Backtracking

Dev Karan Singh (devkaran1231)
Expert at codeforces (1817)
5 star at codechef (2040)

Today's plan

- Print subsequences
- How to count using backtracking? trick
  - Count subsequence with given sum
  - Count paths in a matrix
- Rat in a maze
  - Print one answer
  - Print all answers
- N Queens

3 weeks

today's class

backtracking

2

live problem solving

subsequences

an → [1, 2, 3, 4, 5]

subarray	*subsequence	subset
→ contigious	→ non contigious	→ Anything from the array in any order
	→ order	
Eg: 2, 3, 4	Eg: 2, 3, 5	Eg. 4, 3
2, 3	1, 5	

printing subsequence

↳ print all subsequence of an array

arr ⇒ [2, 3, 7]

[ ]

[2]

[3]

[7]

[2, 3]

[3, 7]

[2, 7]

[2, 3, 7]

all the subsequences

ind

an → [2, 3, 7]

0 1 2

ind *

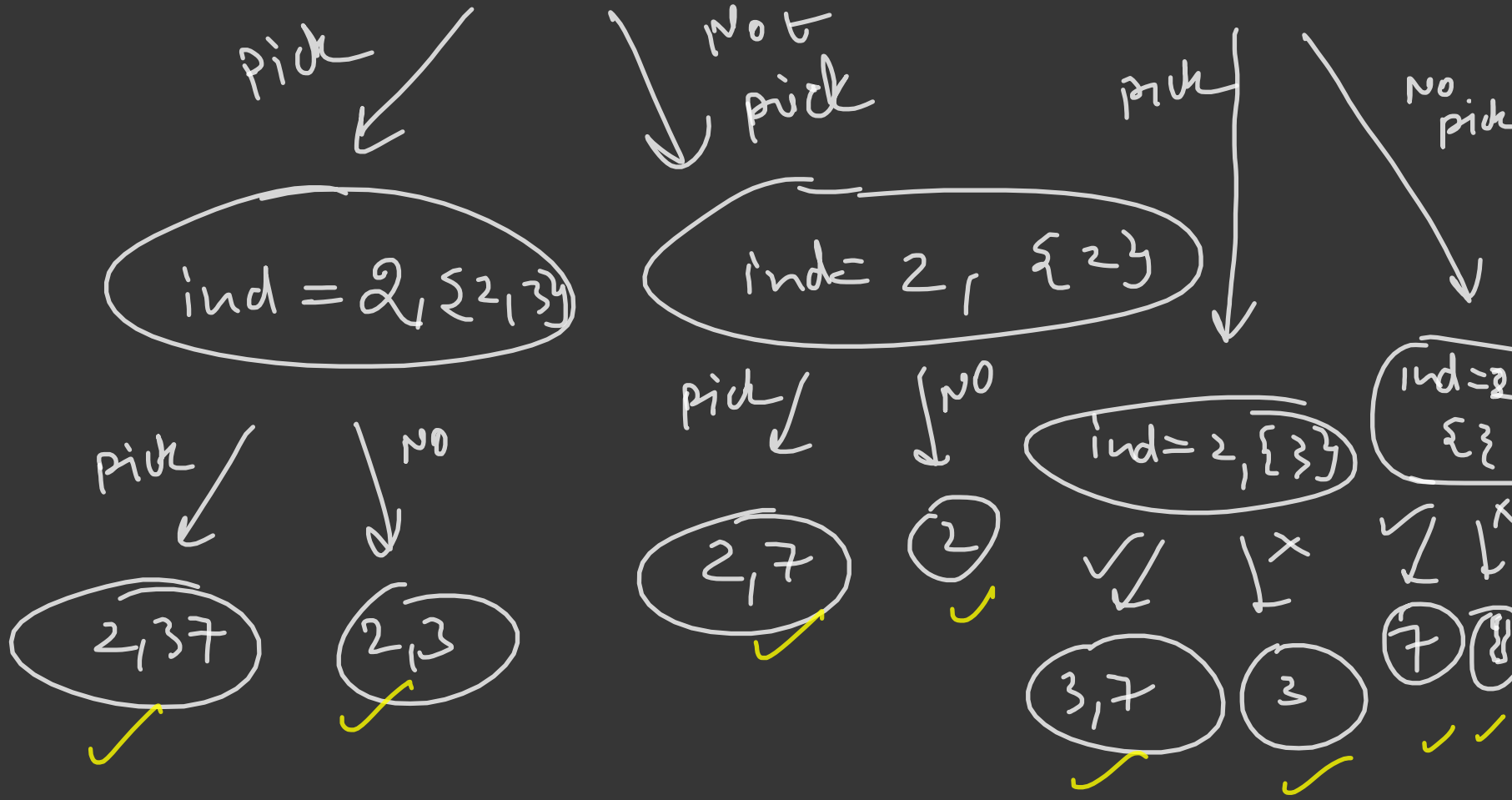
ind = 0, {}

Pick

Not pick

ind = 1, {2}

ind = 1, { }


pick
Not pick
ind = 2, {2,3}
ind = 2, {2}
pick
No
pick
No
ind = 2, {3}
ind = 2, {}
pick
No
pick
No
2,37
2,3
2,7
2
3,7
3
7
{}

ind

pick ✓ | not pick

```
void function(int ind, vector<int> &vec){
    if (ind == n){
        print(vec),
        return;
    }
    vec.push_back(arr[ind]),
    function(ind+1, vec);
    vec.pop_back();
```

pick ⟸

not pick ← [function(ind+1, vec),]

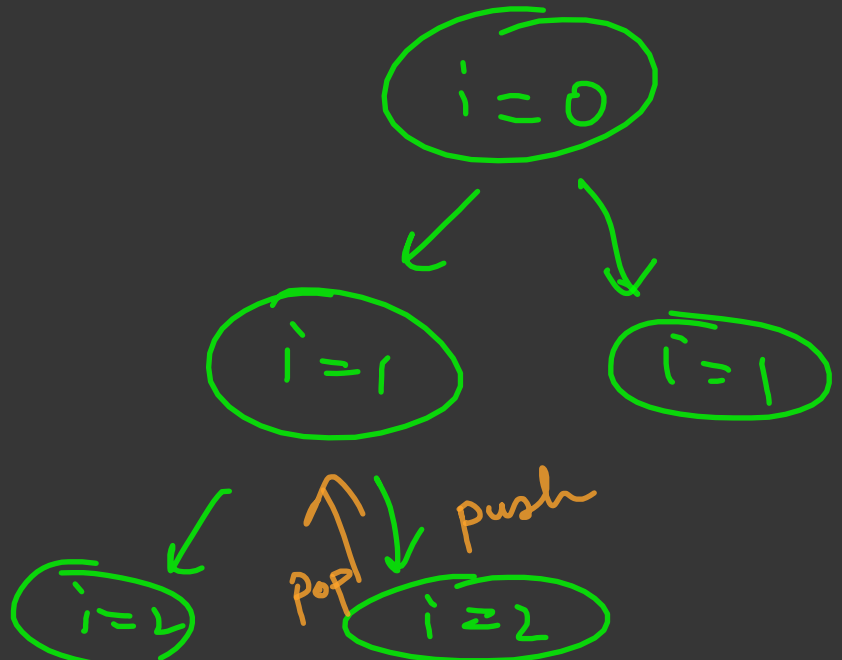
}

arr → [2, 3]

vec → [ ]

[ ]

[2]

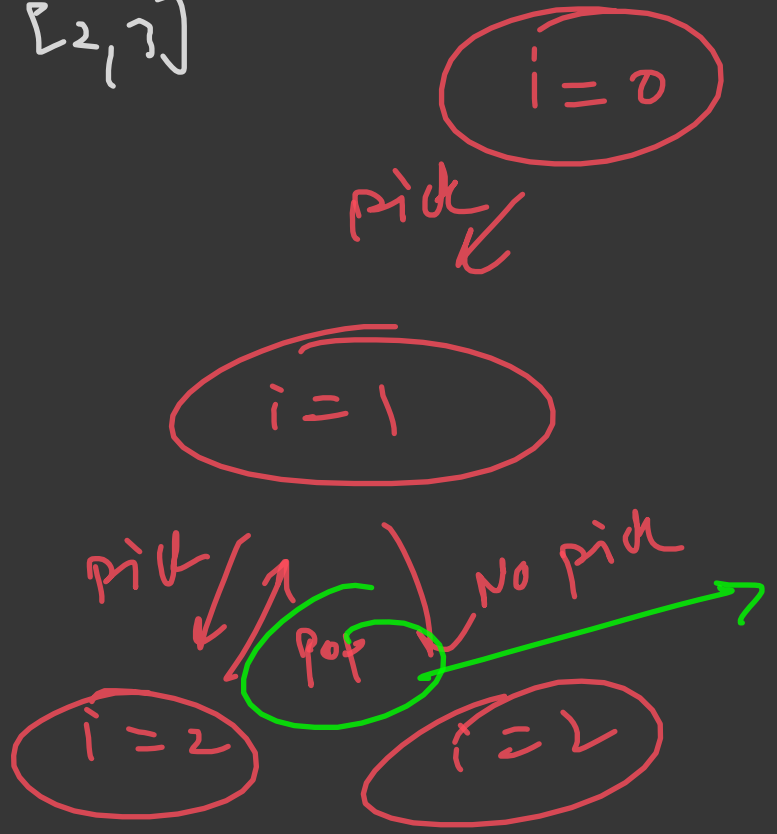


arr → [2, 3]

vector → [2]

2, 3

2



Backtracking step

before making a call at i = 1

↳ [2]

Backtracking

→ pick something, do the work,

then pop (backtrack to the prev state)

recursion

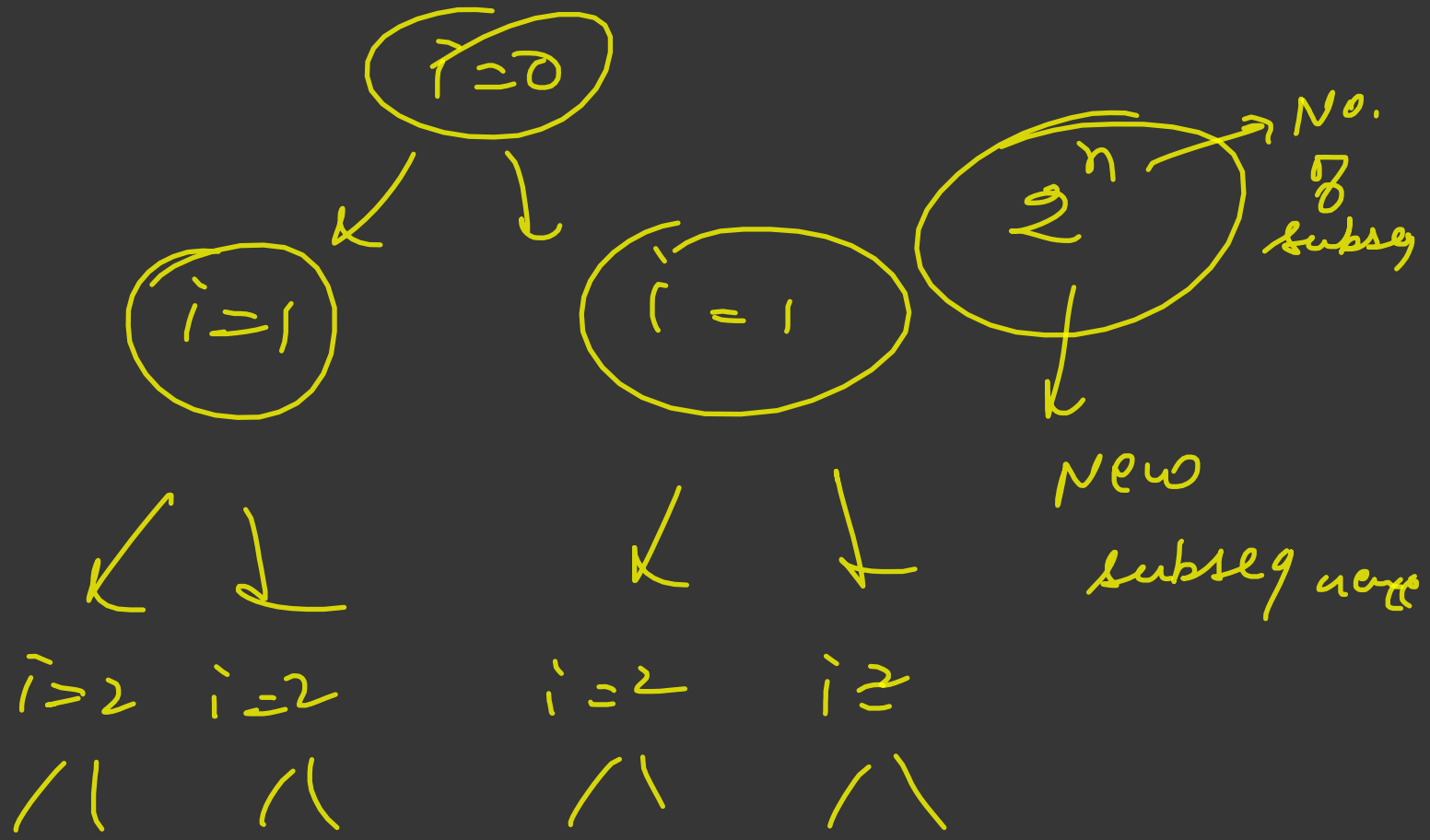
→ trying all ways

n index → pick
→ No pick

2 options

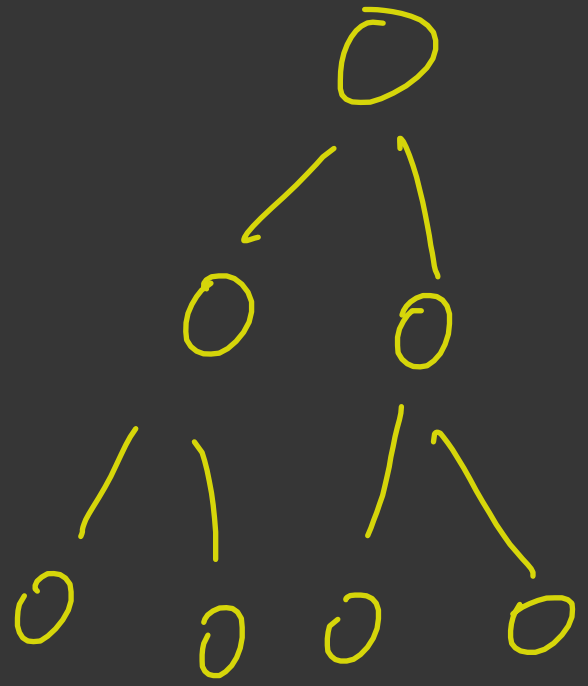
$$\underbrace{2 \times 2 \times 2 \times 2 \cdots 2}_{n \text{ times}}$$

→ 2^n final outputs


i=0
i=1
i=1
i=2
i=2
i=2
i=2
2^n
No. of subseq
New subseq

space

$O(n)$



2^n nodes

$2^n \times n \longrightarrow$ $> 10^8$ for $n = 30$

$\longrightarrow$ HC

How to count using backtracking?

Q count subsequences with sum = s

```
int generat() {
    if (i == n){
        if (sum == target){
            return 1;
        }
```

New Base case

```
        return 0;
    }
    int pick = generate(i+1, sum+arr[i])
    int no-pick = generate(i+1, sum);
    return pick + no-pick;
}
```

arr → [2, 3] target = 3

2^n

n → sc

~~recursion stack~~

i = 0, s = 0 → 0+1 → 2

✓ 0

0 ✗ 0+1

i = 1, s = 2

i = 1, s = 0

0

0

✓ 1

0 ✗

i = 2, s = 5

i = 2 s = 2

i = 2 s = 3

i = 2 s = 0

count paths in a matrix

	0	1	2
0	s	✓	✓
1	✓	✗	✓
2	✓	✓	e

if () {
return 1;
}
return 0;

→ end

n-1, m-1 → 0 based

2

s, e are safe

→ Right

↓ Down

✗ → A trap

✓ → safe

$$(i, j) \xrightarrow{\text{Right}} (i, j+1)$$

$$(i, j) \xrightarrow{\text{Down}} (i+1, j)$$

(0,0)

Down → (1,0)

Right → (0,1)

(2,0)

→ (3,0)

am[1][1] = 1 → trap

$tc \rightarrow 2^{(n \times m)}$

$sc \rightarrow n \times m$

$n \times m$

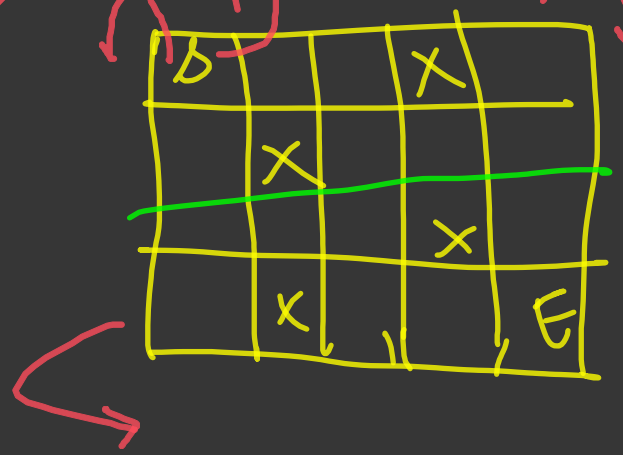
$n+m$

$2 \times 2 \times 2 \times 2$

2

nm times

Rat in a maze

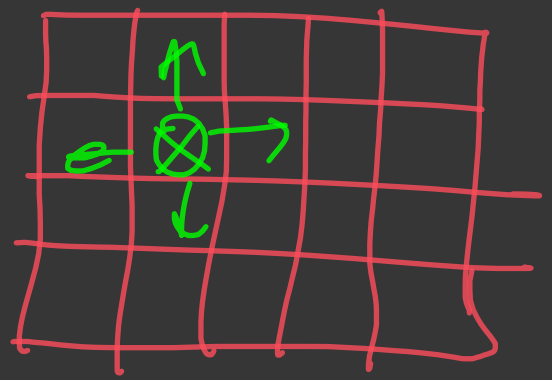


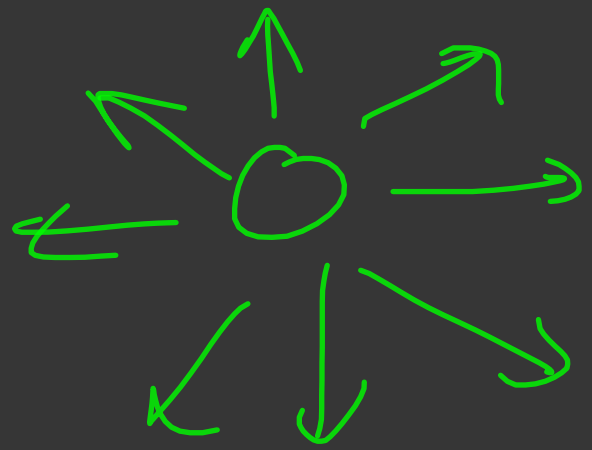
(i, j) → trap

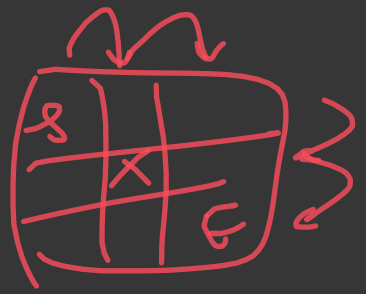
work

(Backtrack)

You cannot visit a block more than once

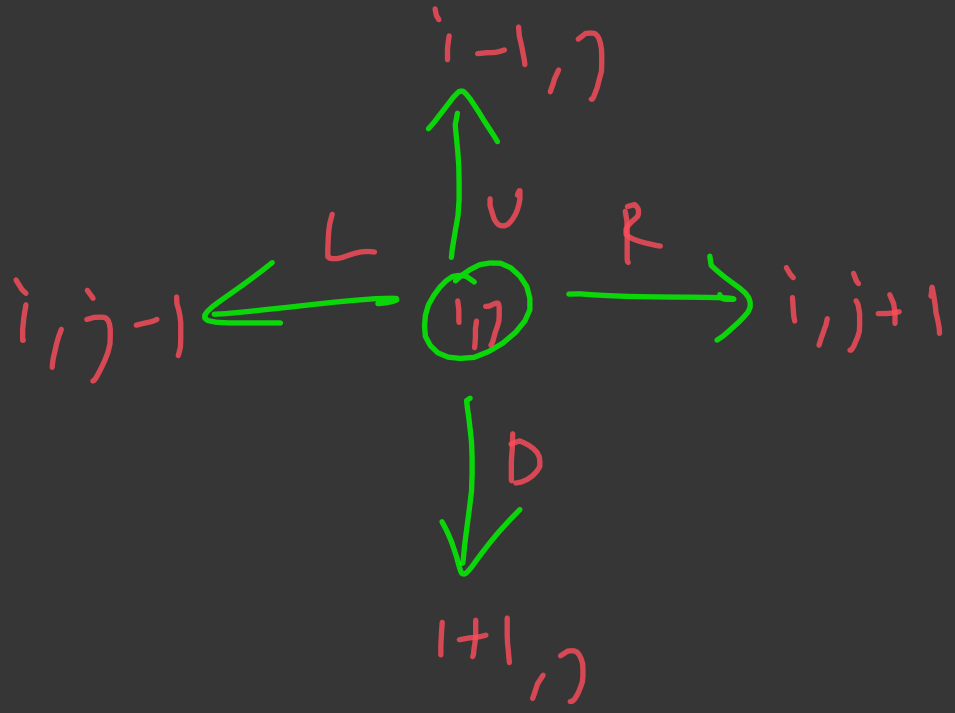






RRDD

DDRR



$d_i \rightarrow$

0	1	2	3
0	+1	0	-1

$d_j \rightarrow$

0	1	2	
+1	0	-1	0

string →

0	1	2	
R	D	L	U