Note Title 15-11-2018

Iterative ← int fact (int n){
result = 1
for (int i = n; i >= 1; i--) {
result = result * i;
}

COL100
Last Class

Recursion
Tail Recursion → Tail Recursive!

int fact(int n){
if(n==0)
return 1;
else
return n*fact(n-1);
}

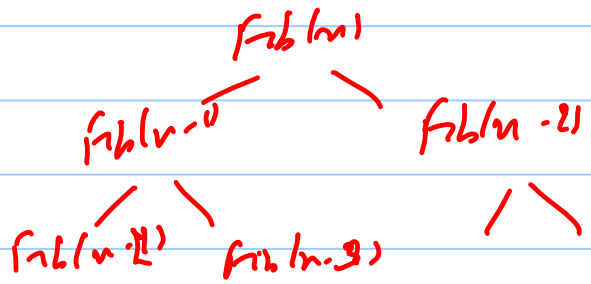
int fact_h(int n, int result){
if(n == 1)
return result;
return fact_h(n-1, n*result);
}

solution O(n) 0, 1, 1, 2, 3, 5, 8

fib(n) 0th, 1st, ... Fibonacci

```
int fib(n) {
    if((n==0) || (n==1)) {
        return 1;
    }
    return fib(n-1) + fib(n-1);
}
```

$\sim O(2^n)$ exponential



Write a Tail recursive program for Fibonacci

```
int fib(int n) {
    int prev1 = 0;
    int prev2 = 1;
    for(int i=2; i<=n; i++) {
        res = prev1 + prev2;
        prev1 = prev2;
        prev2 = res;
    }
    return res;
}
```

0th → 0, 1, 1, 2, (3)

int fib_h(int n, int prev1, int prev2) {

Base case if(n == 1)
return prev2;

return fib_h(n-1, prev2, prev1+prev2);

}

O(n) tail recursive program

int fib(int n) { if(n==0) return 0;
fib_h(n, 0, 1);

}

Write the equivalent iterative program for two Tail recursive Fib functions.

```
int fib_h(int n, int i, int prev2, int prev1) {
    if(n == i)
        return prev2;
    return fib_h(n, i+1, prev2, prev1+prev2);
}
```

"invariant"
prev2 = fib(i)
prev1 = fib(i-1)

Tail recursive program

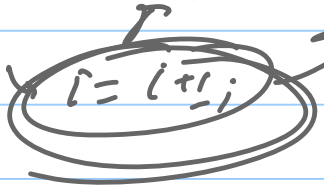
O(n) Array

```
int fib(int n) {
    if(n==0)
        return 1;
    fib_h(n, 1, 0, 1);
}
```

i ↑ prev2

Any finitable loop can be converted into a Tail Recursive program

// while (cond) → Break step recursion



Argument

Language which do not use loops
↳ use Tail recursion to implement loops.

Exhaustive search & Backtracking

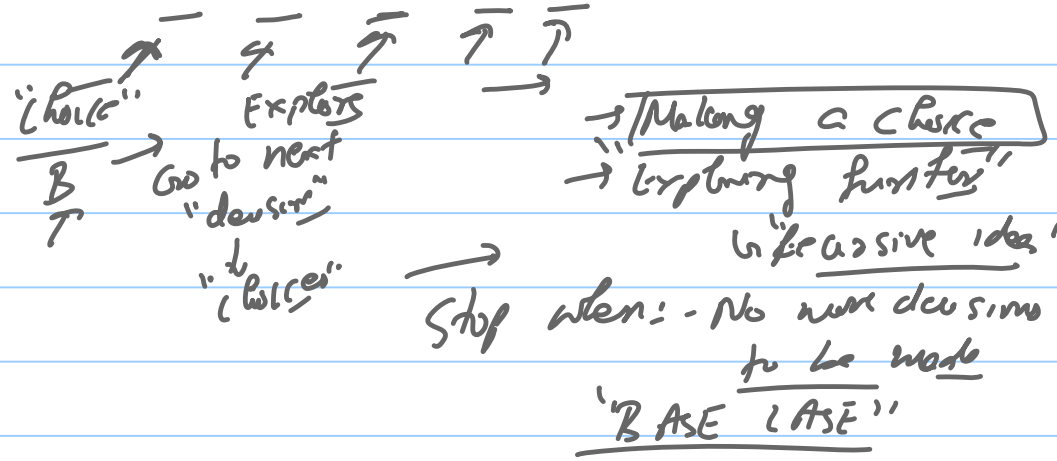
Example:-

3 letter words

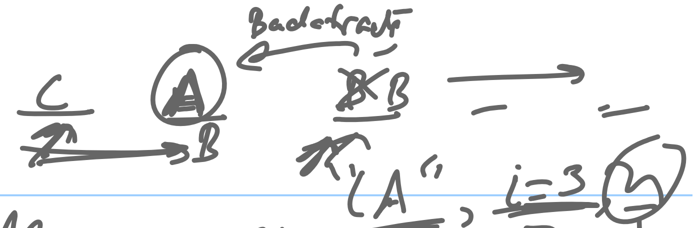
A, B, ... Z

26

26^3



"Wrapper function"



Making choices -

① Iterate over all possibilities. — Exploration
(Go over) — current result

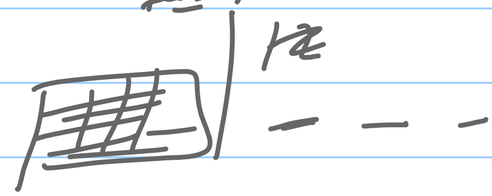
& choose one after another

② Exploration
↳ recursive calls.

"modify"
"unmodify"

"Current result" state & Exploration done so far.

→ ① solving puzzles
② playing chess.
③ Tic Tac Toe.



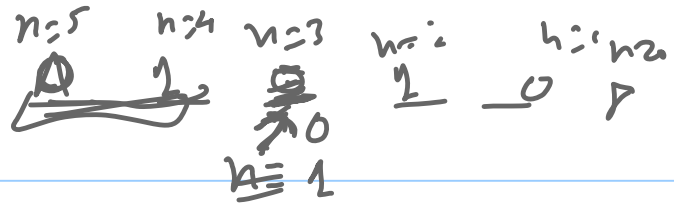
n = 6

0 1 0 1 1 1

void printAllBinary_h( ){

}

$O(2^n)$




```
void printAllBinary(int n) {
    printAllBinaryRec(n, "");
}

void printAllBinaryRec(int n, string soFar) {
    if (n == 0) {
        cout << soFar << endl;
        return;
    }
    printAllBinaryRec(n-1, soFar + "0");
    printAllBinaryRec(n-1, soFar + "1");
}
```

Base case

Recursive calls

Explanation:- Recursively

void printAllDecimal(int n) {

printAllDecimal(n, ___);

}

printAllDecimalH(int n, string soFar) {

if(n == 0) {

cout << soFar << endl;

return;

}

for(int i = 0; i <= 9; i++) {

printAllDecimalH(n-1, soFar + integerToString(i));

}

}

$O(10^n)$

recursive choices

current "recursive"

n = 0 ... n = 8

Recursion & Enumeration

Backtracking