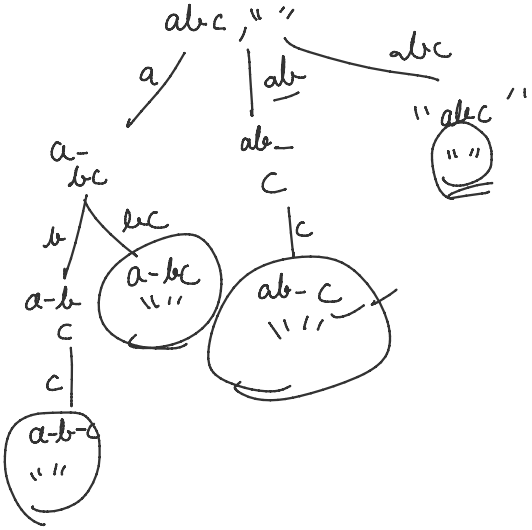
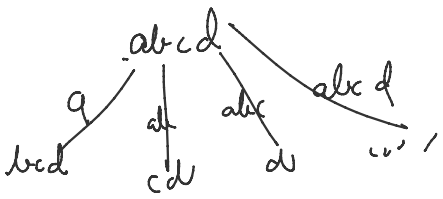
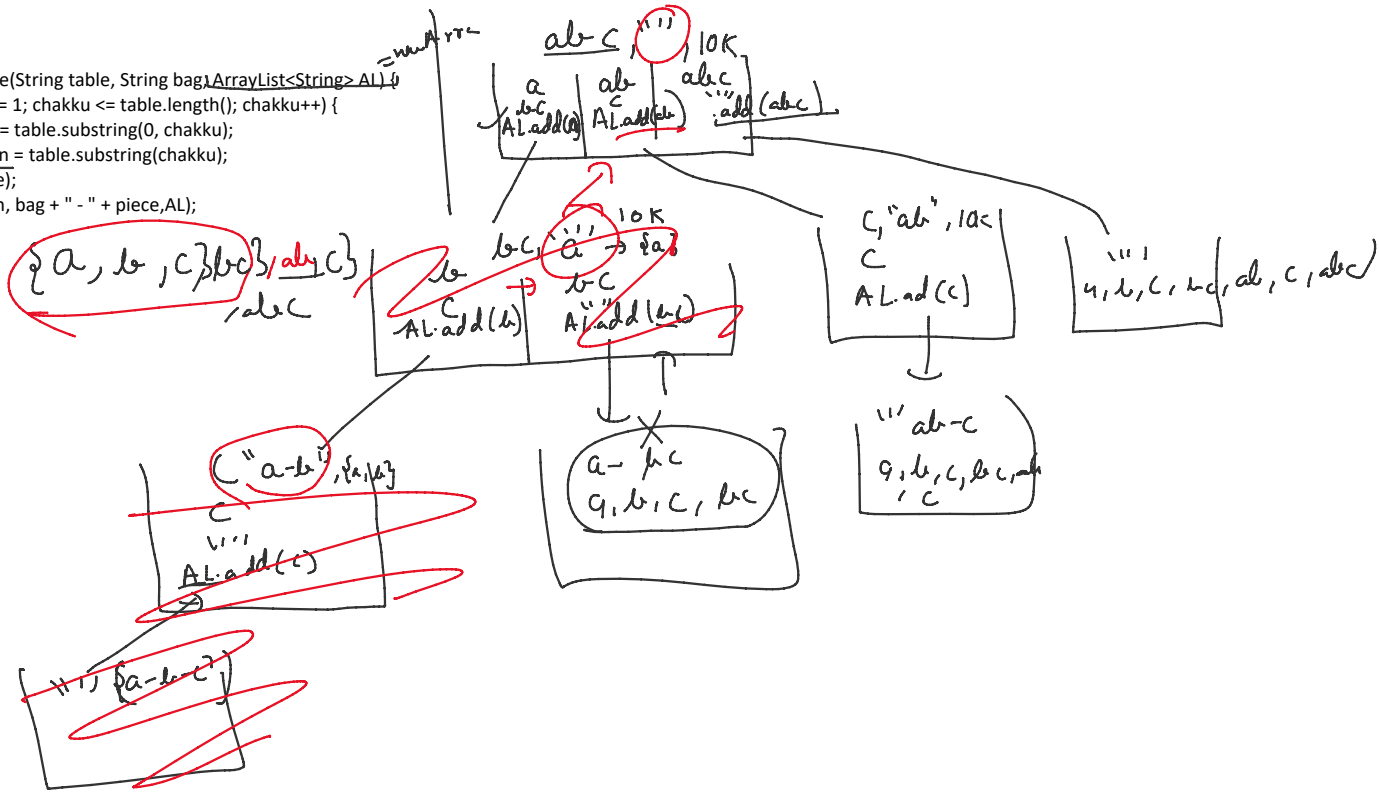
26 March 2023 16:33

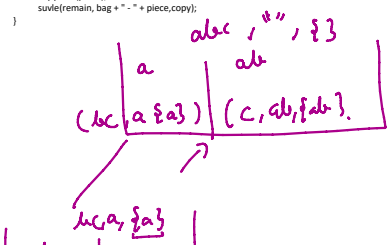


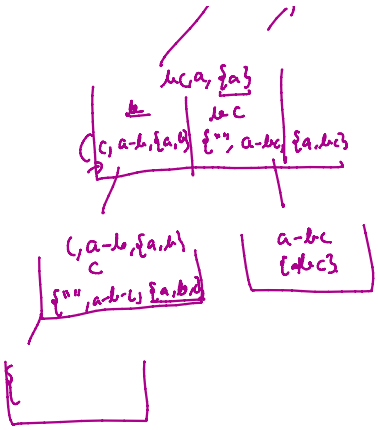


```
public static void suvle(String table, String bag,ArrayList<String> AL) {
        for (int chakku = 1; chakku <= table.length(); chakku++) {
            String piece = table.substring(0, chakku);
            String remain = table.substring(chakku);
            AL.add(piece);
            suvle(remain, bag + " - " + piece,AL);
        }
    }
```

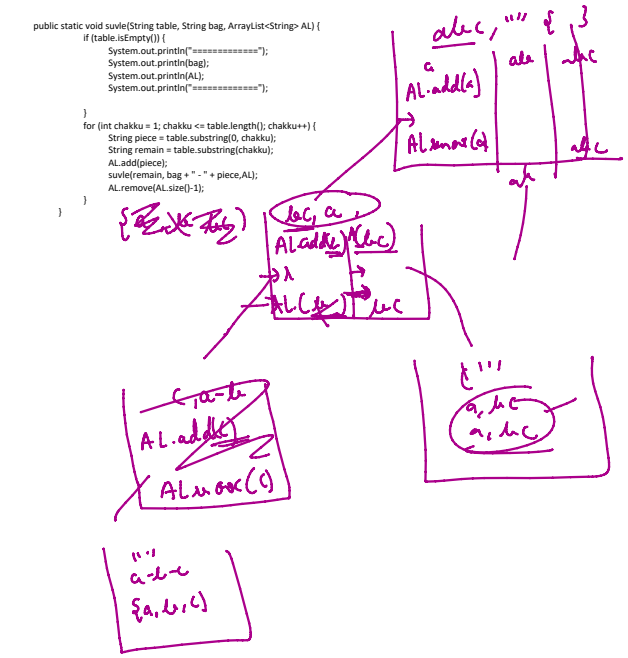


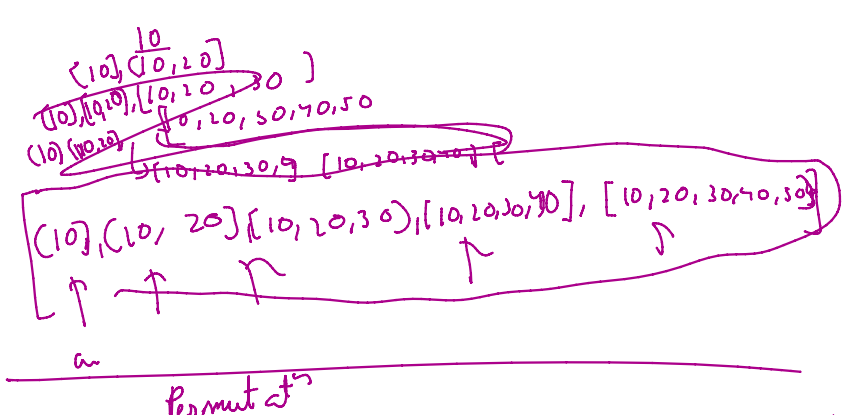
```
for (int chakku = 1; chakku <= table.length(); chakku++) {
                String piece = table.substring(0, chakku);
                String remain = table.substring(chakku);
//                  AL.add(piece);
                ArrayList<String> copy = new ArrayList<>(AL);
                copy.add(piece);
                suvle(remain, bag + " - " + piece,copy);
            }
```

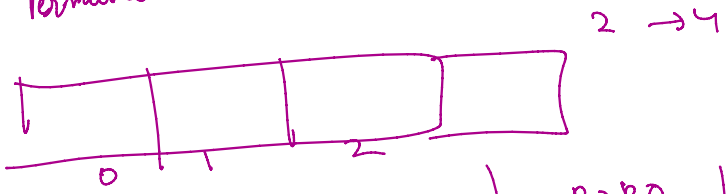




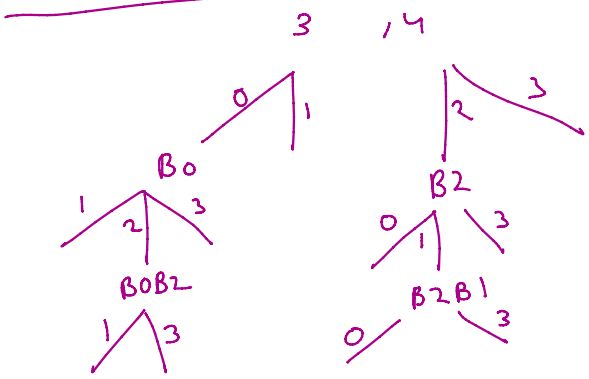
```
public static void suvle(String table, String bag, ArrayList<String> AL) {
        if (table.isEmpty()) {
            System.out.println("=============");
            System.out.println(bag);
            System.out.println(AL);
            System.out.println("=============");

        }
        for (int chakku = 1; chakku <= table.length(); chakku++) {
            String piece = table.substring(0, chakku);
            String remain = table.substring(chakku);
            AL.add(piece);
            suvle(remain, bag + " - " + piece,AL);
            AL.remove(AL.size()-1);
        }
    }
```



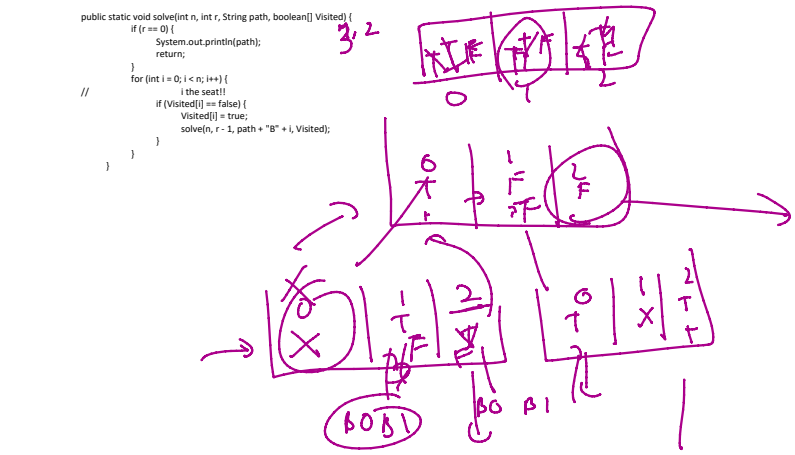


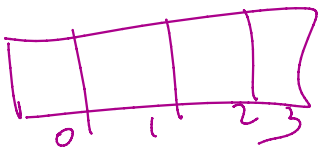


B0 B1 | B1 B2 | B2 B1 | B3 B1
B0 B2 | B1 B3 | B2 B3 | B3 B2
B0 B3

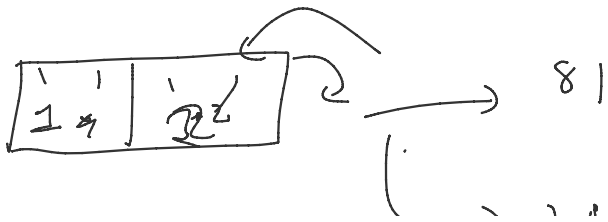


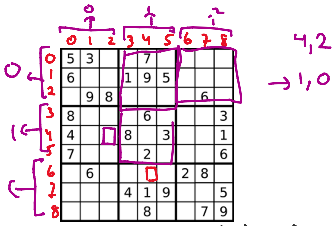

```
public static void solve(int n, int r, String path, boolean[] Visited) {
        if (r == 0) {
                System.out.println(path);
                return;
        }
        for (int i = 0; i < n; i++) {
//                      i the seat!!
                if (Visited[i] == false) {
                        Visited[i] = true;
                        solve(n, r - 1, path + "B" + i, Visited);
                }
        }
}
```

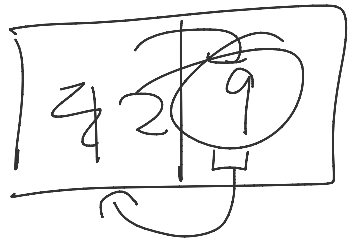




0 1 | 1 2 | 2 3
0 2 | 1 3 |
0 3 |







$f(n) = f(n-1) + 1$

M1) substitute

M2)

∀ f^n: work in each f^n frame

Σ operation in each f^n frame

fact(n) {
sp = (n-1)!
ret n·sp
}

(a^b)
$sp = a^{b-1}$
return $sp \cdot a$

a^b
$sp = a^{b/2}$
return $sp \cdot sp$
→ $\log(b)$

Recurrence relationship

f(n) {
ret f(n-1) + f(n-2)
}

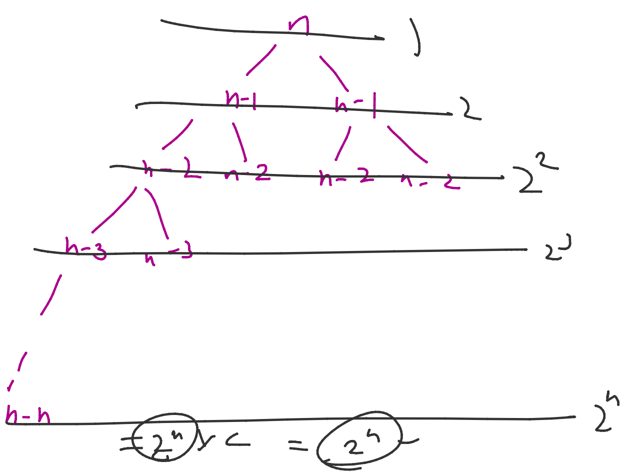
$f(n) = f(n-1) + 1$
$f(n-1) = f(n-2) + 1$
$f(n-2) = f(n-3) + 1$
$f(n-(n-1)) = f(0) + 1$

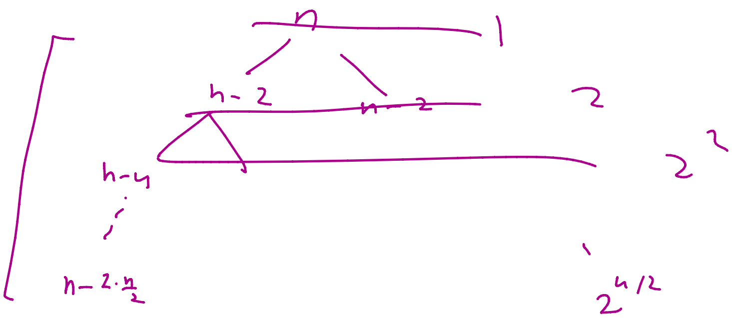
$f(n) = n$

$f(b) = f(b/2) + 1$
$f(b/2) = f(b/2^2) + 1$
$f(b/2^2) = f(b/2^3) + 1$
K
$f\left(\frac{b}{2^K}\right) = f\left(\frac{b}{2^{K+1}}\right) + 1$

$b/2^K = 1$
$b = 2^K$
$K = \log_2 b$

$f(b) = 1 + 1 + \dots = K = \log b$





$n - 2 \cdot \frac{n}{2}$

$2^{n/2}$

$2^{n/2} \cdot k$

2^n se $2^{n/2}$

$f(n) = f(n-1) + f(n-2)$ +1

$f(n-2) \leq f(n-1)$

$f(n) = 2f(n-1) + 1$

$2 \cdot f(n-1) = 2^2 f(n-2) + 1 \cdot 2$

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

$2^3 f(n-3) = 2^4 f(n-4) + 1 \cdot 2^3$

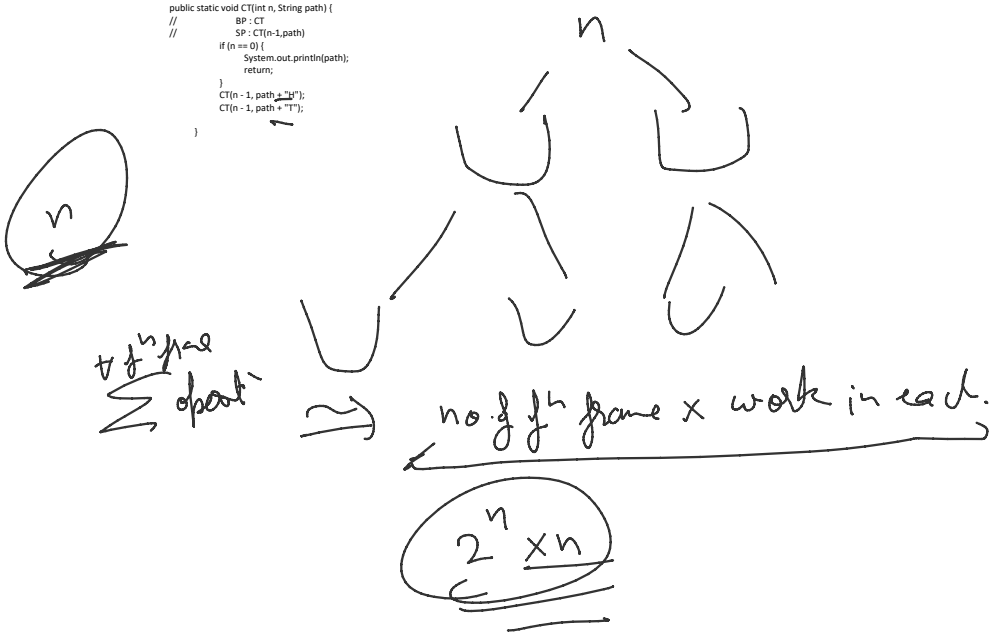
$2^k f(n-k) = 2^{k+1} f(n-(k+1)) + 1 \cdot 2^k$

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

$f(n) = 1 + 2 + 2^2 + 2^3 + \ldots 2^k \ldots 2^{n-1}$

$= 2^n$

```
public static void CT(int n, String path) {
//                BP : CT
//                SP : CT(n-1,path)
        if (n == 0) {
                System.out.println(path);
                return;
        }
        CT(n - 1, path + "H");
        CT(n - 1, path + "T");
}
```



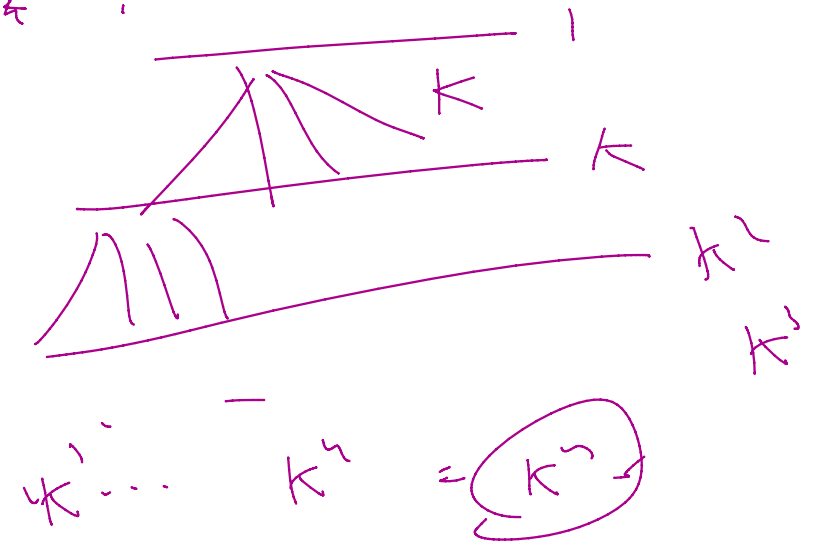
# of nodes at $\sum$ depth $\rightsquigarrow$ no. of f^n frame $\times$ work in each.

$2^n \times n$

$f(n) = 2f(n-1) + n$

$2f(n-1) = 2f(n-2) + (n-1)$

$n \cdot 2^n$



$K^1 \cdots K^n = K^n$

$f(n) = Kf(n-1) + nK$

$nK \cdot K^n$

nK^n

nK^n

$n \to n^n$

$K \to$

$8 \cdot 4^8$

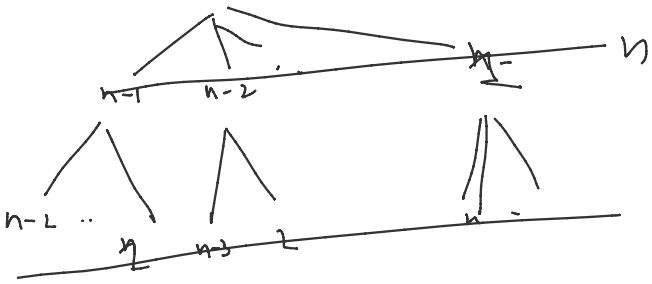
n

$f(n) = f(n-1) + f(n-2) + f(n-3) \cdots f(1) + n$

$f(n-1) = f(n-2) + f(n-3) + \cdots f(1) + 1$

$f(n) = 2f(n-1) + 1 \to 2^n$

$n - 1$



$n \cdot (n-1)(n-2)$

Subtype

$n! + n + n(n-1) + n(n-1)(n-2)$

$n!$

Array sort recursive

page sort

B

f^n ( )