CSE 4502/5717: Big Data Analytics
Prof. Sanguthevar Rajasekaran
Notes by Smit Shah
Lecture 10- 02/26/2018

(l, m)-merge algorithm
Case 1: $M/B \geq \sqrt{M}$ $l = m = \sqrt{M}$
Let $K = \sqrt{M}$;
Let T(i, j) = # of passes needed to merge i sequences of length j each. We can sort N elements using the following algorithm:

1) Form runs each of length M; This takes one pass
2) Merge N/M runs of length M each using the (l, m)-merge algorithm.

We want to compute T(N/M, M).

$N/M = K^{2C} \rightarrow C = \frac{\log\left(\frac{N}{M}\right)}{\log M}$

$2C \log K = \log(N/M)$

$C = \frac{\log\left(\frac{N}{M}\right)}{2 * \log K}$.

For case 1, we use: $l = m = \sqrt{M}$; It is easy to see that

$T(N/M, M) = T(K,M) + T(K,KM) + T(K, K^2 M) + T(K, K^{2C-1}, M)$ ----- eq 1

Now consider the problem of merging K sequences of length K^iM each, for any i. This merging can be done using the (l, m)-merge algorithm with l = m = K. Unshuffling will take one pass. Recursive mergings will take $T(K, K^{i-1}M)$ passes. Shuffling and cleaning the dirty sequence can be done in one more pass. Thus it follows that $T(K, K^iM) = T(K, K^{i-1}M)+2$. This means that

$T(K, K^iM) = 2i+T(K, M) = 2i+3$. ----- eq 2

Plug 2 into 1

$T(N/M, M) = \sum_{i=0}^{2C-1} (2i + 3) = 4C^2 + 4C$ ---- eq3

Case 2: $M/B < \sqrt{M}$; In this case we use l = m = M/B.
Let M/B = Q let $Q^d = N/M \Rightarrow d \log Q = \log(N/M) \Rightarrow$

$d = \log(N/M) / \log(M/B)$ --- eq 4

$T(Q^d, M) = T(Q, M) + T(Q, QM) + \ldots + T(Q, Q^{d-1}M)$ ----- eq 5

The **base case is: T(Q, M) = 3**

What is $T(Q, Q^iM)$, for any i? Similar to case 1, we can show that

$T(Q, Q^iM) = 1 + T(Q, Q^{i-1}M) + 1$
$= 2i + 3$ --- eq 6

Plug 6 into 5 to get:
$T(Q^d, M) = = \sum_{i=0}^{d-1} (2i + 3) = \frac{2(d-1)d}{2} + 3d$
$= d^2 - d + 3d = d^2 + 2d$ --- eq 7
Therefore # of passes $\leq \max(4C^2 + 4C + 1,\ d^2 + 2d + 1)$
$= \left(\frac{\log\left(\frac{N}{M}\right)}{\log\left[min\left(\sqrt{M}, \frac{M}{B}\right)\right]} + 1\right)^2. \ --- eq\ 8$

Example
Consider the case of $N = M^2$ and $B = \sqrt{M}$
Total number of passes = $\left(\frac{\log M}{\frac{\log M}{2}} + 1\right)^2$ = 9. Note that in your homework problem you showed that this sorting can be done in 7 passes. Thus eq 8 indeed presents an overcount on the number of passes taken by the (l, m)-merge sort algorithm.
Example
$N = M^3$ and $B = M^{2/3}$
Total number of passes
$\left(\frac{2\log M}{\frac{1}{3}\log(M)} + 1\right)^2 = 49.$

Rajasekaran and Sen 2004

We can sort n keys in $\tilde{O}\left(\frac{\log N/M}{\log\left(\frac{M}{B}\right)}\right)$ passes through the data.
Idea: randomly permute the input and apply D-way merge.
Lemma: If X is any sequence where each element is an integer picked uniformly randomly from [1, R] (R being arbitrary), we can sort X in O $\left(\frac{\log N/M}{\log\left(\frac{M}{B}\right)}\right)$ passes, with a high probability (this probability being on the space of all possible inputs).
Proof: From runs of length M each in one pass;
Use M/B- way merge to merge the N/M runs.
When BD element are ready in the output, write them in the disks;
When we run out of keys in any run, do one parallel I/O.
Assume that the memory size is CBD, for some constant C; When BD elements are ready in the output, the # of these elements coming from any run R is expected to B. Using Chernoff bounds, this # lies in the interval of $(1 \pm \epsilon)$ B, with a high probability.
This implies that the number of passes needed is $\tilde{O}\left(\frac{\log N/M}{\log\left(\frac{M}{B}\right)}\right)$.
To perform a random permutation: Assign a random label to each input key in the range $[1, N^{1+\beta}]$ for some constant $\beta < 1$. Sort the sequence based on the labels. Scan through the sorted sequence to permute equal keys.

Suffix Trees and Applications

Consider any string: $S = a_1 a_{2\ldots} a_m \in \Sigma^m$, $\Sigma \rightarrow$ Alphabet.
A suffix tree on S is

1) A directed rooted tree
2) There are m leaves one for each suffix of S,
3) Each none leaf has a degree of ≥ 2
4) Each leaf is labeled with an integer i, $1 \leq i \leq m$
5) Every edge has a label that is substring o S
6) The labels of no two edges going of any node can start with the same character
7) The ordered concatenation of edge labels starting from the root and ending in the leaf of i spells the suffix $a_i a_{i+1} \cdots a_m$.

Example: S = cdabacd
For this example we are not able to generate a suffix tree since cd is a suffix and there is another suffix that has cd as a prefix. To solve this problem we use a symbol not in the alphabet (denoted as $) at the end of the string.