CS 880: Advanced Complexity Theory	31-Mar-06
Lecture 23: Static Membership Problem	
Instructor: Dieter van Melkebeek	Scribe: Prem Mansukhani and Matt Anderson

The purpose of today's lecture is to describe a data structure that provides an effiecient means to determine if an element is part of a universe, or a specific subset of a universe.

1 The Goal

The setting for this problem is a Universe U of size N in which we want to to create a data set that allows us to determine if an element is part of that Universe U, and where the scheme also works for all subsets $S \subseteq U$ with $|S| \leq K$ To accomplish this we must complete two phase;

1. Construct Data Structure: given a particular S construct $\phi(S)$. From this point on this structure is static.
2. Query Data Structure: use the structure $\phi(S)$ to answer membership queries such as: is $u \in S$. This takes the form of $\phi(S) \subseteq \{0,1\}^M \to \{0,1\}$

The goal while constructing $\phi(S)$ is to

- minimize its size, $|\phi(S)| = M$, while
- using as few elements of $\phi(S)$ as possible, say q, per query.

1.1 Extremes

The problem seems to require us to minimze two opposing traits. If we go to one extreme and minimize the number of elements we probe, we can set $q = 1$, but by doing so we require that we have as many symbols as we have elements in our universe, $M = N$. if we go to the other extreme and minimize M, then

$$M \geq \log\left(\sum_{i=0}^{k} \binom{N}{i}\right). \tag{1}$$

We know we can not any smaller because we need an M large enough so that we do not get the same answer for every query. So, we know that

$$\left(\frac{N}{K}\right)^K \leq \sum_{i=0}^{k} \binom{N}{i} \leq \left(\frac{eN}{K}\right)^K \tag{2}$$

therefore with some simple munipulation:

$$\log\left(\sum_{i=0}^{k} \binom{N}{i}\right) = \theta\left(k \log\left(\frac{N}{K}\right)\right) \tag{3}$$

So if $K \leq N^{1-\zeta}$ for some $\zeta > 0$ then we can state that

$$M = \log\left(\sum_{i=0}^{k}\binom{N}{i}\right) = \theta\left(k\log\left(N\right)\right) \tag{4}$$

This is possible by explicitly storing every element in S, but to do this you need to order all the element n S and list them lexigraphically.

Now we need to figure out how many probes are needed if we use the smallest M possible. The Naíve approach would be to query ever bit, but this would lead to K queries, we can do better. The next approach would be to use a binary search. This would make $O(\log k)$ queries, but would require us to use a sorted list, meaning we would have to make a total of $q = \theta((\log k)(\log N))$ probes. If we only store explicitly the elements of the set, then this is the best result possible. We can still do better though, if store a little bit more information. For example if we use FKS hashing functions we can realize an $M = O(K\log N)$ and use only $q = O(\log N)$ during the queries.

1.2 The best of both worlds

If we can use some randomness we can combine the best of both a small M and a small q. The rub is that this new approach is now probablistic, so we will get the correct result with probability, for a particular query, of $1-\epsilon$ where we can set ϵ as close to zero as we wish. When designing this new construction we mush keep two newrequirements in mind. First, during phase one, ϕ must be constructed in a deterministic manner. Second, during phase two, for any $u \in U$: $Pr[WrongMembership] \leq \epsilon$.

2 Construction

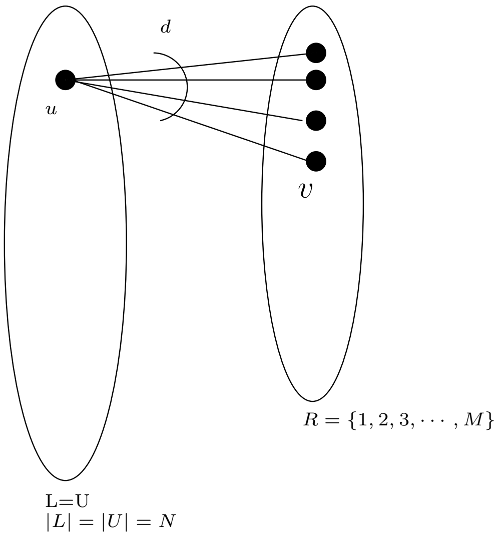

Figure 1: Mapping between Universe U and assignment $\phi(S)$

We will define a very simple algorithm that will provide all the properties we talked about in the last section.

Query Algorithm:

- given $u \in U$
- pick a $v \in \Gamma(u)$ uniformally at random, then
- answer $\phi(S)[v]$

Where $\phi(S)$ is an assignment to R. The quesion remaining is it possible to pick a $\phi(S) \in \{0,1\}^M$ for any fixed $S \subseteq U$. These properties can be generalized to disjoint sets $U_0, U_1 \subseteq U$with $\min(|U_0|, |U_1|) \leq k$. Then for all $u \in U_0 \cup U_1$:

1. if $u \in U_1$ then $\frac{|\Gamma(u) \cap \overline{\phi(U_1)}|}{d} < \epsilon$
2. if $u \in U_0$ then $\frac{|\Gamma(u) \cap \phi(U_1)|}{d} < \epsilon$

This problem can be solved by recursively solving the problem onsmaller instances. This will be shown by induction on the side of$|U_0| + |U_1|$. Without loss of generality assume $|U_1| \leq |U_0|$.

First, consider assigning 1 to all $v \in R$ such that $v \in \Gamma(U_1)$ and 0 for all other bits. Then clearly property onestill holds because the assignment is correct for all $u \in U_1$.This assignment may not satisfy property 2 however.

Let B be the set of bad elements (elements such that our queryproduces the incorrect result):

$$B = \{u \in U_0 | |\Gamma(u) \cap \Gamma(U_1)| > \epsilon d\} \tag{5}$$

Claim 1. *$|B| < |U_1|$, provided G is $(2k, d(1 - \frac{\epsilon}{2}))$-expanding.*

Proof. Assume the opposite, pick some $B' \subseteq B$ with $|B'| = |U_1|$

Then, using the expansion property and the definition of B a contradiction is created:

$$\begin{aligned} d(1 - \frac{\epsilon}{2}) \cdot 2|U_1| \leq & |\Gamma(U_1 \cup B')| \\ & |\Gamma(U_1 \cup B')| && < d|U_1| + d|B'| - \epsilon d|B'| \\ d(2 - \epsilon)|U_1| \leq & && < d(2 - \epsilon)|U_1| \end{aligned} \tag{6}$$

□

Now we can recursively find an assignment by solving the smallersubproblem $(U_0', U_1') = (B, U_1)$. Note that $|U_0'| + |U_1'| < 2|U_1| \leq |U_0| + |U_1|$. The base case is an exercise left to thereader.

This construction will provide us with a datastructure that is of size$M = |R| = O(\frac{k}{\epsilon^2} \log N)$, though we do not have timeto prove it.

Remember, for a given N, k and ϵ, we want to make M assmall as possible. It is possible to acheive $M = O(k\frac{d}{\epsilon})$with $d = \frac{\log N}{\epsilon}$, giving $M = O(\frac{k}{\epsilon^2} \log N)$ using the probabilistic method. The trivial derandomizationof the query scheme $\forall \epsilon < \frac{1}{2}$ (probe all of$\Gamma(u)$ and take the majority vote) yields a deterministic schemewith optimal M (up to a constant factor) and $O(\log N)$ bitprobes. This matches the bit probe complexity of the FKS scheme.However, in the FKS scheme the data structure $\phi(S)$ is dividedinto a table of $O(k)$ cells of $O(\log N)$ bits each and eachmembership query only visits $O(1)$ cells. In the deterministicscheme we obtain by derandomizing our randomized schere, the $O(\log N)$ bit probes are scatter all over $\phi(S)$. The best known explicit construction (those computable indeterministic polynomial time in N) also give $M = O(k\frac{d}{\epsilon})$, but with $d = \left(\frac{\log N}{\epsilon}\right)^{O(1)}$, giving $M = O\left(k \left(\frac{\log N}{\epsilon}\right)^{O(1)}\right)$.