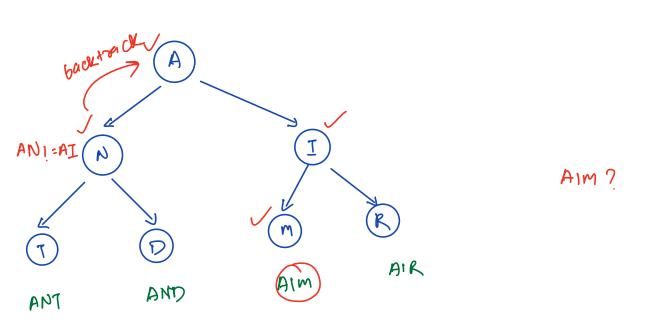
Backtracking 1



Suestion

Course an integer A. Write a function to generate

all combinations of well-formed parantherm of

length 2*A.

well-formed monor of open = total no. of clok

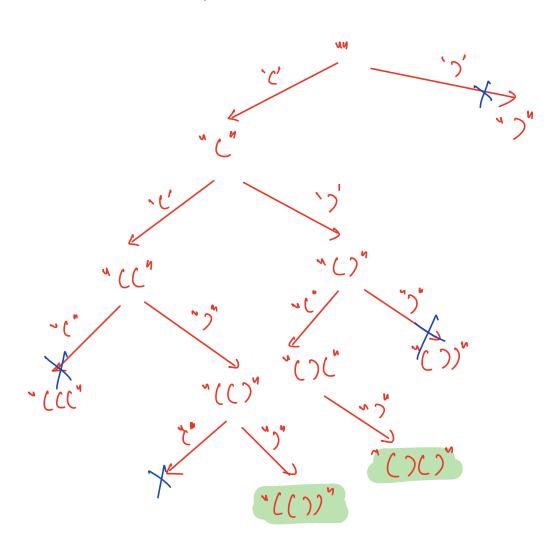
fotal no. of open = total no. of open >= no. of

if you go from left to right, no. of open >= no. of

clex

A = 3

"((()())", "(()())", "()())", "()())"



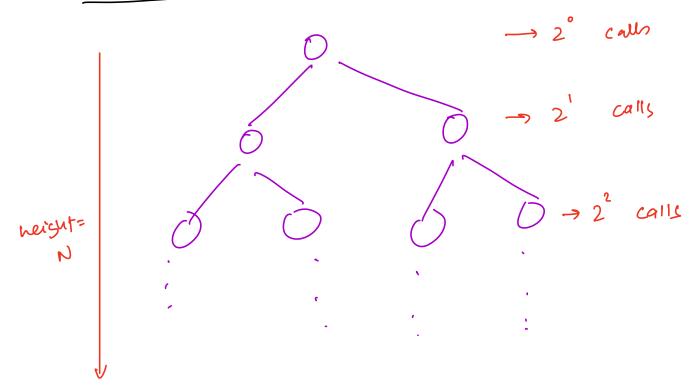
solve (" , D, 0,0)

Code

def some (str, N, open, close) } if (open = = N lt U0& = = N) } private str) if (open <N) 3 solve (str+"C", N, open+1, close); 3

```
if ( close < open ) & // open >= close
               Solve ( Str + ")", N, open, close +1);
          3
    open=1,110x-0 "C"
                          "()" open:1, close=1
open=2,close="(("
                               open=2
open=2,40421"(C)"
                        "()()"
                                 open = 2
                                   clos=2
```

time complexity



$$2^{\circ} + 2^{'} + 2^{2} + \cdots + 2^{N-1} = 2^{\circ} \left(\frac{2^{N} - 1}{2 - 1}\right) = 2^{N} - 1$$

total
$$TC = O(2^N)$$
 N=10 N=20 $\approx 10^6$

$$SC = O(N)$$

Subset > any possible combination of original array

A = [1 2 3 4 5]

 subset = [1 4],
 ORDER

 doesn't
 MATTER

 [3 1]
 MATTER

Subsequence of array elements after deleting

A = [1 2 3 4 5]

subsequence: [14],

[125],

order matters

Subarray > continous part of array

Every Subarray is a subsequence but not opposite Every subsequence is a subset but not opposite

count = 8

Sustion

Ciner an arrang of distinct etements.

Retim all subsets using recursion.

A=[1 2 3]

ofp: 33

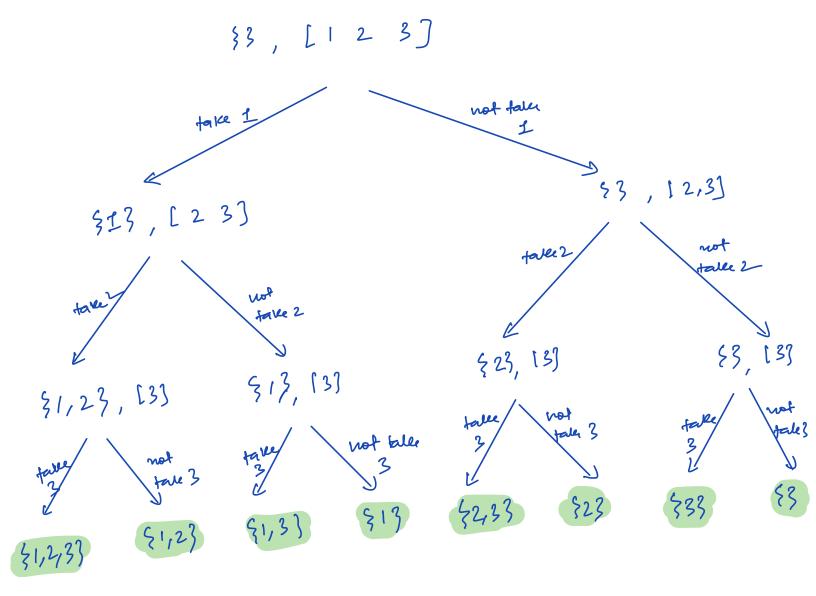
313

3123

31 2 33

923

32 33



```
I sofom all embecks
           > we need array of arrays.
list < list < int>) aun; - global variable
                                            subcets (A,0, [])
     subsets ( int A1), inti, list ( int ) worset) &
void
      if ( i == A. sizel)) &
            am. push ( wroset)
            schm
      11 two options
       1. Pick AU) in subset
          currect add (AUI)
          subsels (A, i+1, wrrset);
       2. don't pick Au)
           wrrset. Jemone Back(); -> property of
                                           back tracking
           subsits (A, i+1, worset);
```

```
subsets (int AT), inti, list (int) correct) &

if (i == A-size()) &

ano. push (urrect)

3 return

currect add (AUI)

subsets (A, i+1, correct);

currect. semone Back();

subsets (A, i+1, correct);
```

```
A=[269]
vaid subsets ( int A1), int i, list < int > correct) &
     if (i == A. sizel) 3
          am. push ( wroset)
          sch3M
     current add (Acil) [2]
     Subsels ('A, i+1, wrrset);
     (wrrset. semone Back();
                                                                       if ( i == A. sizel)) &
    (Subsal (A, i+1, worset);
                                                                            am. push ( wroset)
 3
                                                                       3 schon
                                                                       currect add (AUI)
                                                                       subsen(A, i+1, wrrset);
                                                                       wriset. semone Back();
     subsets ( int AT), inti, list (int) correct) &
                                                                      subsect (A, i+1, worset);
    if (i == A. sizel)) {
          am. push ( wroset)
      3 schom
    current add (AUI) [2,6)
     Subsch ('A, i+1, wrrset);
     (wrrset. semone Back(); (2)
    subsus (A, i+1, wreset);
 3
                                                                   subschs (int AT), inti, list (int) worset) &
     Subsects ( int At), inti, list(int) correct) q
                                                                   if ( i == A. sizel)) }
    4 (i == A. sizel)) }
                                                                        aus. push ( wroset)
          am. push ( wroset)
                                                                       જામા
         જામજા
                                                                    current add (AUI)
     current add (Aül) [269]
                                                                   Subsch ('A, i+1, wrrset);
    Subsch (A, i+1, wrrset);
                                                                   wrrset. semone Back();
     wrrset. semone Back();
                               [26]
                                                                  Subsch (A, i+1, worset);
   (Subsiti (A, i+1, wriset);
                                                                           i=3,[29] (2:3,[2]
                                       C2 69)
                                                              subsets ( int A1), int i, list (int) worset) &
     subschol int Al), inti, list(int) correct) q
     if ( i == A. sizel) }
                                                               if ( i == A. size()) }
         am. push ( wroset)
                                                                    am. push ( worset)
      3 schm
                                                               3 schom
      current add (AUI)
                                                               currect add (AUI)
      Subsencia, in, worset);
                                                               Subsch (A, it, wrrset);
     wrrset. semone Back();
                                                               wrrset. semone Back();
     subsets (A, i+1, worset);
                                                              subsets (A, i+1, worset);
```

ans. [2,6,9), 12,67, 12,97, 127, [6,97, [67, [9], [7]]

Time complexity
$$\rightarrow O(2^N)$$

 $SC = O(N)$

fitbit wants to arrange workours in different order.

ABCD ACBD DABC

Circu a string witer distinct characters, print all permutations.

abc
$$\rightarrow$$
 abc bac cab (3! =6) acb bca cba

string of size N -> N! promutations

$$[a--], [a,b,c]$$

$$[a--], [b,c]$$

$$[b--], [a,c]$$

$$[a--], [a,b]$$

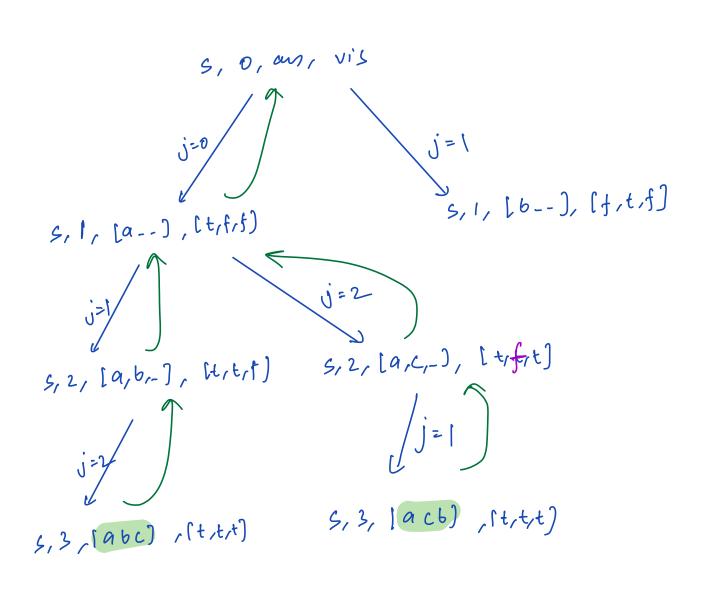
$$[a--], [a,b]$$

$$[a--], [a--], [a--],$$

```
Cocle
```

permutation (abc , 0, [---) , [0,0,0])

```
void permutations (S, i, am, vis) &
      if (i == 5.8ize()) {
           privi (am)
      for (j=0 to n-1) {
           if (vislj) == false) }
                 am (i) = s(j)
                 vislj) = tone
                 permutation (s, i+1, ans, vis);
                 vislj) = false
```



Time nomphonity
$$\Rightarrow$$

total f^n calls $*$ $TC of one f^n call

 $N!$ $*$ N
 $TC = O(N*N!) \Rightarrow O(0*N!)$
 $S(=O(N)$$