(CS602) Applied Algorithms 06 Jan, 2014

Lecture 1: Introduction

Lecturer: Nutan Limaye *Scribe: Nutan Limaye*

This course will focus on algorithms for large data. The course is vaguely divided into four modules.

- Algorithms for large datasets – frequency moments computations.
- Algorithms for problems in linear algebra – dimension reduction, norm computation.
- Algorithms for graph problems.
- Lower bound results via information theory.

The first two modules will be discussed before the midsem and the last two after midsem. The grading scheme will be uploaded on the Moodle page of the course.

1.1 Introduction

We begin with two simple problems:
Problem 1: Given a stream of numbers, output the length of the input stream.
Problem 2: Given a graph as a set of edges, output the maximum matching in a graph.

Problem 1 is easily solvable in space $O(\log n)$, where n is the length of the input. Problem 2 has a polynomial time algorithm. Therefore, both the problems are *easy* in the classical model of computation. However, the model that we will study in this course is slightly restrictive:

- The input data is *very large* as compared to the space available to the algorithm.
- Rereading input bits is *very expensive.*
- Time for computation per input bit is *very small.*

Typically, if the input is of length, say n, we will be required to design algorithms which use space $o(n)$ and read every input bit $O(1)$ times. Under these assumptions, let us first design a 2-approximation algorithm for Problem 2.

1.2 Problem 2

Consider a simple greedy algorithm which starts with an empty set M and adds a new edge say e to M iff $M \cup \{e\}$ is a matching. This algorithm looks at every edge exactly once, uses space $O(n \log n)$ which is $o(n^2)$. However, this does not necessarily output the maximum matching. For every edge it adds to M, it may in the worst case exclude 2 edges from the optimal matching. Therefore, this algorithm gives a 2-approximation.

1.3 Problem 1

For Problem 1 the naive algorithm already has all the desired properties, i.e. it looks at every input element exactly once, and uses space $O(\log n)$ which is $o(n)$. However, one could ask whether we can reduce the space used by the algorithm even further. It is easy to see that if we wish to compute the length of the input exactly, then $\Omega(\log n)$ bits are necessary. We now discuss a randomized algorithm design by Morris [**?**], which gives a 2-approximation with probability at least 3/4 and uses space $O(\log \log n)$.

```
$Y \leftarrow 0$;
while there exists $x_i$, an input element do
    $Y \leftarrow Y + 1$ w.p. $\frac{1}{2^Y}$;
    $Y \leftarrow Y$     w.p. $1 - \frac{1}{2^Y}$;
end
Output $2^Y$
```

We will analyze the algorithm in the next class.