1. Merge Sort and Analysis of Recurrences

2. Loop Invariants

Merge Sort:

Sort the first and second halves of the array recursively.

Merge these two sorted lists to obtain the sorted order of all the elements.

Merge(A[1,2,...,n],B[1,2,...m]):

Given two sorted lists of size m &n;

Merge outputs A union B in sorted order using at most (m+n) comparisons.

A	**2**	**5**	**6**	**10**	**14**		
B	**1**	**3**	**4**	**11**	**15**	**18**	20
C							

Merge(A[1,2,...,n],B[1,2,...m]):

```
i=1, j=1, k=0
While ( i<n and j<m)
    If (A[i]<B[j]) set C[k]=A[i] and increment i
    Else Set C[k]=B[j] and increment j
If (i=n) Copy remaining elements of B to C.
If (j=m) Copy remaining elements of A to C.
Output C.
```

Merge Sort:

Sort the first and second halves of the array recursively.
Merge these two sorted lists to obtain the sorted order of all the elements.

$$T(n)=2T(n/2)+O(n)$$

Analysis:

$T(n)=2T(n/2)+cn$

$=cn+2(2T(n/4)+cn/2))$

$=cn+cn+4T(n/4)$

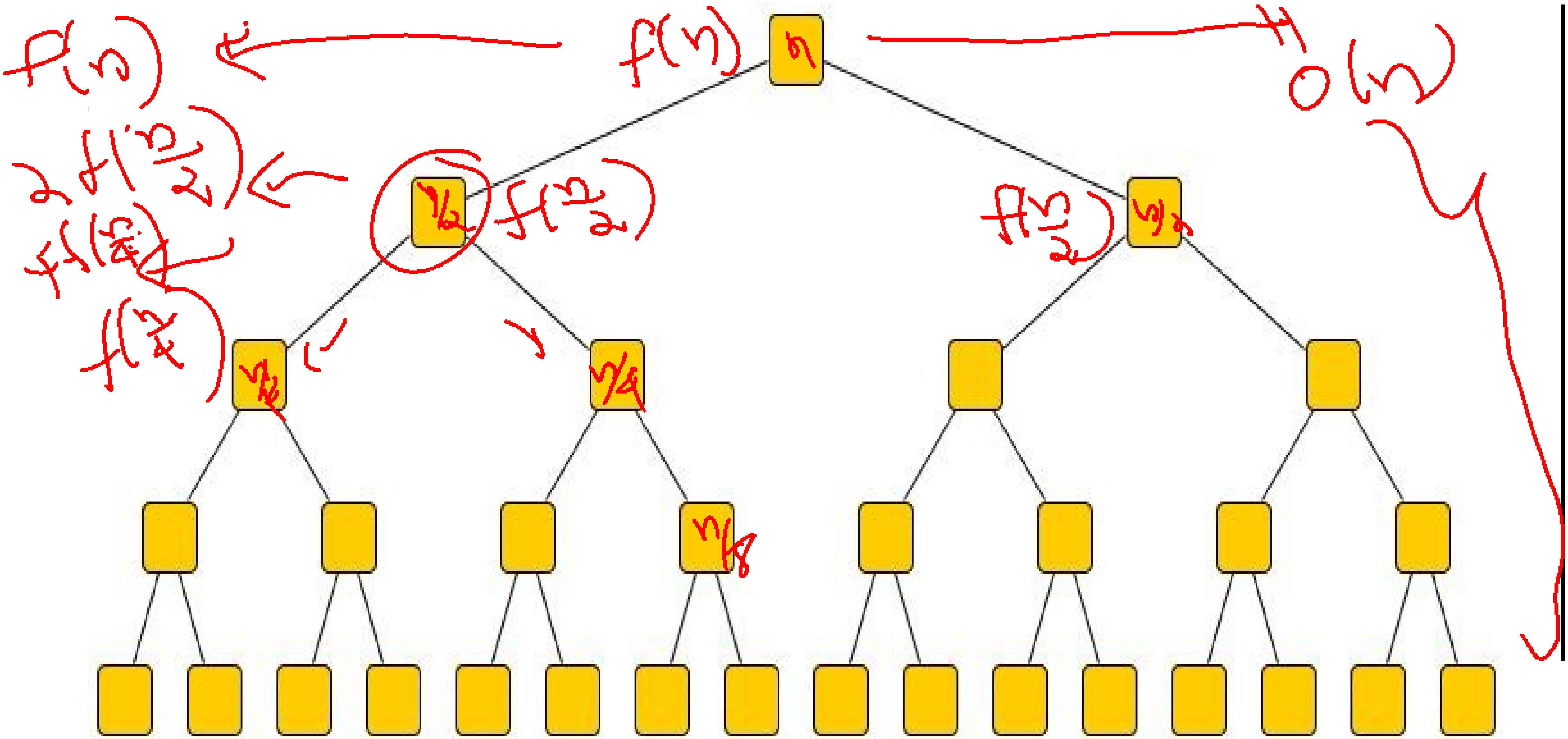
$=cn+cn+cn+8T(n/8)$

$$= \underbrace{cn+cn+\ldots+cn}_{k \text{ times}}$$

$n = 2^k$

$k = \log_2 n$

$$= cn\log_2 n = O(n\log n)$$



f(n)
2f(n/2)
f(n/4)
f(n)
n
n/2
f(n/2)
f(n/2)
n/2
n/4
n/4
n/8
O(n)

Instructions at level 1: f(n)
Instructions at level 2: 2f(n/2)
Instructions at level 3: 4f(n/4)

$f(n) = cn$

$T(n)=f(n)+2f(n/2)+4f(n/4)+\cdots+2^k f(n/2^k)$

$n = 2^k$

$= cn + 2(c\frac{n}{2}) + 4(c\frac{n}{4}) + \cdots$

$= cn\log_2 n$

Ex 1: T(n)=2T(n/2)+sqrt(n)

Ex 2: T(n)=8T(n/2)

$= 8\times 8\, T(n/4) = 8^3 T(n/8)$

$T(n/2) = 8\, T(n/4)$

$$\boxed{1 + r + \dots + r^n = O(r^n)} \quad \frac{r^{n+1} - 1}{r - 1}$$

$k = \log n$

$$T(n) = \sqrt{n} + 2\left(\sqrt{\frac{n}{2}}\right) + 4\left(\sqrt{\frac{n}{4}}\right) + \dots$$

$$\frac{r^{n+1} - 1}{r - 1} = O(r^n)$$

$$= \sqrt{n}\left(1 + \sqrt{2} + (\sqrt{2})^2 + \dots + (\sqrt{2})^k\right)$$

$$n^{\log_2 2}$$

$$= \sqrt{n}\, O\left((\sqrt{2})^k\right)$$

$$= \sqrt{n} \cdot O(\sqrt{n}) = O(n)$$

$$T(n) = 8T\left(\frac{n}{2}\right) + O(1)$$

$$8\,t\left(\frac{n}{2}\right)$$

$$8^2 t\left(\frac{n}{4}\right)$$

$$= 8\cdot 8T\left(\frac{n}{4}\right) = 8^2 T\left(\frac{n}{2^2}\right)$$

$$= 8^3 T\left(\frac{n}{8}\right) = 8^3 T\left(\frac{n}{2^3}\right)$$

$$\vdots$$

$$= 8^k T\left(\frac{n}{2^k}\right) \qquad \frac{n}{2^k} = 1$$

$$\log_2 n$$

$$= 8^{\log_2 n} = n^{\log_2 8} = n^3$$

$$a^{\log_b c} = b^{\log_c a}$$

$$= \log_c b \cdot \frac{\log a}{\log b} \cdot \log_c a$$

Method 2: Master Theorem

$T(n)=aT(n/b)+f(n)$ a,b>1

Compare $n^{(\log_b a)}$ and f(n). If one of them is larger, that's the solution.

If both are equal, the solution is $T(n)=O(f(n)\log n)$.

$f(n)$

$f(n)$

$\vdots$

$f(n)$

$n^{\log_b a}$

$T(n) = aT(\frac{n}{b})$

$= a^{\log_b n}$

$= n^{\log_b a}$

$f(n)(1+\gamma+\gamma^2+\dots) = \frac{1}{1-\gamma}$

$\gamma < 1$

$2T(\frac{n}{2}) + O(n)$

$n^{\log_2 2} = n$

Master Theorem E.g. 1

$T(n)=3T(n/2)+5n$

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$

$\Theta\left(n^{\log_2 3}\right)$ vs $\Theta(n)$

$\log_2 3 \approx 1.58$

$3^{\log_2 n} = n^{\log_2 3}$

$T(n) = \Theta\left(n^{\log_2 3}\right)$

Master Theorem E.g. 2

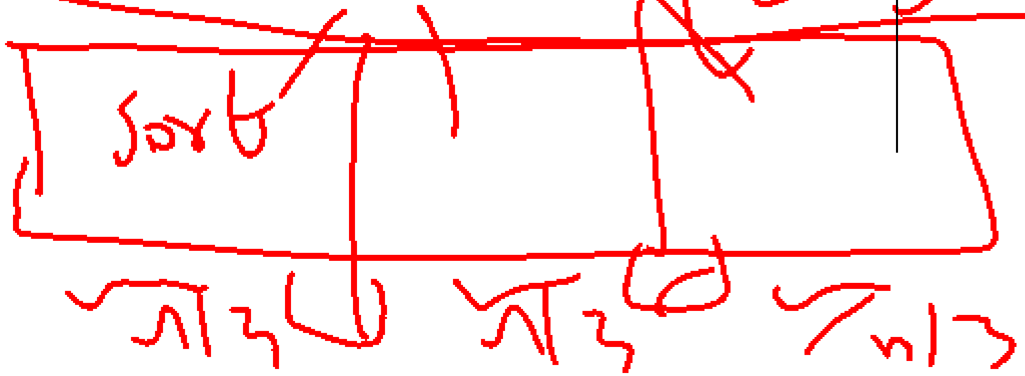
$T(n)=4T(n/2)+n^2$

$n^{\log_2 4} = n^2$

$T(n) = O(n^2 \log n)$

(1) $T(n) = 3T\left(\frac{n}{5}\right) + O(\sqrt{n})$

(2) $T(n) = 3T\left(\frac{n}{3}\right) + O(n)$



$n^{\log_5 3} \gg n^{1/2}$

$n^{\log_5 3}$ vs n

$\log_5 3$ vs $\frac{1}{2}$

$3 > 5^{1/2}$

$\log_2 n = O(\log_3 n)$

$\log_3 n = O(\log_2 n)$

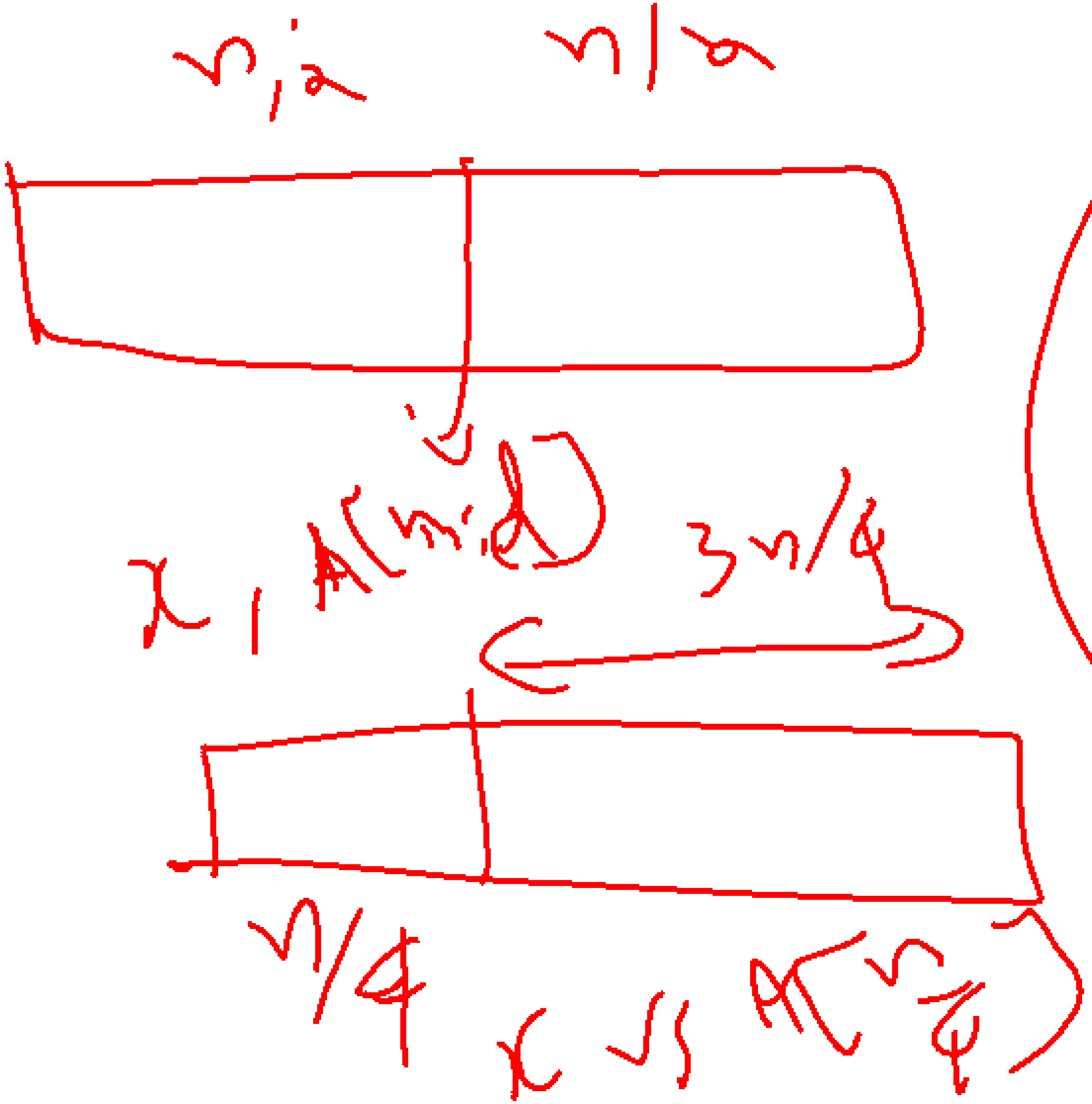
$= O(\log n)$

$\log_3 3$

n vs n

$n = n$

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



Skewed binary search

Sorting:

Input: A[1,2,…,n]

Output: B[1,2…,n]:

$$B[1]<=B[2]<=\ldots=<B[n]$$

and

$$\{B[1],B[2],\ldots,B[n]\}=\{A[1],A[2],\ldots,A[n]\}$$

5	2	4	6	1	3

5	2	4	6	1	3

5	2	4	6	1	3

5	2	4	6	1	3

5	2	4	6	1	3

Insertion Sort

```
For j=2 to n
    key=A[j]
    //Insert A[j] into the sorted sequence A[1,…,j-1]
    i=j-1
    while (i>0 and A[i]>key)
        A[i+1]=A[i]
        i=i-1
    A[i+1]=key
```

Insertion Sort: Loop Invariants

```
For j=2 to n
    key=A[j]
    //Insert A[j] into the sorted sequence A[1,...,j-1]
    i=j-1
    while (i>0 and A[i]>key)
        A[i+1]=A[i]
        i=i-1
    A[i+1]=key
```