CMSC 351 - Introduction to Algorithms
Spring 2012
Lecture 10

Instructor: MohammadTaghi Hajiaghayi
Scribe: Rajesh Chitnis

1 Introduction

In this lecture we will look at Bucket Sort, Radix Sort and Lexicographic Sort.

2 Bucket Sort

The simple mailroom sort is to allocate a sufficient number of "boxes", called buckets, and put each element in the corresponding bucket. This is called as bucket sort.

If the elements are letters and need to be sorted according to states then we need 50 states, however if it is according to zipcodes (5 digits) then we need 100,000 buckets. Thus bucket sort works very well only for elements from a small, simple range known in advance. Say if we have n elements ranging from 1 to m. We need m buckets and for each i, we put x_i in the bucket corresponding to its value in total $O(n)$ time. At the end we scan the buckets in $O(m)$ time to collect all the elements. Thus we need a total of $O(n + m)$ time and $O(m)$ space which is not good for $m >> n$.

3 Least Significant Digit Radix Sort

The Least Significant Digit Radix Sort algorithm works in the following steps:

1. Take the least significant digit of each key (say the number of records to be sorted).
2. Group the keys based on that digit, but otherwise keep the original order of the keys.
3. Repeat the grouping process with each more significant digit

Radix sort is $\underline{\text{stable sort}}$, i.e., it maintains the relative order of records with equal keys. Step 2 is usually done using bucket sort, which is efficient in this case since there are usually only a small number of digits.

The running time of the algorithm is $O(nk)$ where k is the maximum key length, since we need $O(k)$ repetitions of the loop (each step of the loop requires just a single pass over data). We can implement this by a linked list. However by first counting the number of keys that belong to each bucket before moving the keys into those buckets, and storing them in an array we can indeed implement this by an array as well (using pointers). The pointers here are integers (indices) and we move them after each insert. We can prove the correctness using induction: the hypothesis is that we know how to sort elements with $< k$ digits.

Example: Suppose we want to sort the numbers 170, 45, 75, 90, 802, 24, 2, 66. We have the following steps:

- Sort by least significant digit: 17$\underline{0}$, 9$\underline{0}$, 80$\underline{2}$, $\underline{2}$, 2$\underline{4}$, 7$\underline{5}$, 4$\underline{5}$, 6$\underline{6}$
- Next: 8$\underline{0}$2, 2, $\underline{2}$4, $\underline{4}$5, $\underline{6}$6, 1$\underline{7}$0, $\underline{7}$5, $\underline{9}$0
- Sort by last digit: 2, 24, 45, 66, 75, 90, $\underline{1}$70, $\underline{8}$02

4 Lexicographic Sort (Most Significant Digit Sort)

It can be used to sort keys (especially strings) in lexicographic (dictionary) order. The Most Significant Digit Radix Sort algorithm works in the following steps:

1. Take the most significant digit of each key.
2. Sort the list of elements based on that digit, grouping elements with the same digit into one bucket.
3. Recursively sort each bucket, starting with the next digit to the right.
4. Concatenate (append) the buckets together in order.

Implementation: Again a two-pass method can be used to first find out how big each bucket needs to be and then place each key into the appropriate bucket using pointers. A single pass can also be used by linked lists. We could also use this algorithm to sort the integers if we pad them with zeros. Again the running time is $O(nk)$. The current implementation of lexicographic sort is not in-place but there are some simple in-place implementations for lexicographic sort.

Example: Suppose we want to sort the strings cab, ab, bb, cd, a, c, cda. We have the following steps:

- $\underline{\text{a}}$b, $\underline{\text{a}}$, $\underline{\text{b}}$b, $\underline{\text{c}}$ab, $\underline{\text{c}}$d, $\underline{\text{c}}$, $\underline{\text{c}}$da
- a, a$\underline{\text{b}}$, b$\underline{\text{b}}$, c, c$\underline{\text{a}}$b, c$\underline{\text{d}}$, c$\underline{\text{d}}$a
- a, ab, bb, c, ca$\underline{\text{b}}$, cd, cd$\underline{\text{a}}$

References

[1] Udi Manber, *Introduction to Algorithms - A Creative Approach*