CSc445
Brief Class Notes

In this note, I will provide some brief descriptions and pointers to the material discussed in class, and note coved in the slides handouts. I will try to give some useful information, but these notes cannot replace being in class and/or read the text.

Asympthotic notations and recursive formulas After a brief introduction, we discussed the assymptotic running time notations: big-O, Ω and Θ. We showed seval examples. This material is well covered in the text-book, and hence we do not provide more details here. We analysed the running time of a few examples. One, of particular interset, is the following

```
read(n)
for( i = 1 ; i ≤ n ; i++ )
    for( j = 1 ; j ≤ n ; j+= i )
        print( "*" ) ;
```

We showed that its running time is $\sum_1^n 1/i = O(n \log n)$.

Next we moved to recursive formulas. We analyzed using the iterative method (see details below) simple recursions, such as the function *NoNeed(n)* .

```
NoNeed(n)
read(n)
for( i = 1 ; i ≤ n ; i++
```

Whose reursion formula is $T(n) = cn + T(n-1)$, (for some constant c) and $T(1) = c$. Here as easily seen,

$$T(n) = cn + T(n) =$$

$$Tcn + (c(n-1) + T(n-1)) =$$

$$cn + c(n-1) + c(n-2) + T(n-2) = \ldots \text{ after } k \text{ stages}$$

$c(n+(n-1)+(n-2)+(n-3)+(n-k))+T(n-k) =$ setting $k = n-1$

$$c(n+(n-1)+(n-2)+\cdots+2)+T(1) = c\sum_{i=1}^{n} i = cn(n+1)/2$$

Hence $T(n) = \Theta(n^2)$.

Stable Marriage Algorithm – please check the slides handout.

SkipList I taught a version of the skip list at which each element is a single cell. See slides handouts. In class we have also discussed briefly a slightly different version, at which each element is represented as a small array. If more details are needed, you could check the textbook " Structures & Their Algorithms, (Lewis & Denenberg)."

Some facts about the first version:

1. The expected number of levels in the SL is $O(\log n)$.
2. The expected size (memory used) is $O(n)$.
3. The expected search time is $O(\log n)$. This dominates the expected time for insertion and deletion operations, and also of **successor** operation. The operations $succ(x)$ finds the smallest element in the data structure which is strictly larger than x.

Augmenting data structures We demonstrated this structure on binary search trees. The idea was to assign extra information (to augment the tree) so each node v of contains some extra information — in the example shown in class v also stores the number of keys in v's subtree.

These fields can also help is find the k'th smallest element in the list, in time $O(\log n)$, or to report the rank of any key x in $O(\log n)$.

Details about this implementation in search trees can be find the chapter on augmented data structures in the textbook.

Quicksort and median selection in an array $A[1..n]$. Defined a *good pivot* p as a pivot that at least %10 of the keys in A are smaller than p, and at least %10 of the keys in A are larger than p. Discuss the 5-random-elements method, and the probability that a pivot picked using this method is a good one. Saw the applications to the expected time of quick sort and of median selection.

Rigorous Analyzing the running time of QuickSort (not covered in the sylibous, and not required for the exam). Consider sorting the keys $\{k_1 \ldots k_n\}$. Assume that We assume that we use a version of Quick-Sort at which all elements are different, and the probability of each

element to be picked as a pivot is uniform. We showed that the running time is proportional to the number of pairs of elements which are compared to each other (which happens when one of these elements is the pivot). We define a random variable X_{ij} which is 1 is at some point at the course of the algorithm k_i and k_j are compared, and 0 otherwise. Note that $E(X_{ij})$, the expected value of X_{ij} is

$$E(X_{ij}) = 1 \cdot Pr(X_{ij} = 1) \ + 0 \cdot Pr(X_{ij} = 0) = Pr(X_{ij} = 1) \ .$$

Note that the running time is $O(\sum_{1 \le i < j \le n} X_{ij})$, and the expected running time is

$$E\left(\sum_{1 \le i < j \le n} X_{ij}\right) = \sum_{1 \le i < j \le n} E\left(X_{ij}\right) = \sum_{1 \le i < j \le n} Pr(X_{ij} = 1).$$

To evaluate $Pr(X_{ij} = 1)$, we only need to note that the algorithm compares k_i and k_j if and only if either k_i or k_j were the first key to be picked as pivot, from the set $\{k_i, k_{i+1} \ldots k_j\}$. Since each of them has the same probability to be picked, and there are $j - 1 + 1$ keys between (and including) k_i and k_j, we figure that $Pr(X_{ij} = 1) = \frac{2}{j-i+1}$. Thus

$$E\left(\sum_{1 \le i < j \le n} X_{ij}\right) = \sum_{1 \le i < j \le n} \frac{2}{j - i + 1}.$$

Since from the Harmonic sum formula we know that $1 + \frac{1}{2} + \frac{1}{3} + \ldots \frac{1}{n} = O(\log n)$, we conclude that

$$E\left(\sum_{1 \le i < j \le n} X_{ij}\right) = \sum_{1 \le i < j \le n} \frac{2}{j - i + 1} \ = \ n \cdot O(\log n) \ = \ O(n \log n)$$

Hashing (from the slides). We discussed the common applications to storage items in a hash table. Chain hashing vs. Open addressing — pros and cons of each.

On the whiteboard, we discussed the 'obsession' for primes. Analyse the behavior of the sequence $((i \cdot a) \mod m)$ where $i = 0, 1 \ldots, m - 1$, $0 < a \le m - 1$ is an integer and m is a prime.

We will see usages of hashing to generating random numbers, Bloom Filter and string matching.