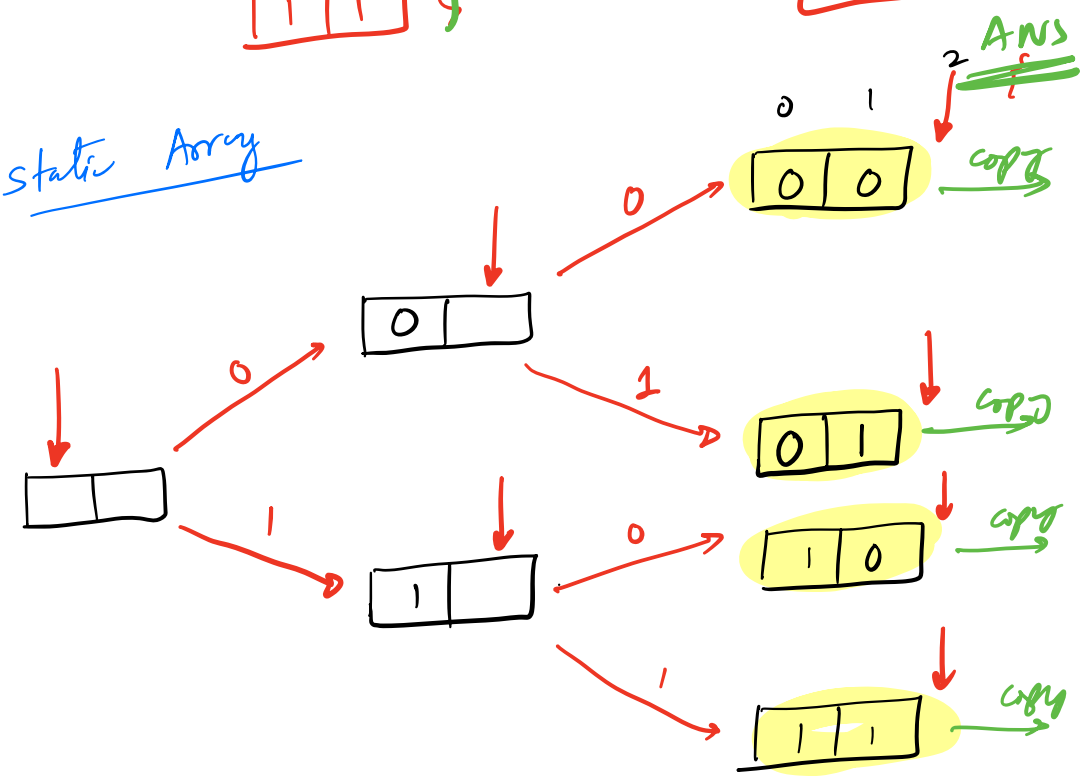
Backtracking < Bottom Up >

Q Given N. Generate all combinations of binary Array of size N.

N = 2 { [0 | 0], [0 | 1], [1 | 0], [1 | 1] } 4

N = 3 [0 | 0 | 0], 0 0 1, ..., [1 | 1 | 1] } 8

Static Array

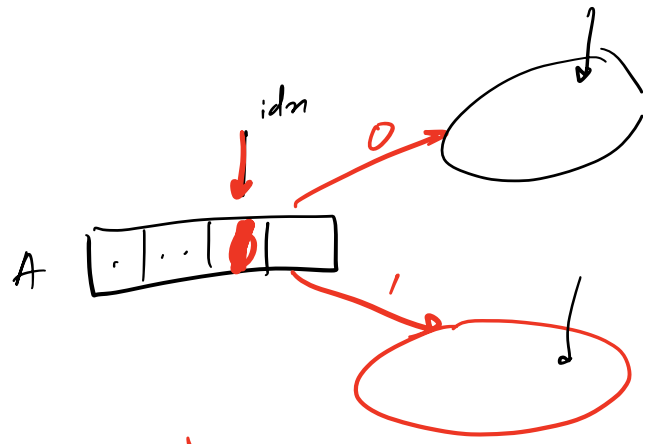


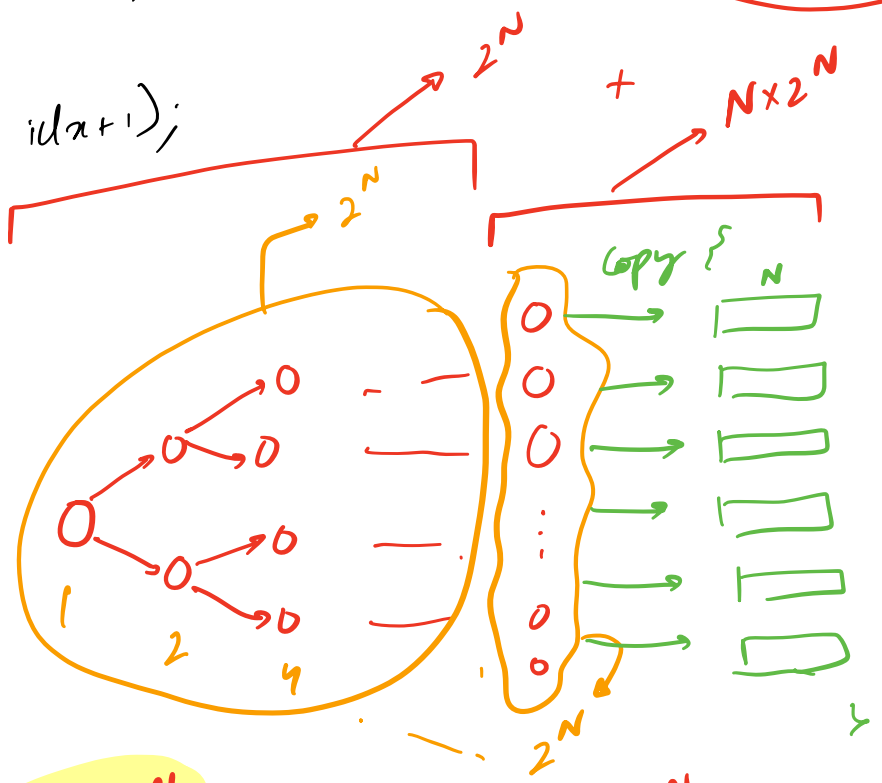
}

CODE

```
ANS → { }
int A[N];
void gen ( A[], idx ) {
  if (idx == N) {
    ANS.add( copy(A) );
    ret;
  }
  A[idx] = 0;
  gen( A[], idx+1);
  A[idx] = 1;
  gen ( A[], idx+1);
}
```

N





2^N

N^2

N

#f^n calls → $1+2+4+ \cdots 2^N$

$\sim 2^{N+1}$

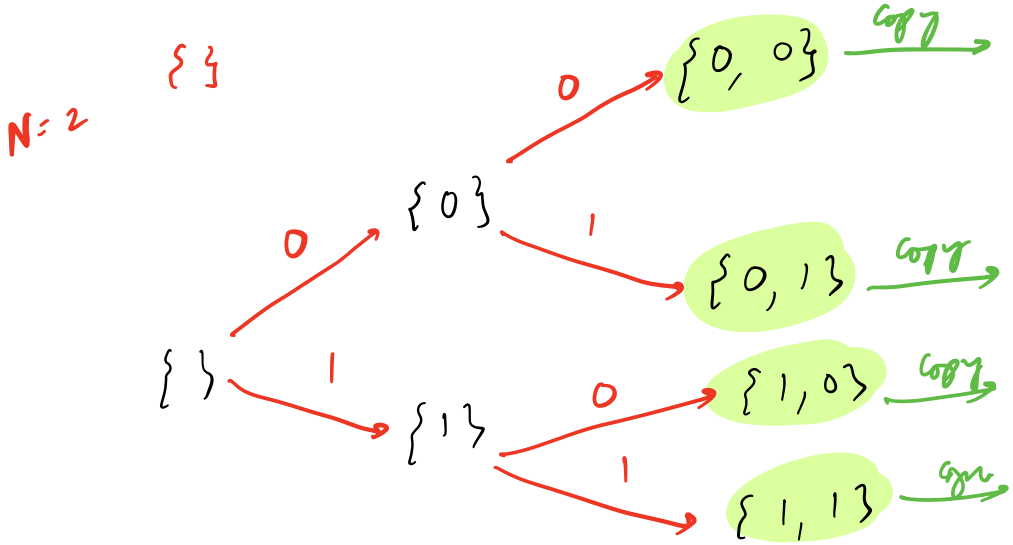
$\boxed{TC = O(N \cdot 2^N)}$

$\boxed{SC = O(N)}$

Dynamic Array

ANS

N = 2



ANS → { }

int cur[ ] → Dynamic Array

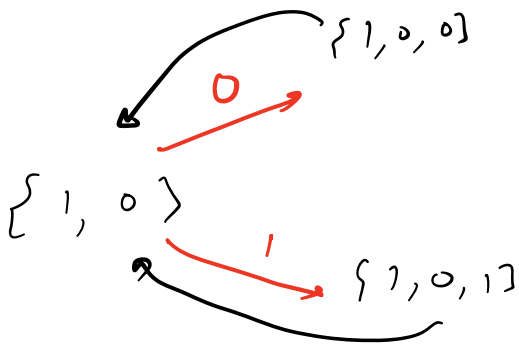
TC : SAME AS

SC : PREV

```
void gen ( cur[ ] ) {
    if ( cur.size() == N) {
        ANS. add( copy(cur));
        ret;
    }
    cur.push_back(0)
    gen(cur);
    cur.pop_back();
    cur.push_back(1);
    gen(cur);
    cur.pop_back();
}
```

do f^n undo

do f^n undo



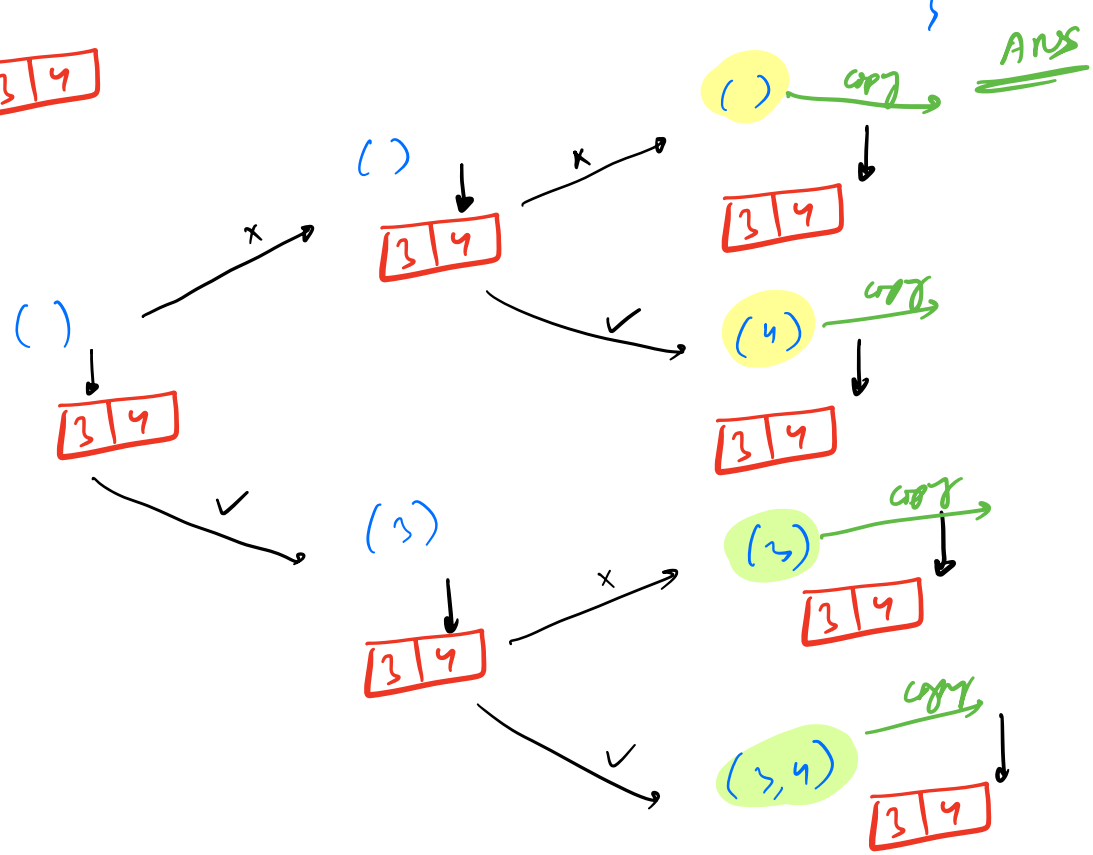
Q Given an Array A. Generate all subsets of it!

x → 0
✓ → 1

A | 1 | 2 | 3 |

0 0 0	:	x	x	x	←	{ ( )
1 0 0	:	✓	x	x	←	(1)
0 1 0	:	x	✓	x	←	(2)
0 0 1	:	x	x	✓	←	(3)
1 1 0	:	✓	✓	x	←	(1, 2)
1 0 1	:	✓	x	✓	←	(1, 3)
0 1 1	:	x	✓	✓	←	(2, 3)
1 1 1	:	✓	✓	✓	←	(1, 2, 3) }

A: | 3 | 4 |



// A[]

ANS → { }

cur [ ] → Dynamic Array

```
void gen(cur, idx) {
    if (idx == N) {
        ANS.add(copy(cur));
        ret;
    }
    gen(cur, idx+1);
    cur.push_back(A[idx]);
    gen(cur, idx+1);
    cur.pop_back();
}
```

fn

do

fn

undo

TC = SAME AS PREV

SC =

Q Given an array of N distinct elements. Generate all permutations

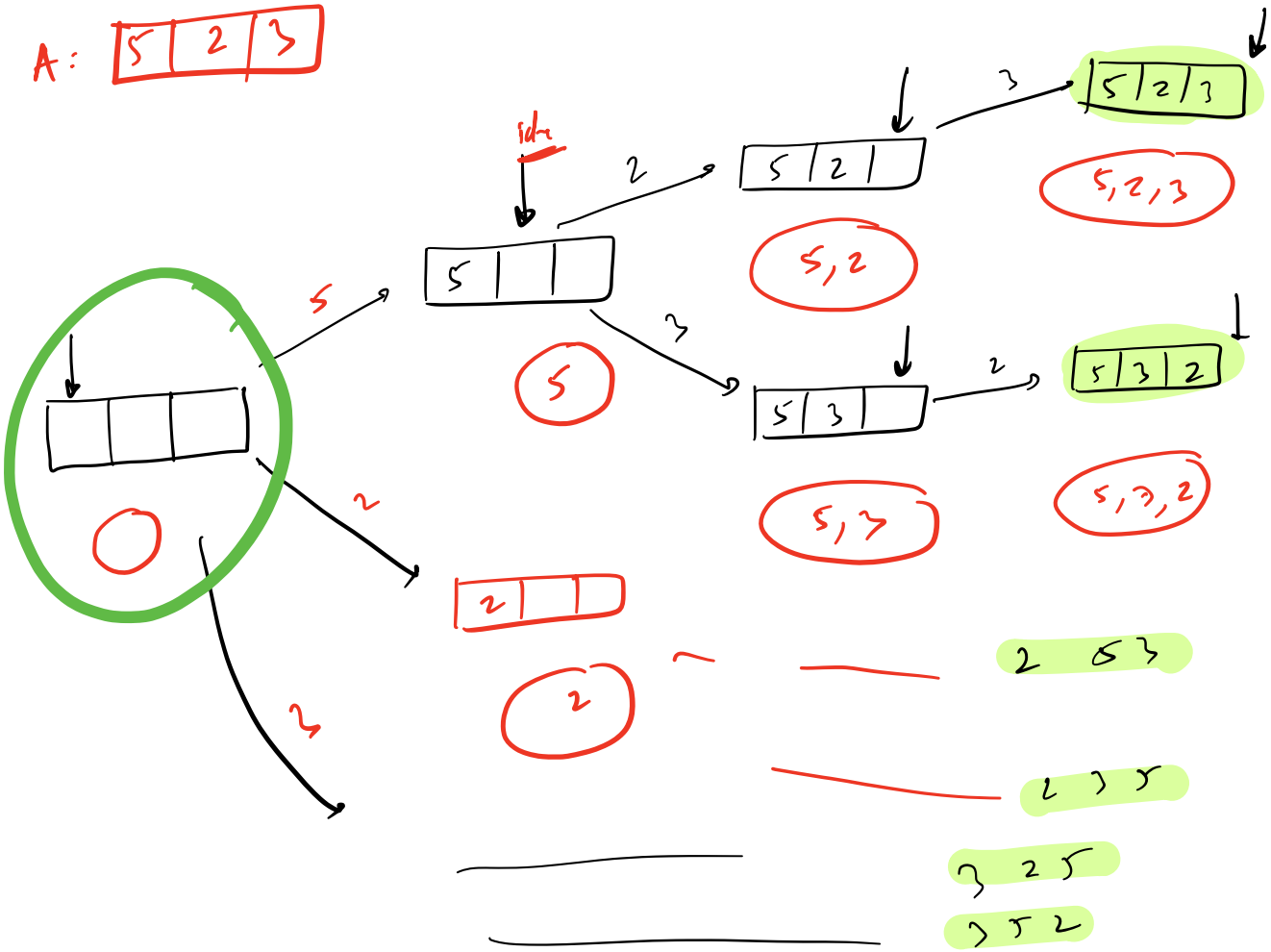
A : | 2 | 5 | 3 |

N

2 5 3
2 3 5
3 5 2
3 2 5
5 2 3
5 3 2

6 : N!

A : | 5 | 2 | 3 |

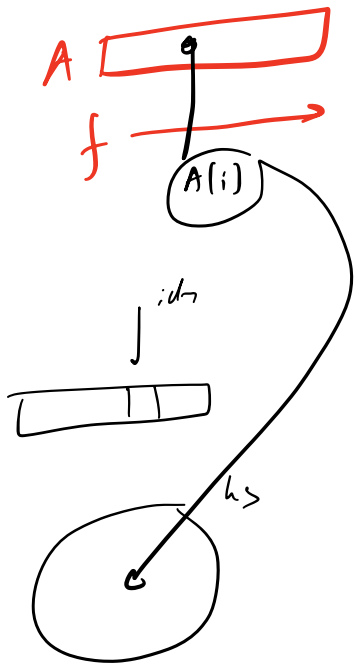


```
// A[]
int curr[N];                    N
ANS {}  |  Hashset<int> hs;      N

void gen( curr, idx, hs) {
    if ( idx == N) {
        ANS.add( Copy( curr));
        ret;
    }

    f( i:0; i<N; i++) {
        if( hs.find(A[i]) == false) {
            curr[idx] = A[i];
            hs.insert( A[i]);
            gen( curr, idx+1, hs);
            hs.remove( A[i]);
        }
    }
}
```

do

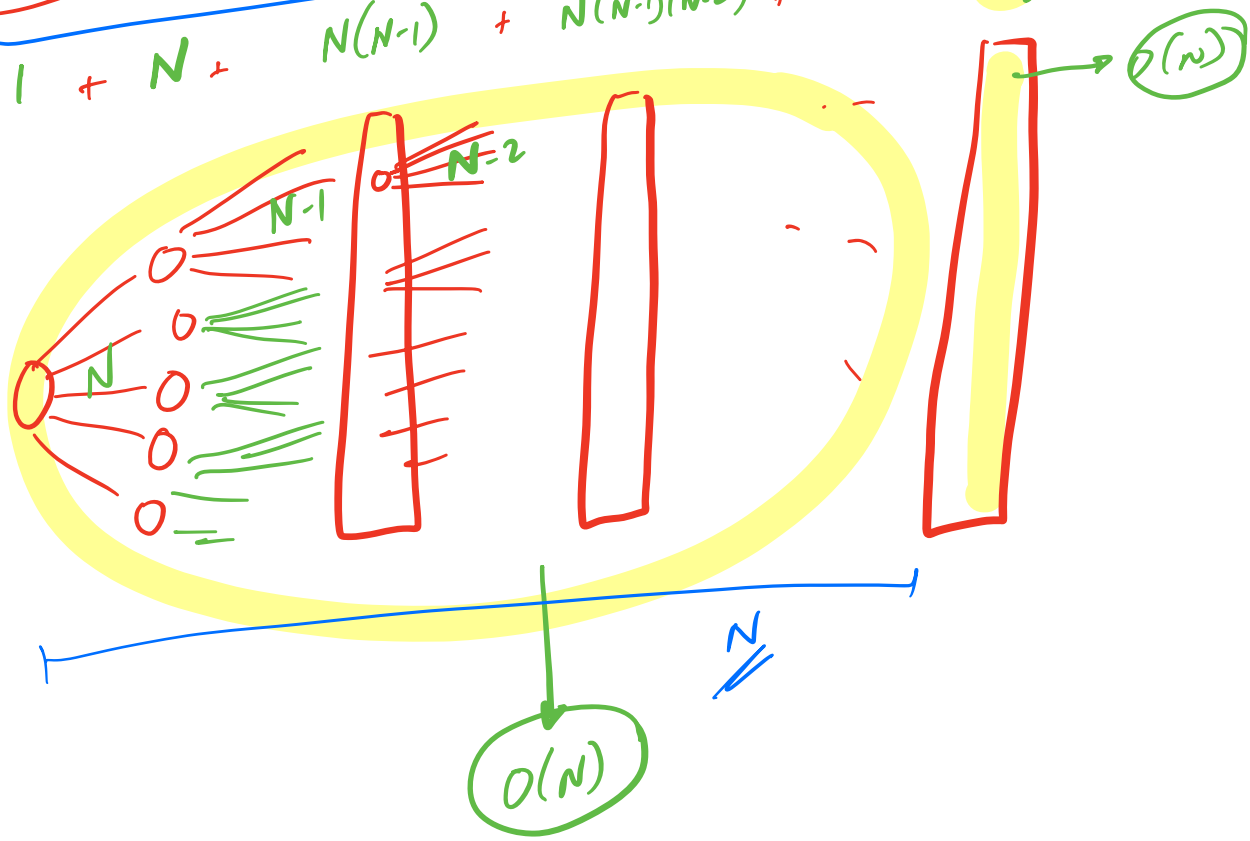
f^n

undo



f^n calls

$$\underbrace{1 + N + N(N-1) + N(N-1)(N-2) + \cdots + N!}_{< N!}$$



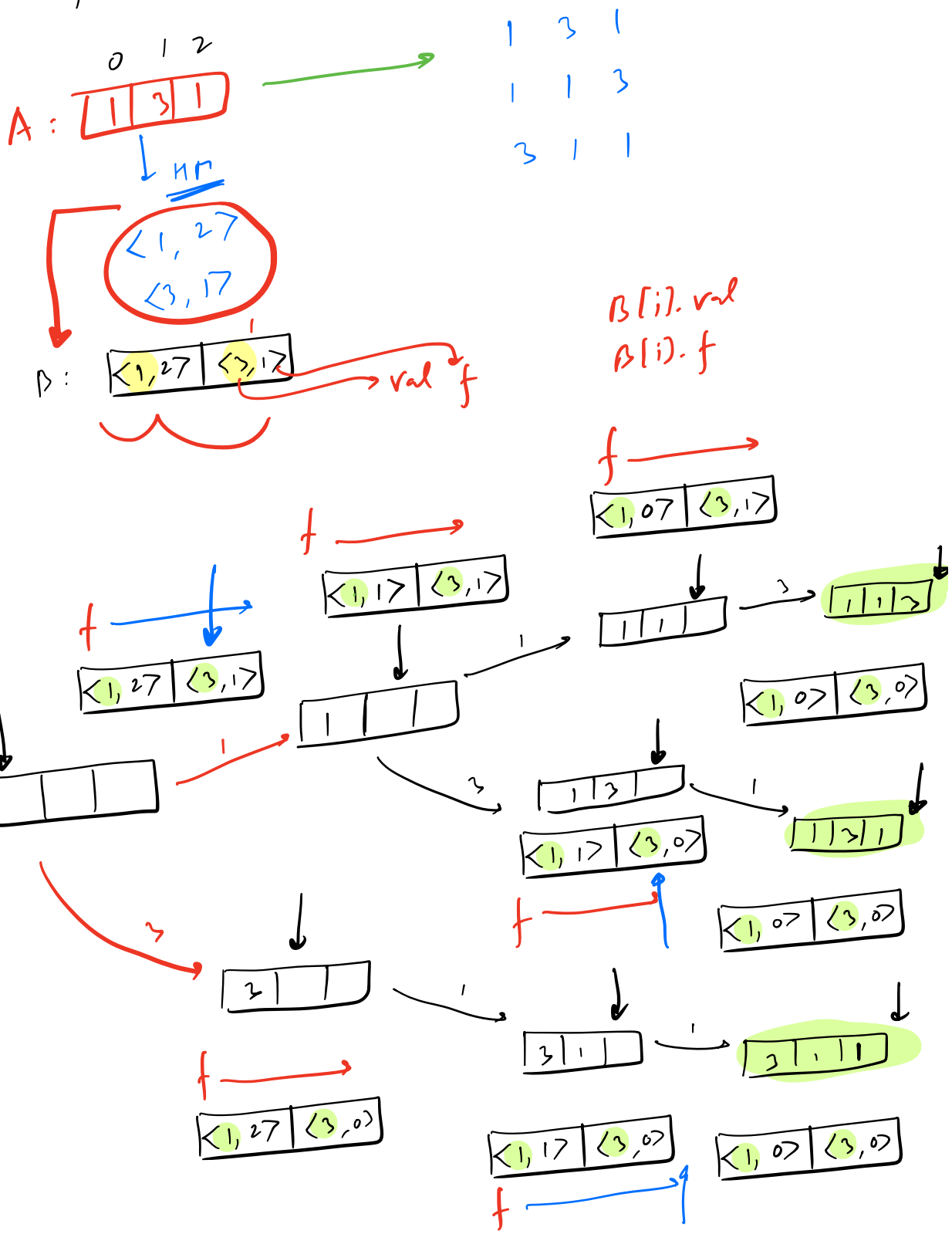
$$N! \times N + N! \times N$$

$$\boxed{TC = O(N \cdot N!)}$$

$$\boxed{SC = O(N)}$$

Explore more ways!

Q Given an array (DUPLICATES)
Generate all permutations!



// CODE

A[] → HM → B[]

A[] ↔ N, B[] ↔ M

```
ANS{}

int cur[N];
void gen( cur[], idx, B[]) {
    if ( idx == N) {
        ANS. add( copy( cur));
        ret;
    }
    f( i: 0 ; i< M ; i++) {
        if( B[i].f >0) {
            cur[idx] = B[i].val
            B[i].f --;
            gen( cur, idx+1, B);
            B[i].f ++;
        }
    }
}
```

do

f^n

undo

$N = 3$

A : | 1 | 2 | 3 |

0	0	0	0	0	( )
	1	0	0	1	(3)
	2	0	1	0	(2)
	3	0	1	1	(2,3)
	4	1	0	0	
	5	1	0	1	
	6	1	1	0	
$2^N - 1$	7	1	1	1	

i

ANS[]

```
for ( i=0; i< (1<<N); i++) {        → 2^N
    cur[];
    for(j=0; j<N; j++) {            → N
        if((i & (1<<j)) > 0){
            cur.add(A[j]);
        }
    }
    ANS.add(cur);
}
```

$2^N \cdot N$

$N = 20$