Lecture :- Backtracking

Agenda

- Quizzes
- What is backtracking ?
- Subsets of array
- Permutations of string.

Quiz1

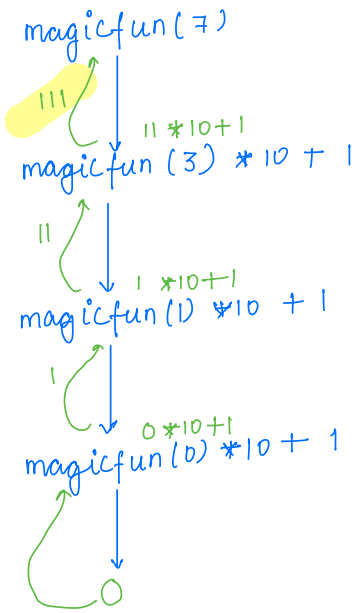
```
int magicfun(int n) {
    1 if(n==0) {
        return 0;
    }
    else {
    2. return magicfun(n/2) *10 + (n%2);
    }
}
```

a.) 100

b.) 111

c.) 99

d.) 112.

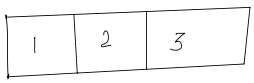


Qu.

```
void fun(char[] s, int x) {
    print(s);
    char temp;
    if (x < s.length/2) {
        temp = s[x];
        s[x] = s[s.length-x-1];
        s[s.length-x-1] = temp;
        fun(s, x+1);
    }
}
```

Output for fun("SCROLL", 0) ?

Qu: Given A[n] of distinct integers print all subsets using recursion. {Med- hard}



output:-

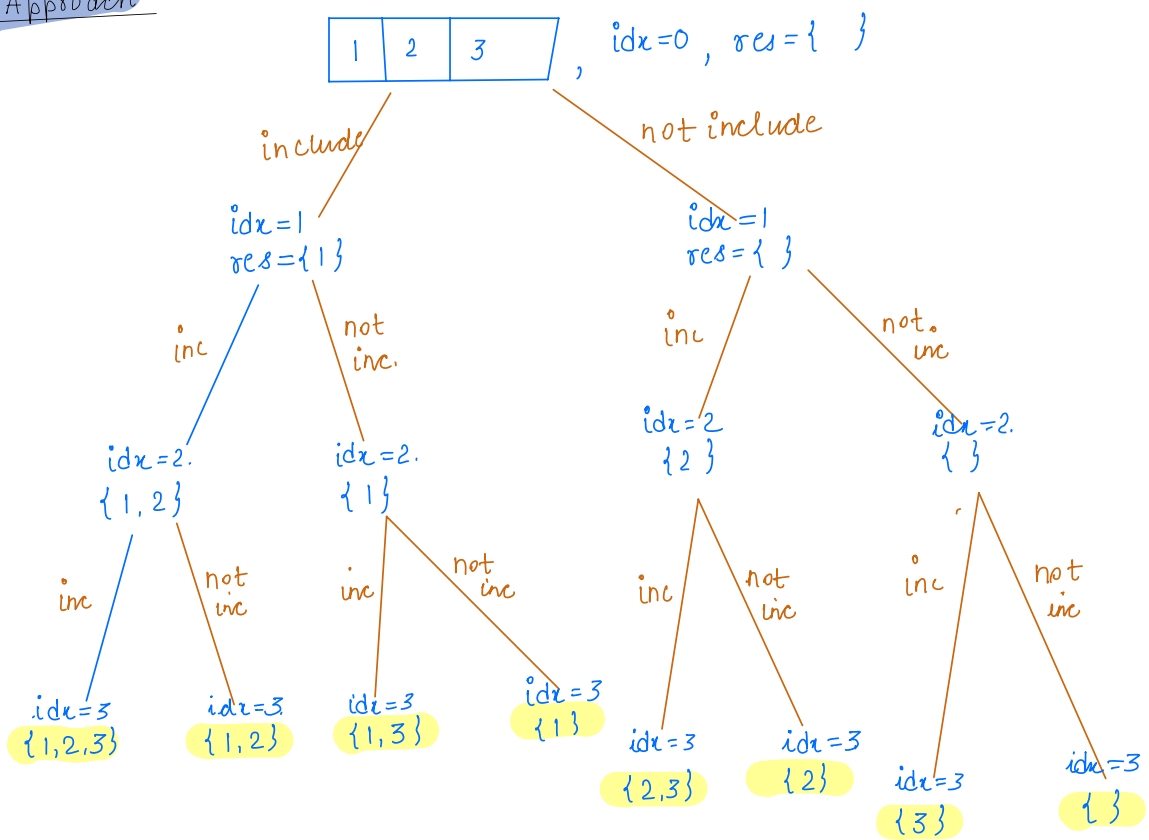
[ ]
{3}
{2}
{2 3}
{1}
{1 3}
{1 2}
{1 2 3}

abc → Subsequences

ordered | non-continuous

" "
c
b
a
ac
ab
bc
abc.

Approach


1 2 3
idx=0 , res={ }
include
not include
idx=1
res={1}
idx=1
res={ }
inc
not inc.
inc
not inc
idx=2.
{1,2}
idx=2.
{1}
idx=2
{2}
idx=2.
{ }
inc
not inc
inc
not inc
inc
not inc
inc
not inc
idx=3
{1,2,3}
idx=3
{1,2}
idx=3
{1,3}
idx=3
{1}
idx=3
{2,3}
idx=3
{2}
idx=3
{3}
idx=3
{ }

Pseudocode

```
void subsets (int[] arr, int idx, List<Integer> res) {
        if (idx == arr.length) {
            print (res);
            return;
        }
        res.add (arr[idx]);
        subsets (arr, idx+1, res);
        res.remove (res.size() -1);
        subsets (arr, idx+1, res);
}
```

inc

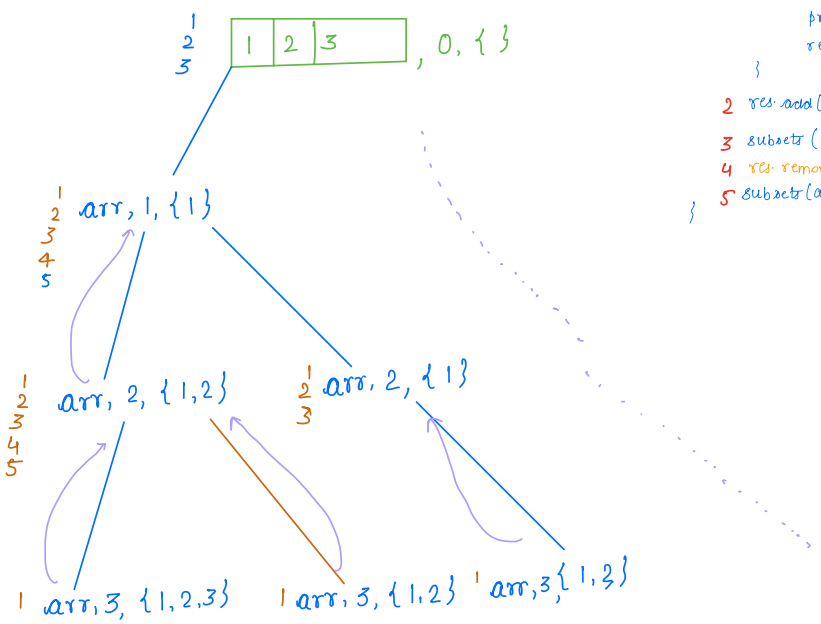
not inc

TC: >=O(2^n)

SC: O(n) → height of rec tree.

↑ len of array

Dry run:




```
void subsets(int[] arr, int idx, List<Integer> res) {
1   if(idx == arr.length) {
        print(res);
        return;
    }
2   res.add(arr[idx]);
3   subsets(arr, idx+1, res);      } inc
4   res.remove(res.size()-1);      } not inc
5   subsets(arr, idx+1, res);
}
```

{1,2,3}
{1,2}
{1,3}

Break: 10:04 – 10:14

Qu Total no. of permutations of a string with unique characters?

a) n^2

b) $n + \frac{n+1}{2}$

c) $\frac{n * (n-1)}{2}$

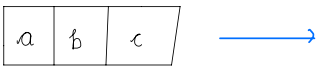
d) $n!$

perm(abc) $\longrightarrow$ abc, acb, bac, bca, cab, cba

$n! = 3! = 6$

Qu Given a character array with distinct element. Print all permutations of it without modifying it.

Example



→

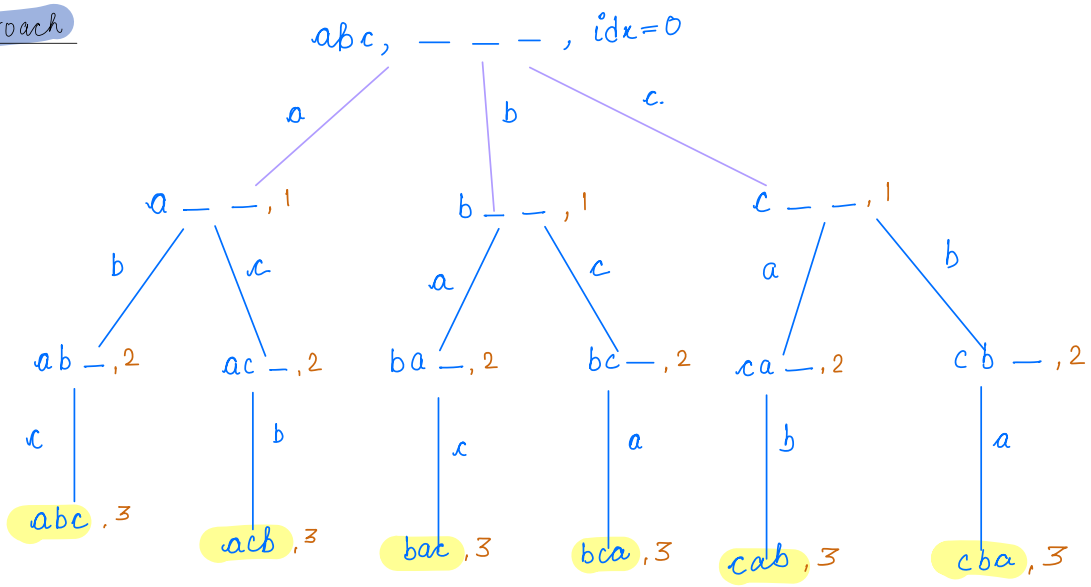
a b c
a c b
b a c
b c a
c a b
c b a

len = 3

3 * 2 * 1 = 6 perm.

↑ a, b, c (3)

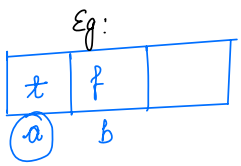
Approach


abc, _ _ _ , idx=0
a
b
c
a _ _ , 1
b _ _ , 1
c _ _ , 1
b
c
a
c
a
b
ab _ , 2
ac _ , 2
ba _ , 2
bc _ , 2
ca _ , 2
cb _ , 2
c
b
c
a
b
a
abc , 3
acb , 3
bac , 3
bca , 3
cab , 3
cba , 3

Pseudocode

```
void perm(char[] arr, int idx, ans[], visited[n]) {
    if (idx == arr.length) {
        print(ans);
        return;
    }
    All possibilities
    for(i=0; i<n; i++) {
        if (visited[i] == false) {
            visited[i] = true;
            ans[idx] = arr[i];
            perm(arr, idx+1, ans, visited);
            visited[i] = false;
        }
    }
}
```

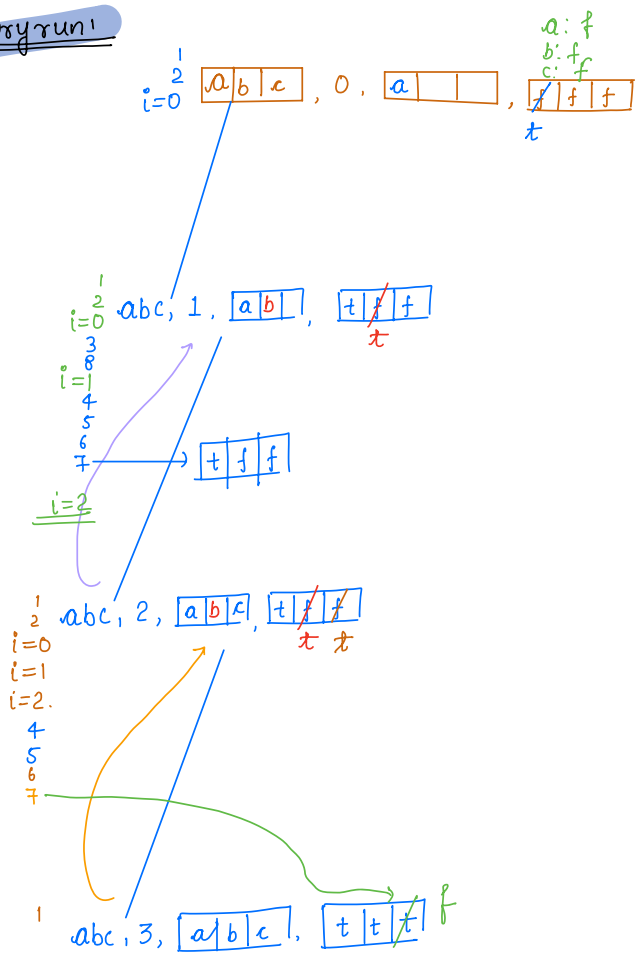
Eg:



TC: >= O(n!)

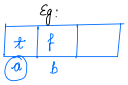
SC: O(n)

Dry run:




```
void perm(char[] arr, int idx, ans[], visited[n]) {
1   if (idx == arr.length) {
        print(ans);
        return;
    }
    All possibilities
2   for(i=0; i<n; i++) {
3       if (visited[i] == false) {
4           visited[i] = true;
5           ans[idx] = arr[i];
6           perm(arr, idx+1, ans, visited);
7           visited[i] = false;
        }
    }
8}
```

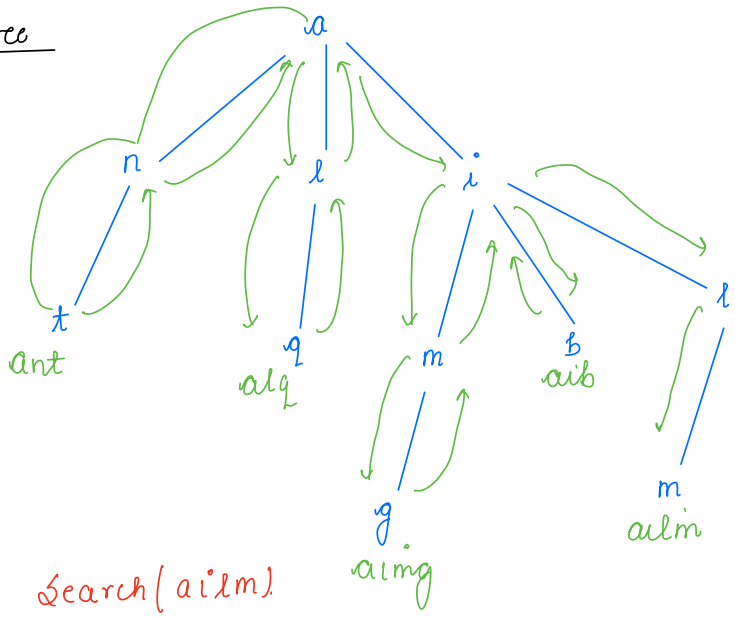
Eg:



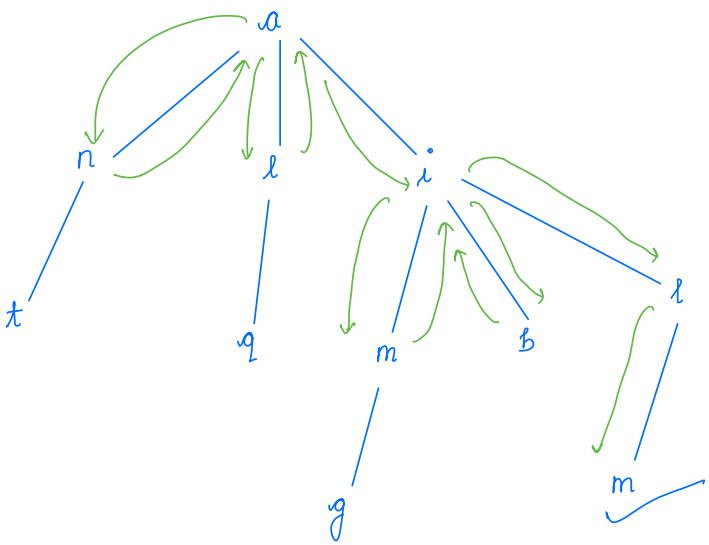
{ a b c }

Backtracking [pruning]

Brute force



Search(ailm)



Thankyou