1. In a Parallel Disks Model (PDM) there are $D$ disks. In one parallel I/O we can bring a block (of size $B$) of elements from each of the disks. We typically assume that $M$ is a constant multiple of $DB$. We briefly described the DSM and SRM algorithms for sorting on the PDM. We then introduced the $(\ell, m)$-merge sort (LMM) algorithm and showed that it can be used to sort $N$ given elements in no more than $\left\lfloor \frac{\log(\frac{N}{M})}{\log(\min(\sqrt{M}, B))} \right\rfloor + 1$ number of passes through the data.

2. Suffix tree is a powerful data structure that can be used to perform a variety of operations on strings and much more. We showed the following results: 1) Given a text $T$ and a pattern $P$ we can search for $P$ in $T$ in $O(m + n)$ time where $m = |T|$ and $n = |P|$; 2) Given a text $T$ and a set $P = \{P_1, P_2, \ldots, P_q\}$ of patterns, we can find all the occurrences of all the patterns in $T$ in $O(m + N + K)$ time where $m = |T|$, $N$ is the total size of all the patterns and $K$ is the total number of occurrences of all the patterns in $T$; 3) Given a database DB of texts $\{T_1, T_2, \ldots, T_k\}$ and a set of patterns $P = \{P_1, P_2, \ldots, P_q\}$, we can find occurrences of all the patterns in DB in $O(M + N + K)$ time where $M$ is the total size of all the texts in DB, $N$ is the total size of all the patterns, and $K$ is the total number of occurrences of all the patterns in DB; 4) Given two strings $S_1$ and $S_2$, we can find the longest common substring between them in $O(|S_1| + |S_2|)$ time; 5) Given two strings $S_1$ and $S_2$ and an integer $l$, we can find all the substrings of $S_2$ of length $\geq l$ that occur in $S_1$ in $O(|S_1| + \sum_{i=1}^{q} |C_i|)$ time; 6) Given a string $S_1$, a collection of strings $C_1, C_2, \ldots, C_q$ and an integer $l$, we can find all the occurrences of $C_i$ of length $\geq l$ in $S_1$ (for $1 \leq i \leq q$) in $O(|S_1| + \sum_{i=1}^{q} |C_i|)$ time; 7) Given a set of strings $S_1, S_2, \ldots, S_n$ we can compute $l[2 : n]$ such that $l(i)$ is the length of the longest common substring that occurs in at least $i$ strings (for $2 \leq i \leq n$) in $O(Mn)$ time where $M$ is the total length of the $n$ strings; and 8) Given $n$ strings of total length $M$, we can solve the all pairs suffix-prefix problem in $O(M + n^2)$ time.

3. We showed that a suffix array on a given string of length $m$ can be constructed in $O(m)$ time. We can use the suffix array and the longest common prefix (LCP) array to search for a pattern $P$ in a text $T$ in $O(n + \log m)$ character comparisons, where $m = |T|$ and $n = |P|$. We also pointed out that we can compute the LCP array (for pairs of interest in string matching) in $O(m)$ time.