CSE548: Analysis of Algorithms, Fall 2017 **Lecturer:** Rezaul A. Chowdhury
Topic: Master Theorem(General form) and Deterministic Selection Algorithm **Date:** Mon, Sep 25, 2017
Scribe: Aman Agarwal

8.1 Master Theorem

In the previous lecture, we read about Master Theorem (5.1.1) and its solutions for three different cases. Following are the three different cases.

$$T(n) = \begin{cases} \Theta(1), & \text{if n} \leq 1. \\ \\ aT\left(\dfrac{n}{b}\right) + f(n), & \text{otherwise (a}\geq 1\text{,b>1).} \end{cases} \tag{8.1.1}$$

Case 1: $f(n) = \mathcal{O}\left(n^{\log_b a - \epsilon}\right)$ for some constant ϵ>0

$$T(n) = \Theta(n^{\log_b a})$$

Case 2: $f(n) = \Theta(n^{\log_b a} lg^{k+1} n)$ for some constant $k \geq 0$

$$T(n) = \Theta(n^{\log_b a} lg^{k+1} n)$$

Case 3: $f(n) = \Omega(n^{\log_b a + \epsilon})$ and $af(\dfrac{n}{b}) \leq cf(n)$ for constants ϵ>0 and c<1

$$T(n) = \Theta(f(n))$$

The solution for these recurrences can be figured out very quickly if we compare $f(n)$ (cost of division and merging) to $n^{\log_b a}$ (which represents number of leaves or the amount of actual work).

- If number of leaves asymptotically/polynomially dominates $f(n)$, then the solution is number of leaves.
- If f(n) asymptotically/polynomially dominates number of leaves then solution is $f(n)$.
- If neither dominates polynomially and they are within polylog factor, $(\log n)^k$ where $k \geq 0$, then the running time is $f(n) \times \log^k n \times \log n$.

Using these and keeping in mind the recurrence relation (5.1.1), we will solve the applications of Master Theorem

8.1.1 Application of Master Theorem

Example 1: $T(n) = 3T(\frac{n}{2}) + \Theta(n)$
Here $a = 3$ and $b = 2$, so number of leaves is $n^{\log_2 3}$ which is polynomially larger than $f(n)$ i.e $\Theta(n)$.
Hence $T(n) = \Theta(n^{\log_2 3})$ (Master Theorem Case 1)

Example 2: $T(n) = 7T(\frac{n}{2}) + \Theta(n^2)$
Here $a = 7$ and $b = 2$, so number of leaves is $n^{\log_2 7}$ which asymptotically dominates $f(n)$ i.e $\Theta(n^2)$.
Hence $T(n) = \Theta(n^{\log_2 7})$ (Master Theorem Case 1)

Example 3: $T(n) = 2T(\frac{n}{2}) + \Theta(n)$
Here $a = 2$ and $b = 2$, so number of leaves is n which is within polylog factor of $f(n)$ i.e $\Theta(n)$.
Hence $T(n) = \Theta(n \log n)$ where $k = 0$. (Master Theorem Case 2)

Example 4: $T(n) = T(\frac{n}{2}) + \Theta(n^2)$
Here $a = 1$ and $b = 2$, so number of leaves is $n^{\log_2 1}$ which is asymptotically smaller than $f(n)$ i.e $\Theta(n^2)$.
Hence $T(n) = \Theta(n^2)$ (Master Theorem Case 3)

8.1.2 Recurrences not solvable using the Master Theorem

Master Theorem will work for many recurrences we find in textbooks but it will also not work for many recurrences we will encounter in life. Here are few examples.

Example 1: $T(n) = \sqrt{n}T(\frac{n}{2}) + n$
Here $a = \sqrt{n}$, which is not a constant and Master Theorem requires a to be a constant.

Example 2: $T(n) = 2T(\frac{n}{\log n}) + n^2$
Here $b = \log n$, which is not a constant and Master Theorem requires b to be a constant.

Example 3: $T(n) = \frac{1}{2}T(\frac{n}{2}) + n^2$
Here $a = \frac{1}{2}$, which is not ≥ 1 and Master Theorem requires a to be ≥ 1.

Example 4: $T(n) = 2T(\frac{4n}{3}) + n$
Here $b = \frac{3}{4}$, which is not > 1 and Master Theorem requires b to be > 1.

Example 5: $T(n) = 3T(\frac{n}{2}) - n$
Here $f(n) = -n$, which is negative and Master Theorem requires $f(n)$ to be positive.

Example 6: $T(n) = 2T(\frac{n}{2}) + n^2 \sin n$
Here $f(n) = n^2 \sin n$, which will be both positive and negative, hence violates the Master Theorem conditions.

Example 7: $T(n) = 2T(\frac{n}{2}) + \frac{n}{\log n}$
Here $f(n) = n(\log n)^{-1}$, so k is -1, but k should be ≥ 0, so we do not know how to solve this using Master Theorem.

Example 8: $T(n) = T(\frac{n}{2}) + 2T(\frac{n}{4}) + n$
Here a and b are not fixed and Master Theorem only works for single value of a and b.

8.2 General Form of Master Theorem

We will see how to write Master Theorem in more general form, which will help us understand Akra-Bazzi Recurrence better.

$$T(n) = n^{\log_b a} + \sum_{j=0}^{\log_b n - 1} a^j f(\frac{n}{b^j})$$

Here,

- $n^{\log_b a}$ is the number of leaves, and,
- The summation of cost of internal nodes (cost of division and merging) that goes from level 0 to level $\log_b n - 1$
- a^j is the number of subproblems at each level.
- $f(\frac{n}{b^j})$ is the cost of divide and merge for each subproblem

Lets assume,

$$p = \log_b a \tag{8.2.2}$$

$$n_j = \frac{n}{b^j} \tag{8.2.3}$$

So from equation 5.2.2, we have,

$$a = b^p$$

$$a^j = (b^j)^p$$

Replace b^j from equation 5.2.3,

$$a^j = (\frac{n}{n_j})^p \tag{8.2.4}$$

Substituting values of equation 5.2.2, 5.2.3 and 5.2.4, we get

$$T(n) = n^p + \sum_{j=0}^{\log_b n-1} (\frac{n}{n_j})^p f(n_j)$$

Lets concentrate only on the summation for now, here j represents the different levels and j gives us $\log_b n$ different values for n_j. Intuitively, in a line of length n, we are choosing $\log n$ different values.

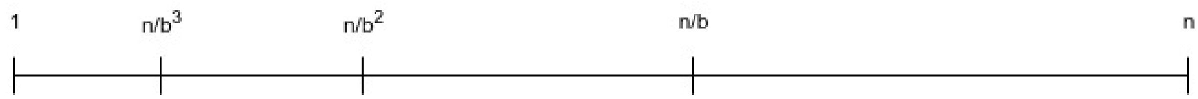

Figure 8.2.1: Reference figure

We know, in the divide and conquer tree, the input size(n_j) at the top level is n, and then it drops at each level by a factor of $\frac{1}{b}$.
Thus, when $j = 0$, $n_j = n$ (Since $n_j = \frac{n}{b^j}$)
when $j = 1$, $n_j = \frac{n}{b}$
when $j = 2$, $n_j = \frac{n}{b^2}$
when $j = \log_b n - 1$, $n_j = b$
Also, lets assume, $m = n_j$
Substituting these values we get,

$$T(n) = n^p + \sum_{m \epsilon \{b, b^2, .., \frac{n}{b^2}, \frac{n}{b}, n\}} (\frac{n}{m})^p f(m)$$

$$T(n) = n^p + n^p \times \sum_{m \epsilon \{b, b^2, .., \frac{n}{b^2}, \frac{n}{b}, n\}} (\frac{f(m)}{m^p}) \tag{8.2.5}$$

Now, we might have points at irregular intervals in the line and then summing over all those points may be difficult. So to generalize we will sum over all the points, from 1 to n, we may be over computing in this case but we can fix that. Lets assume for any point m, such that $\frac{n}{b} < m \leq n$, $f(m)$ reduces by a constant

factor of $f(n)$ i.e $f(m) = \Theta(f(n))$. Similarly for other points, say $\frac{n}{b^2} < m \leq \frac{n}{b}$, $f(m) = \Theta(f(n/b))$ and so on. This is true when $f(n)$ is of the form $f(n) = n^{\alpha} \log^{\beta} n$ where $\alpha, \beta > 0$, which is mostly true for the case we generally deal with. Thus overall, $f(m) = \Theta(f(n))$.
For $m = n$ (considering only the summation part),

$$\sum_{m \epsilon \{b, b^2, .., \frac{n}{b^2}, \frac{n}{b}, n\}} (\frac{f(m)}{m^p}) = \sum_{m=n} (\frac{f(m)}{m^p})$$

Taking all the points between $\frac{n}{b}$ and n, we can say

$$= \sum_{m=\frac{n}{b}+1}^{n} (\frac{f(m)}{m^p})$$

Since $f(m) = \Theta(f(n))$,

$$= \Theta(\sum_{m=\frac{n}{b}+1}^{n} (\frac{f(n)}{m^p}))$$

Since m in within constant factor of n

$$= \Theta(\sum_{m=\frac{n}{b}+1}^{n} (\frac{f(n)}{n^p}))$$

$$= \Theta((n - \frac{n}{b} - 1 + 1) \frac{f(n)}{n^p})$$

$$= \Theta((n(1 - \frac{1}{b}) \frac{f(n)}{n^p})$$

So we have

$$\sum_{m=\frac{n}{b}+1}^{n} (\frac{f(m)}{m^p}) = \Theta(n \times \frac{f(n)}{n^p})$$

which is equal to

$$\sum_{m=\frac{n}{b}+1}^{n} (\frac{f(m)}{m^p}) = \Theta(n \times \sum_{m=n} (\frac{f(m)}{m^p}))$$

which can we written as

$$\sum_{m=n} (\frac{f(m)}{m^p}) = \Theta(\frac{1}{n} \sum_{m=\frac{n}{b}+1}^{n} (\frac{f(m)}{m^p}))$$

Since $m = \Theta(n)$, taking $\frac{1}{n} inside$

$$\sum_{m=n} (\frac{f(m)}{m^p}) = \Theta(\sum_{m=\frac{n}{b}+1}^{n} (\frac{f(m)}{m^{p+1}}))$$

Substituting this in equation 5.2.5, we get

$$T(n) = \Theta(n^p + n^p \sum_{m=1}^{n} \frac{f(m)}{m^{p+1}})$$

Hence,

$$T(n) = \Theta(n^p(1 + \sum_{m=1}^{n} \frac{f(m)}{m^{p+1}}))$$

From the above equation we can say, extending $\log n$ points along the line to the entire line gives us something which is within the constant factor of the original sum. This is the more general form of Master Theorem than the original form and will be useful when we have complicated recurrences with multiple $a's$ and $b's$.

8.3 Akra-Bazzi Recurrences

8.3.1 Sample recurrence (with multiple a and b)

$$T(n) = a_1 T(\frac{n}{b_1}) + a_2 T(\frac{n}{b_2}) + a_3 T(\frac{n}{b_3})$$

Lets assume, $T(n) = n^p$

$$T(n) = a_1 (\frac{n}{b_1})^p + a_2 (\frac{n}{b_2})^p + a_3 (\frac{n}{b_3})^p$$

$$T(n) = [\frac{a_1}{b_1^p} + \frac{a_2}{b_2^p} + \frac{a_3}{b_3^p}] \times n^p$$

If for some p, $[\frac{a_1}{b_1^p} + \frac{a_2}{b_2^p} + \frac{a_3}{b_3^p}] = 1$, then our solution is n^p. This is simple and straightforward as we don't have $f(n)$ here, when $f(n)$ is present we use the general form of master theorem.

8.3.2 Deterministic Select(Selection Algorithm)

Input: An array of n numbers and an integer k
Output: Find k^{th} smallest number
Simplest solution: Sorting, but sorting finds 1^{st} smallest, 2^{nd} smallest and so on. But we are suppose to find only one number, k^{th} smallest, that extra work is not required.

Steps to solve it-

1. If we have $n \leq 140$, then we can sort it directly, because sorting a constant number of n is $\Theta(1)$
2. Else we will divide n into groups of 5, i.e we will have $\left\lceil \frac{n}{5} \right\rceil$ groups.
3. Now we will take medians of every group i.e $\left\lceil \frac{n}{5} \right\rceil$ medians.
4. Then we will find the median of the above medians, using the same algorithm recursively, say x.
5. Now we will scan over the array and partition it into two groups. One group will have elements smaller than x and another group with elements greater than x.
6. Now if our k is equal to $n_1 + 1$, then we have found out item (x). Else if k is smaller than $n_1 + 1$ then k is in the left partition and we can forget about the right partition and vice versa.
7. Similarly we will recurse down either of the partition and find the element.

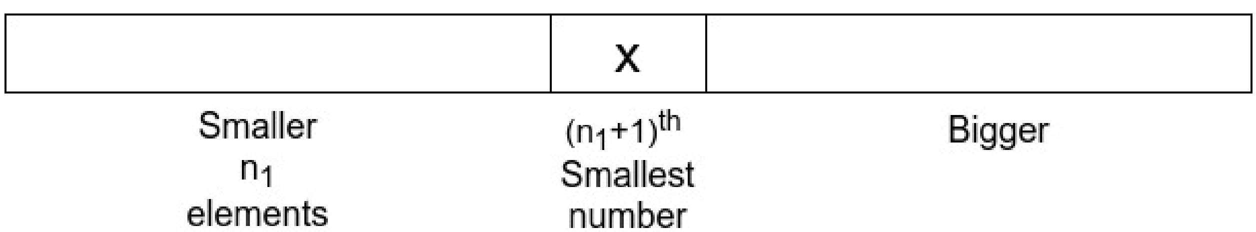

Figure 8.3.2: Reference figure

```
DETERMINISTICSELECT(A[q : r], k)

    n ← r − q + 1
    if n ≤ 140 then
        sort A[q : r] and return A[q + k − 1]
    else
        divide A[q : r] into blocks B'_i s each containing 5 consecutive elements    {last block may contain fewer than 5 elements}
        for i ← 1 to ⌈n/5⌉ do
            M[i] ← median of B_i using sorting
        x ← DETERMINISTICSELECT (M[1 : ⌈n/5⌉], ⌊(⌈n/5⌉ + 1)/2⌋)    {median of medians}
        t ← PARTITION (A[q : r], x)    {partition around x which ends up at A[t]}
        if k = t - q + 1 then return A[t]
        else if k ¡ t - q +1 then return DETERMINISTICSELECT (A[q : t − 1], k)
            else return DETERMINISTICSELECT (A[t + 1 : r], k − t + q − 1)

DETERMINISTICSELECT ENDS
```

Figure 8.3.3: Pseudocode for Deterministic Selection Algorithm

Why more efficient than quick sort: Here we are choosing x very carefully, in such a way that both parts will have at least 30% of the original items. So we will never recurse down a subproblem that is larger than $\frac{7n}{10}$ (We will see more on that below). So we will never end up with a situation where running time is $\Theta(n^2)$.

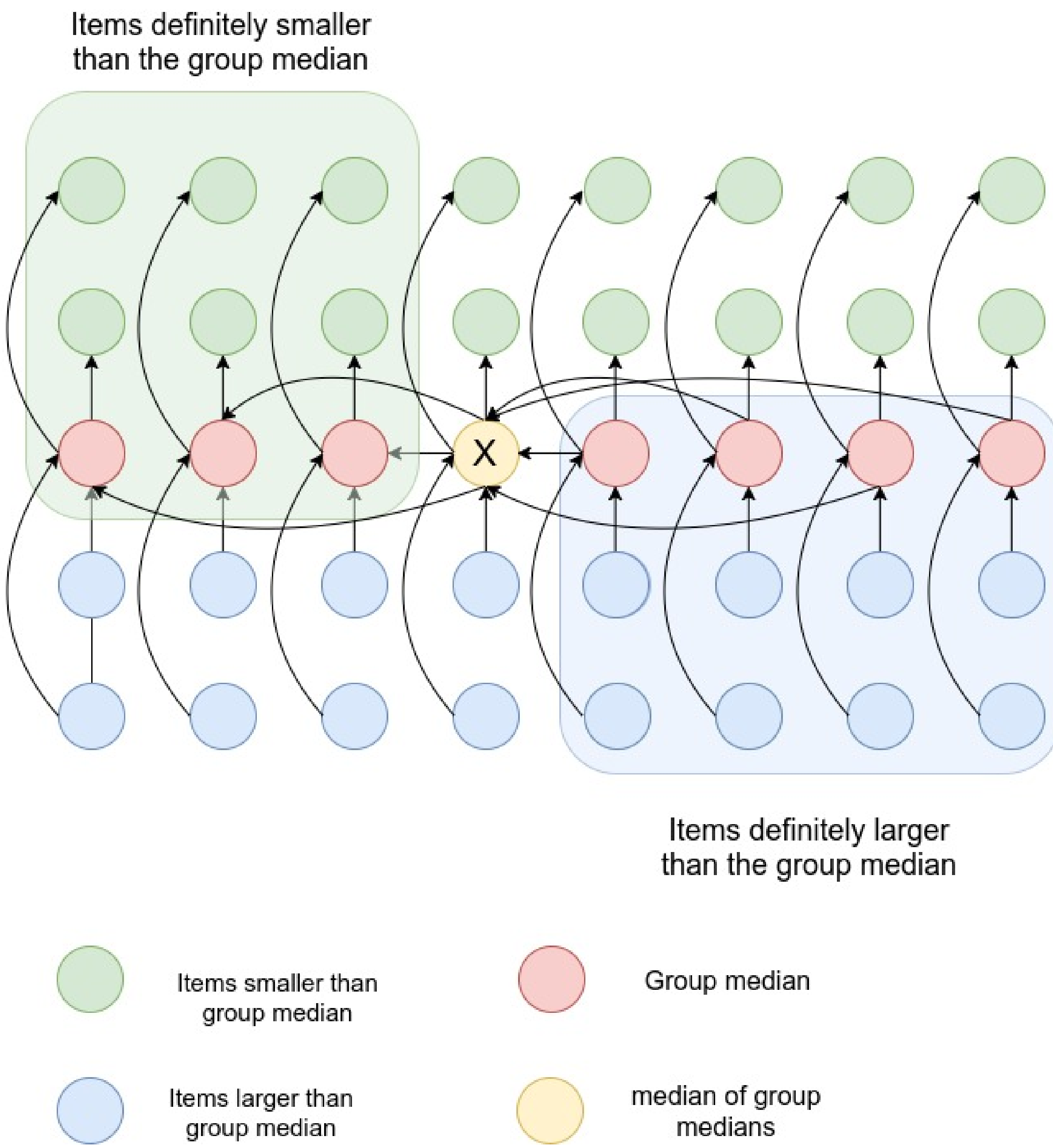

Figure 8.3.4: Model showing selection algorithm

From the above model, we can observe that x is the median of medians and almost half of the group is on the either side of x. Intuitively we can say either of the boxes(green and blue) are at least 25% of the total. Now when we count them, we find

- Items definitely smaller than x is $\geq \dfrac{3n}{10} - 6$ and

- Items definitely larger than x is $\geq \dfrac{3n}{10} - 6$.
- Maximum number of items on the left of x cannot be more than $n - (\dfrac{3n}{10} - 6)$, which is $\dfrac{7n}{10} + 6$.
- Similarly, right side of x cannot have more than $\dfrac{7n}{10} + 6$ items. This means neither left side nor right side can have more than 70% of the items.

Now we can write the recurrence for the algorithm.

$$T(n) \leq \begin{cases} \Theta(1), & \text{if n<140.} \\ \\ T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T(\frac{7n}{10} + 6) + \Theta(n), & \text{if n} \geq 140. \end{cases} \tag{8.3.6}$$

Here,

- If n<140, we sort the input, which is $\Theta(1)$. Else,
- Cost of grouping and finding the median is $\Theta(n)$ as it requires scanning over all the items.
- Cost of finding the median of medians, using the same algorithm is $T(\left\lceil \frac{n}{5} \right\rceil)$
- Cost of partitioning and recursing down one side is $T(\dfrac{7n}{10} + 6)$ as either side cannot have more than $\dfrac{7n}{10} + 6$ items.

Now we will simplify the recurrence. We will remove the ceiling for simplicity and simplify $\dfrac{7n}{10} + 6$ and for that we will find a ratio which is simpler and is as close to $\dfrac{7n}{10} + 6$.
For sufficiently large n, the following will be true.

$$\begin{gathered} \frac{7n}{10} + 6 \leq \frac{8n}{10} \\ 7n + 60 \leq 8n \\ n \geq 60 \end{gathered}$$

And we know $n \geq 140$, so the simplified recurrence is-

$$T'(n) \leq \begin{cases} \Theta(1), & \text{if n<140.} \\ \\ T'(\frac{n}{5}) + T'(\frac{4n}{5}) + \Theta(n), & \text{if n} \geq 140. \end{cases} \tag{8.3.7}$$

Now, we have to be careful while simplifying so that we don't change the terms in such a way that changes the solution of the original problem asymptotically. We know $\frac{8n}{10}$ is larger than $\frac{7n}{10} + 6$, for some large n. Maybe this is too large, and in that case the simplified recurrence will give an bound larger than the original bound. So, we must try to find recurrences that are very close.
For example if we take,

$$\frac{7n}{10} + 6 \leq \frac{7.5n}{10}$$
$$n \geq 120$$

which gives a more tighter bound.