Monday Dec. 10

Exam Review I

Solving Recurrence Relation

$n = 4$

$$\begin{cases} T(1) = 1 \\ T(n) = 2 \cdot T(n-1) + 1 \end{cases}$$

$n-2$ $\quad$ $n+1$ $\quad$ $n-1$

unfold

$T(n) = 2 \cdot T(n-1) + 1$

$= 2 \cdot (2 \cdot \boxed{T(n-2) + 1}) + 1$

$= 2 \cdot (2 \cdot (2 \cdot T(n-3) + 1) + 1) + 1$

$= \cdots$

$\underbrace{2 \cdot (2 \cdot (\cdots}_{2^{n-1}} T(1)) \underbrace{(+1) + 1) + 1}_{n-1}$

$= n - \cancel{1}(n-1)$

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

is $O(2^n)$

$$\begin{cases} T(0) = 1 \\ T(1) = 1 \\ T(n) = \underbrace{2 \cdot T\left(\frac{n}{2}\right)}_{\text{sort } L \text{ and } R} + \underbrace{n}_{\text{merge}} + \underbrace{1}_{\text{split}} \end{cases}$$

<u>assume</u> $n = 2^i \quad i > 0$

$n \cdot \log n$

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

$$= 2 \cdot \left(2 \cdot \boxed{T\left(\frac{n}{4}\right)} + \frac{n}{2} + 1\right) + n + 1$$

$n/2^1$ $\quad n/2^2$

$\log n$: how many #'s of unfoldings?

n :

$T(1)$ $\quad \boxed{\frac{n}{2^{\log n}}}$

Given a recursive method.

```
void m( ____ ) {
    if (..) { ____ };
    else if (..) { ____ };
    else { m( ____ );
}
```

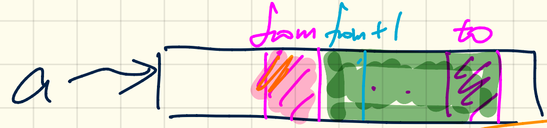
Prove by Math Induction:

① Prove base cases

② State inductive hypothesis (I.H.) on problem size n

③ Argue about the correctness of problem size $(n+1)$

↳ apply the I.H. to conclude the proof.




```
int sum(int[] a){
    return sumH(a, 0, a.length-1);
}

int sumH(int[] a, int from, int to){
    if(from > to){ return 0;}
    else if (from == to){ return a[from]; }
    else { return a[from] + sumH(a, from+1, to); }
}
```

$$\sum_{i=x}^{y} a(i) = a(x) + \sum_{i=x+1}^{y} a(i)$$

int sumH(int[] a, int from, int to){

1 if (from > to) { return 0; }

2 else if (from == to) { return ~~a[to]~~ a[from]; }

3 else { a[from] + sumH(a, from+1, to); }

}

return

line numbers

Proof. (Principle link from math to code)

Base Case 1. for empty array, we know $\sum_{i=x}^{x-1} a[i] = 0$, Line 1 does this.

Recursive Case. State the following Inductive Hypothesis:

sumH(a, from+1, to) return $\sum_{i=from+1}^{to} a[i]$

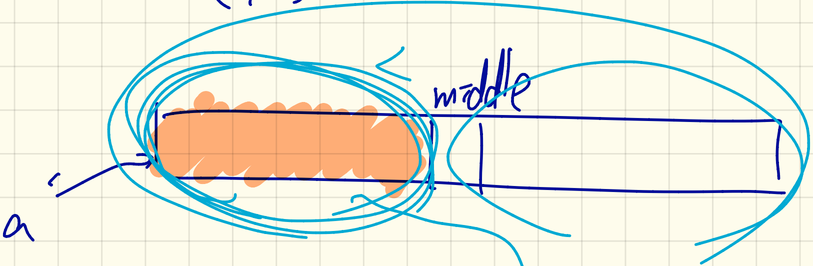
sub problem.

To compute $\sum_{i=from}^{to} a[i]$, we do: $a[from] + \sum_{i=from+1}^{to} a[i]$

original problem

↳ this is done at Line 3 with I.H.

search (k)



k < a[middle]

→ binS(a, from, middle-1, k)

1, 1, 2, 3, 5, . —

```
int[] fibArray(int n) {
    int[] result = new int[n];
    fibArrayH(n, 0, result);
    return result;
}
```

fibA(0) → e.a.

fibA(1) → [1]

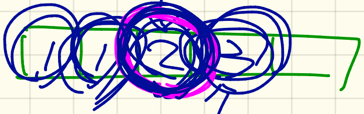
fibA(2) → [1 | 1]

fibA(3) → [1 | 1 | 2]

fib

```
void fibArrayH(int n, int from, int[] result) {
    result[from] = result[from-1] + result[from-2];
    fibArrayH(n, from+1, result);
}
```

1st recursive call fills this in



2nd recursive call fills in this value

```
/* assumption : a > 0 , b > 0 */
int gcd(int a, int b) {
    if (a == 0 || b == 0) {
        if (a > b) { return a; }
        else { return b; }
    }
    else {
        if (a > b) { return gcd(a%b, b); }
        else { return gcd(a, b%a); }
    }
}
```

gcd(x, y) =

return a > b ? a : b;

2

$gcd(46, 14) = gcd(46 \% 14, 14) = gcd(4, 2)$

$= gcd(4 \% 2, 2)$

4 → a, 14 → b, $4 \% 2 = 0$