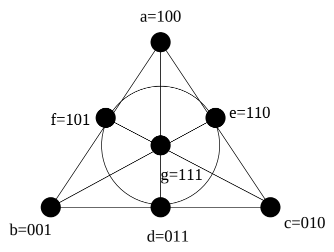

Figure 1:

1.2 Part b2

Let the field be GF(2). Then the corresponding matrix is:

$$\mathbf{N} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Here the columns correspond to a, b, c, d, e, f, g in that order. Note that since we are in GF(2) the only possible coefficients in a linear dependence are 0 and 1. No two columns are thus dependent since no two sum to the zero vector. The only triples of columns which sum to the zero vector are those corresponding to the circuits of M. Therefore the independent sets of N have exactly the same structure as I.

2 Question 2

2.1 Part a

2.1.1 i

Consider a vertex v, then the size of each quantile for v is in $O(\frac{deg(v)}{p})$. Thus, summing up over successful quantiles, the total time due to successes is $O(\sum_v \frac{deg(v)}{p})$. Now the sum of the degrees of the vertices is in $O(2|E|)$, so the time for successes is in $O(\frac{2|E|}{p}) \in O(\frac{m}{p})$, as required.

2.1.2 ii

Every edge can fail at most once since we remove it when it fails. Thus the time due to failures is in $O(m)$ as required.

2.2 Part b

Constructing the quantiles takes $\sum_v deg(v)\ log(p) \in O(m\ log(p))$.

Now there are $O(log(n))$ stages. At every stage we have to visit each vertex. The cost of successes is $O(\frac{m}{p})$. So in total we have $O(log(n)\ (\frac{m}{p}+n)$. The total time due to failures is $O(m)$, so in total we have a running time of $O(log(n)\ (\frac{m}{p}+n))+O(m\ log(p))+O(m)$ which we can simplify to $O(log(n)\ (\frac{m}{p}+n)+m\ log(p))$ as required.
Now consider $p=\lceil log(n)\rceil$, then we have:

$$log(n)(\frac{m}{log(n)}+n)+m\ log\ log(n)$$

which equals,

$$m+n\ log(n)+m\ log\ log(n)$$

Now if $m \geq n\ log(n)$ then this is clearly in $O(m\ log\ log(n))$, as required.

2.3 Extra Credit

We apply the algorithm in problem 4 of homework 1 for $log(log(n))$ stages, which results in $\frac{n}{log(n)}$ components. This takes $O(m\ log\ log(n))$ time. Now view each of the components as a vertex and run the new algorithm on the graph with vertex set these components and edge set the edges between them (but not inside them).

We have from b, the time bound is $O(m+\frac{n}{log(n)}\ log(\frac{n}{log(n)})+m\ log(log(\frac{n}{log(n)}))$. This is in $O(m\ log\ log\ n+\frac{n}{log(n)}\ log(\frac{n}{log(n)}))$.

What we need to show, is that $O(\frac{n}{log(n)}log(\frac{n}{log(n)}))\in O(m\ log\ log(n))$. Now

$$\frac{n}{log(n)}log(\frac{n}{log(n)})=\frac{n}{log(n)}(log(n)-log\ log(n))$$

We can rearrange this expression to:

$$n-\frac{n\ log\ log\ n}{log\ n}$$

This is clearly in in $O(n)$ and since the graph is connected in $O(m\ log\ log(n))$ time as required.

3 Question 3

Given a set of n points $\{p_1,p_2,...p_n\}$ that are ordered by their x coordinate and where no two points have the same x coordinate and no three points are co-linear, we can compute the convex hull in linear time using the following algorithm.

The algorithm works by incrementally building the upper and lower halves of the polygon that forms the convex hull. It is based on the observation that when tracing a clockwise path around a convex hull, any three sequential points must form a right hand turn. If this is not the case, we violate the convexity requirement. When we speculatively add a new point to the hull, we test

```
1.  upper-hull.append($p_1$), upper-hull.append($p_2$)
2.  for $i = 3$, to $n$, do:
3.      upper-hull.append($p_i$)
4.      while size(upper-hull) $>$ 2 and not right-turn(tail(upper-hull, 3)), do:
5.          delete(upper-hull, -2) (i.e. delete second last element)
6.  lower-hull.append($p_n$), lower-hull.append($p_{n-1}$)
7.  for $i = n - 2$ down to 1, do:
8.      lower-hull.append($p_i$)
9.      while size(lower-hull) $>$ 2 and not right-turn(tail(lower-hull, 3)), do:
10.         delete(lower-hull, -2)
11. delete(lower-hull, 1), delete(lower-hull, n)
12. return upper-hull.append(lower-hull)
```

whether it causes this right hand turn rule to be invalidated (see lines 4,9). If so, we remove the middle out of the three elements. However, we now have to check the new final three elements, and so forth, until either there is only two elements in hull or the last three we checked were ok.

We now consider the complexity of our algorithm. Notice that the loops begun at lines 2 and 7 will execute less than n times. Now consider the two nested loops beginning at lines 4 and 9. Notice that for each execution, these loops delete one of the n elements. Thus, the *total time* spent in each of these loops for the n executions of the lines 2 and 7 loops is n. Thus, since all other operations require constant time, we have a complexity of $O(n) + O(n)$ for the upper-hull and $O(n) + O(n)$ for the lower-hull. Hence the algorithm runs in linear time as required.