# Lecture 2

# Lecture 2

## LAST Time

Basic code review - Runtime

Ex1

```
for (i = 1 to n){
    print hello
    i = i + 1;
}
```

O(n)

EX2:

```
for (i=1 to n){
    i = i * 3;
    print hello
}
```

$i = 1, 3, 9, \ldots \quad \boxed{3^i = n}$

$3^i = n$

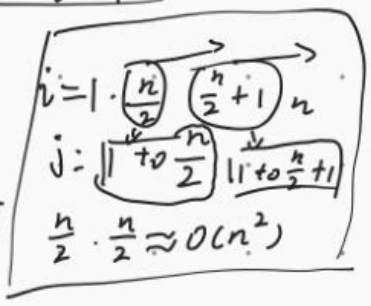
$\log_3 3^i = \log_3 n$

$\boxed{j = \log_3 n}$

EX3:

```
for (i=1 to n){
    for (j=1 to i){
        print hello;
    }
}
```

O(n) · O(n)




$\sum_{i=1}^{n} \sum_{j=1}^{i} 1 = \sum_{j=1}^{n} j = \frac{n(n+1)}{2} = O(n^2)$

$\frac{n^2+n}{2}$

$\sum_{j=1}^{3}$

EX4:

```
For (i=1 to n)
    if (i=20){
        for (j=1 to n^3){        O(n^3)
            print hello
        }
    }
}
```

O(n)

$\Rightarrow n+n^3 = O(n^3)$

$n=7 \quad c=2$

$g(n)=n^2$

$T(n)=O(f(n))$ $\qquad 1^n$

The function $T(n)$ on input is less that $c(n)$ for some c AND as long as $n \geq n_0$

---

Def:

$T(n) = O(f(n))$ if $\exists c, n_0$ s.t.

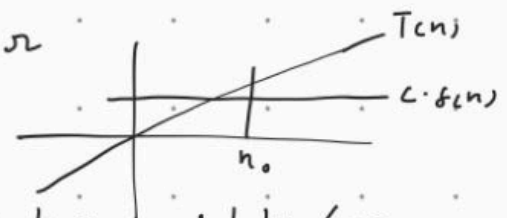
$\forall n \geq n_0 \quad T(n) \leq c \cdot f(n)$

Runtime of my code $5n^2+3n+8 = O(n^2)$.

Big $\Omega$

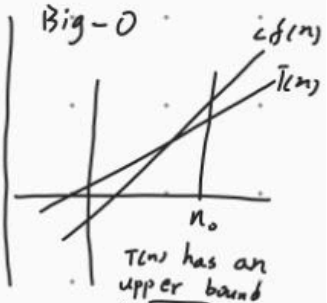
$T(n)$ is $\Omega(f(n))$ if more exist $c, n_0$ $[c>0, n_0 \geq 0]$ such that

$\forall n \geq n_0 \quad T(n) \geq c \cdot f(c(n))$




T(n) is lower Bounded by f(n)

Big-O




T(n) has an upper bound of f(n)

Ex5

$5n^3 + 70n + 100 = \Omega(n^3)$

$\geq \quad \geq \quad \geq \quad \geq cn^3$

$5n^3 + 0 + 0$ ∃ $c=5$ $n_0=0$

$5n^3 + 70n - 100 = \Omega(n^3)$

$\geq \qquad\qquad \geq cn^3$

$5n^3 + 69n + n - 100$ $\quad n \geq 100$

$\geq \quad \geq \quad \geq$

$5n^3 + 0 + 0$

$\geq 5n^3$ $\quad c=5$ $\quad n_0 = 100$

---

Big-θ

$T(n) = \theta(g(n))$ iff $T(n) = O(g(n))$ AND $T(n) = \Omega(g(n))$

---

$T(n) = 5n^3 + 7$ $\boxed{T(n) = \theta(n^3)}$ ← prove

step 1 $T(n) = O(n^3)$

$5n^3 + 7 \leq cn^3$

$\leq$

$5n^3 + 7n^3 \leq 12n^3$ $\boxed{c=12}$ $n_0 = 1$

step 2 $T(n) = \Omega(n^3)$

$5n^3 + 7 \geq cn^3$ $\quad \boxed{\exists \; c=5 \quad n_0=0}$

$\geq \quad \geq$

$5n^3 + 0 \geq 5n^3$

---

Limmit Lemma

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = L$

if $L=0 \Rightarrow f(n) = O(g(n))$

$L=\infty \Rightarrow f(n) = \Omega(g(n))$

$L$ = constant Not zero $\Rightarrow f(n) = \theta(g(n))$

$T(n) = 5n^3 - 100n^2 \qquad f(n) = n^3$

prove $T(n) = \Omega(n^3)$

$$\lim_{n\to\infty} \frac{5n^3 - 100n^2}{n^3} = \frac{15n^2 - 200n}{3n^2} = \lim_{n\to\infty} \frac{30n - 200}{6n} = \frac{30}{6} = 5$$

$$\lim \frac{\infty}{\infty} \quad \lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

$T(n) = \Theta(n^3) \Rightarrow T(n) = \Omega(n^3)$ by def of Theta.

---

$\log(n) \quad n^{.000001}$

$$\lim_{n\to\infty} \frac{\log(n)}{n^{.00001}}$$

$\to \infty \quad \log(n)$

$\to 0 \quad n^{.0001}$ larger

$$\frac{\frac{1}{n}}{.00001\, n^{0.00001-1}} = \frac{\frac{1}{n^1} \cdot [n^{-.00001+1}]}{.00001}$$

$$= \frac{n^{-.00001}}{.0001} = \frac{1}{.00001 n^{.00001}} = 0$$

$$\log(n) = O(n^x) \qquad x > 0$$

---

Divide & conquer given a problem

Divide: The problem into subprusems that smaller instances of the same problem.

int calprofit Reveme (array cost, array Revenue)

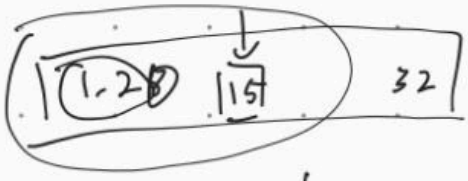
int X = ADDup cost(cost[])

conquer: solve the smaller instance recursively (calling the same function)

if [instance is small enough] = basecase

solve Diceery.

combine: combine the sol. TO smaller instances into a final answer.




7 nlogn

Assume you have an array S such that |S| = n

Find the smallest int in S.

```
for(i=1 to |S|)
        if (smallest > A[i])
        {
        }
Return smallest
```

O(n)

| 50 | 100 | 60 |
|---|---|---|

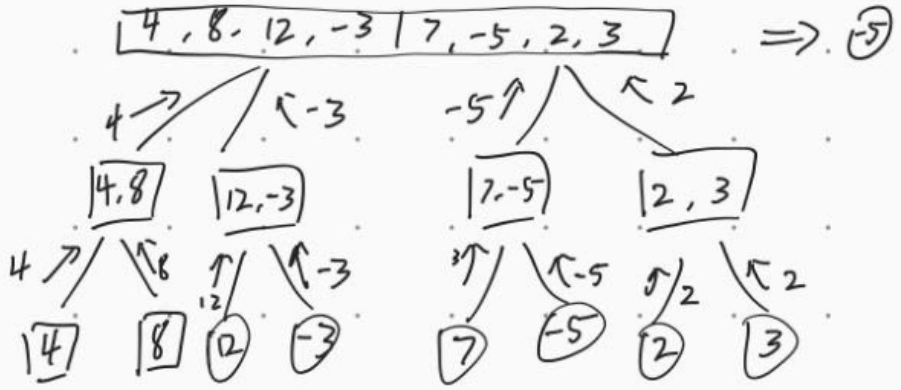
```
int smallest(A[..]){
    if (n==1) Rerven A[i];
    X = smallest(A[1..n/2])
    Y = smallest(A[n/2(1...n])
    if (X<Y)
        return X
    else
        return Y.
}
```

T(n) = Runtime of my code on inract size n

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

---

3 methods of solving Recurrence

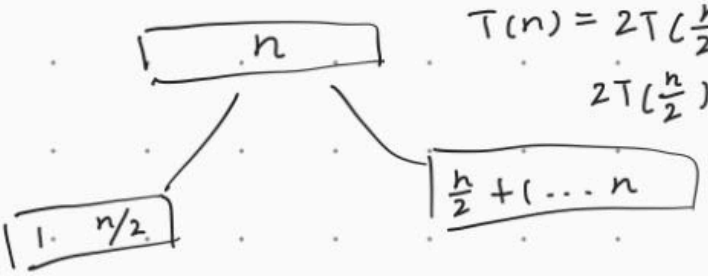



$$T(n) = 2T(\frac{n}{2}) + O(n)$$

nlog(n)

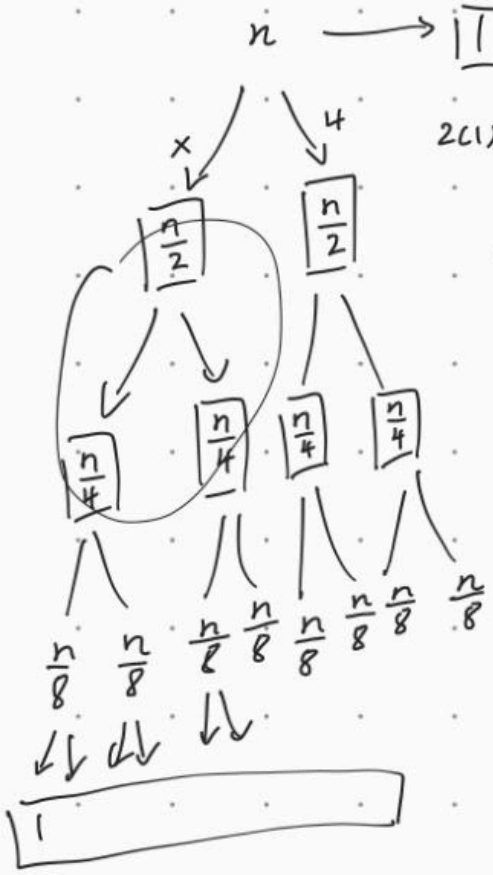
methods

1. currence tree . method
2. sub stitation
3. m . T.




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

$2T(\frac{n}{2}) + 1$




work = Runtime

$2(1)$

$\Sigma$ work per level levels

$2(1)$

$4(1)$

$\Sigma$ work per level depth

$8(1)$

work per level $2^i(1)$


$\frac{n}{2^0} = n$

$\frac{n}{2^i} = 1$ when I stop

$n = 2^i$

$\log_2 n = \log_2 2^i \qquad i = \log_2 n$

$$\sum_{i=0}^{\log_2 n} 2^i(1) = \sum_{i=0}^{\log_2 n} 2^i = 2^{\log_2 n + 1} - 1$$

$$= 2(2^{\log_2 n}) - 1$$

$$2n - 1 = O(n)$$

$$\sum_{i=0}^{x} 2^i = \frac{2^{x+1} - 1}{1}$$

---