★ permutations Another approach :

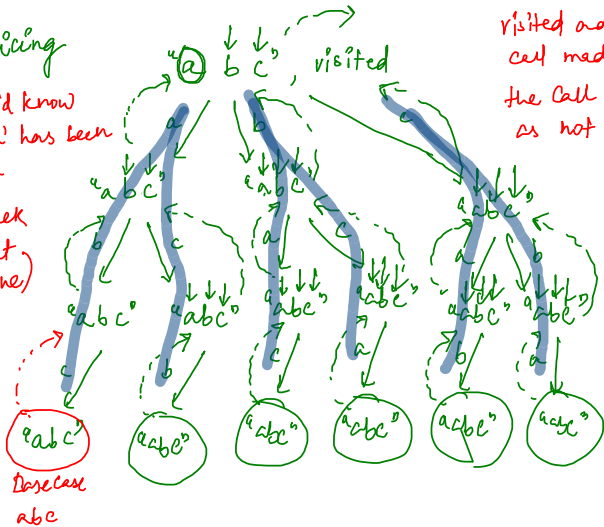
→ no slicing

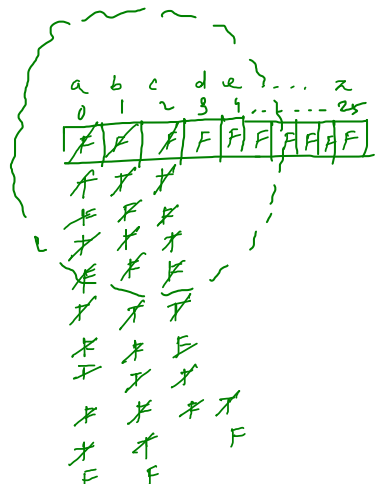
★ I should know that 'a' has been picked

★ Backtrack (erase what you have done)

★ As I picked a'ch' I marked visited and rec call made, after the call mark it as not visited

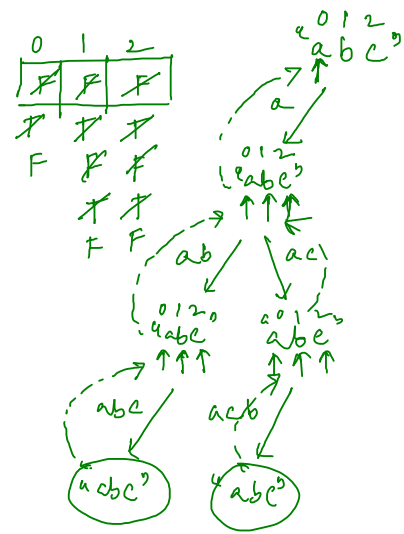
Sunday 11:59 AM deadline to start





```
// another approach
function permute(str, visited, path) {
  if (path.length == str.length) {
    console.log(path);
    return;
  }

  for (let i = 0; i < str.length; i++) {
    if (visited[i] == false) {
      visited[i] = true;
      permute(str, visited, path + str[i]);
      visited[i] = false;
    }
  }
}
```



```
6  function solve(nums, visited, path, ans) {
7
8      if(path.length == nums.length) {
9          ans.push([...path]);
10         return;
11     }
12
13     for(let i = 0; i < nums.length; i++) {
14         if(visited[i] == false) {
15             visited[i] = true;
16             path.push(nums[i]);
17             solve(nums, visited, path, ans);
18             visited[i] = false;
19             path.pop();
20         }
21     }
22 }
23
24
25 var permute = function(nums) {
26     const visited = [];
27     for(let i = 0; i < nums.length; i++) {
28         visited.push(false);
29     }
30     const path = []
31     const ans = []
32     solve(nums, visited, path, ans);
33     return ans;
34 };
```

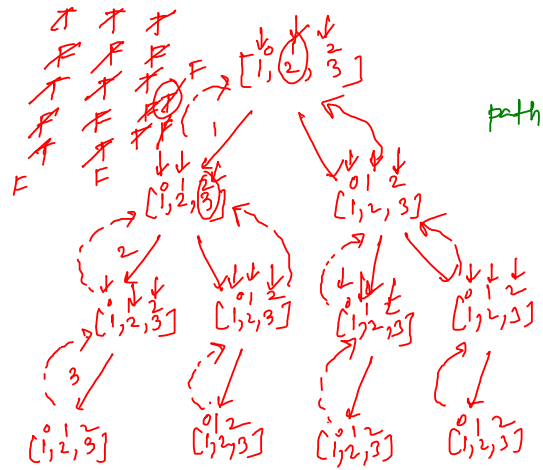
nums = [1, 2, 3]

* make new copy and add

ans = [[1,2,3] 1000, [1,3,2] 2000, [2,1,3] 3000, [2,3,1] 4000

visited = [F, F, F]

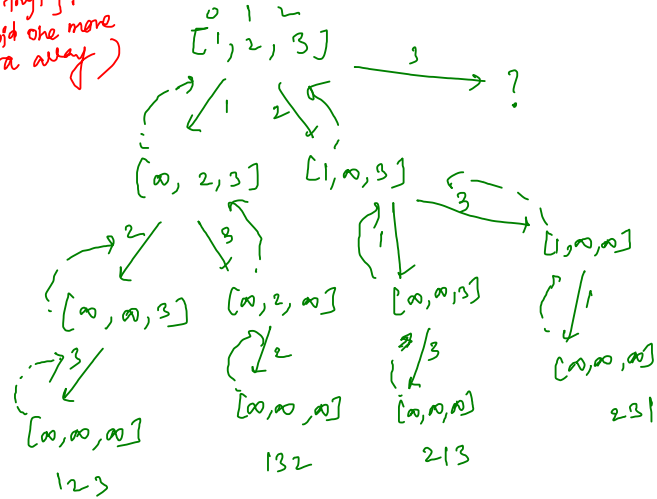
path 100 = [



```
6  function solve(nums, path, ans) {
7
8      if(path.length == nums.length) {
9          ans.push([...path]);
10         return;
11     }
12
13     for(let i = 0; i < nums.length; i++) {
14         if(nums[i] != Infinity) {
15             const oldnum = nums[i]
16             path.push(nums[i]);
17             nums[i] = Infinity;
18             solve(nums, path, ans);
19             nums[i] = oldnum;
20             path.pop();
21         }
22     }
23 }
24
25
26 var permute = function(nums) {
27     const path = []
28     const ans = []
29     solve(nums, path, ans);
30     return ans;
31 };
```

★ we are using the same nums array as visited

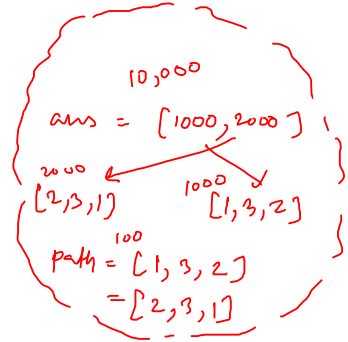
(Just trying to avoid one more extra array)



→ slicing
→ slicing + no duplicate
→ visited
→ visited + no duplicate } Backtrack
→ (use the same array as visited)

ans.push([...path])

① []
② [...path]
③ [...[1,3,2]]
④ [1,3,2]



Heap

9:35pm
- 9:50pm
Break

★ Decode Ways :

1 - a
2 - b
3 - c
4 - d
5 - e
6 - f
7 - g
8 - h
9 - i
10 - j
11 - k
12 - l
13 - m
14 - n
15 - o
16 - p
17 - q
18 - r
19 - s
20 - t
21 - u
22 - v
23 - w
24 - x
25 - y
26 - z

ip : "226"

op : "bbf" ((2)(2)(6))

"vf" ((22)(6))

"bz" ((2)(26))

ip : "123"

op : "abc" ((1)(2)(3))

"lc" ((12)(3))

"aw" ((1)(23))

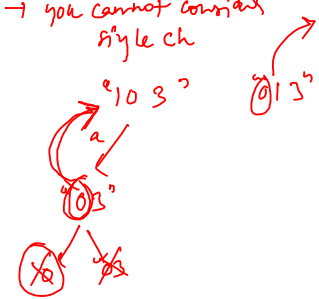
ip : 445

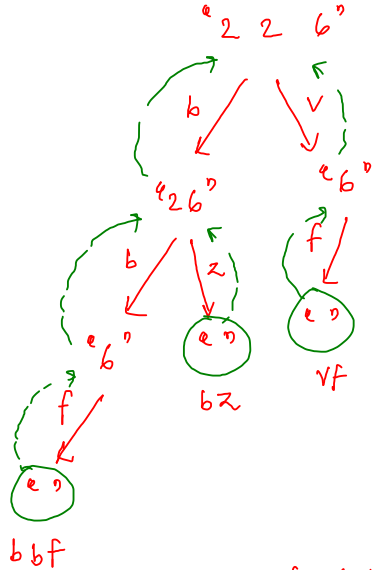
op : "d d e"

★ edge case,

103

→ you cannot consider single ch





* you can take a single digit always

* you can take two digits if it is <= 26

1. faith

f(str, idx, path)

decode idx → n

* for counting → optimize using DP

for printing → recursion + backtracking