1. coin change

[1, 2, 5] ✓ amount 11 ✓

3 (5 5 1) = 11

2, 2, 5, 2 = 11

1, ... 1 = 11

~~1, 2, 3, 4, 6~~

[1, 3, 4, 6] ≠

(5, 1, 1) = 3 coin

2 coin

(3, 4)

we have to do exhaustive search

coin [1, 3, 5]

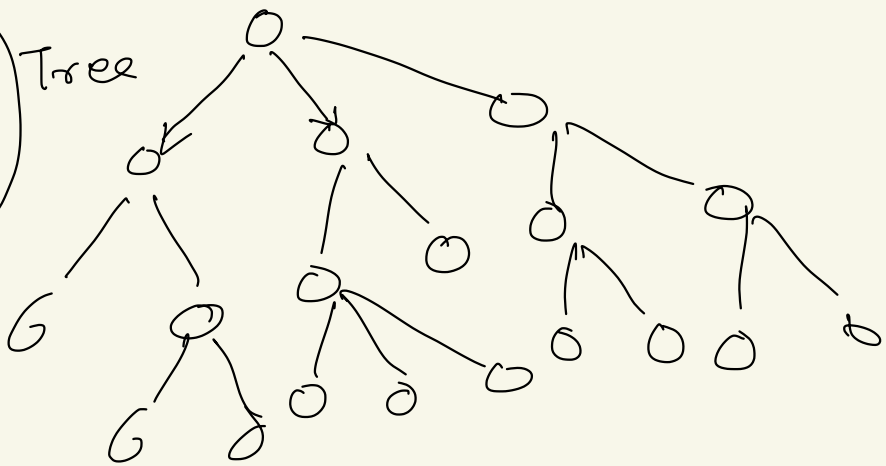
15

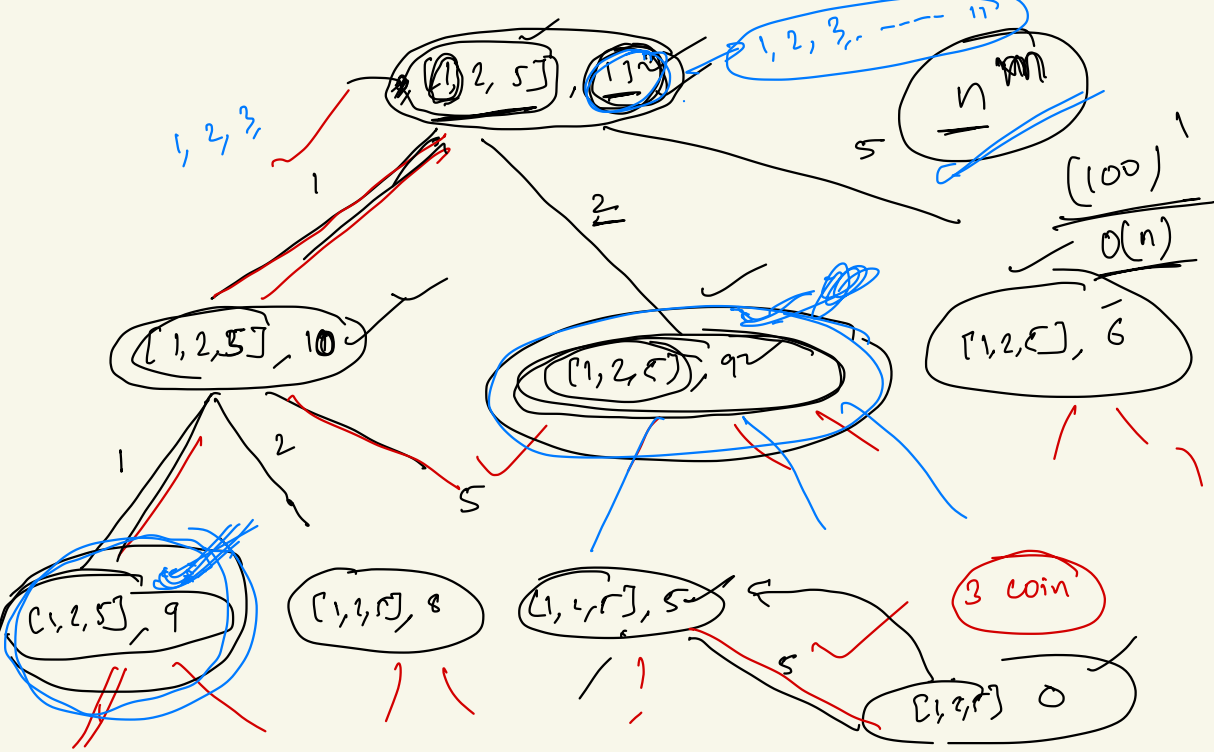
backtracking

DAG

Directed
Acyclic
Graph

Tree




1, 2, 5
11
1, 2, 3, ---- 11
n
5
1, 2, 3,
1
2
(100)
1
O(n)
[1, 2, 5], 10
[1, 2, 5], 9
[1, 2, 5], 6
1
2
5
[1, 2, 5], 9
[1, 2, 5], 8
[1, 2, 5], 5
3 coin
5
[1, 2, 5] 0

```
int dfs (int[] coins, int amount) {                    O(amount)
    if (amount == 0) return 0;
    if (amount < 0) return Integer.MAX_VALUE;
    int res = Integer.MAX_VALUE;
    for (int i=0; i < coins.length; i++)
        int curres = dfs (coins, amount - coin[i]);
        if (curres == Int.MaxValue)
            continue;
        res = Math.min (res, 1+curres)
    }
    return res
}
```

if (hm.containsKey(amount)) { hm.get(amount) }

hm.put (amount, res);

[1, 1, 1, 1, 1] target = 3

+1 +1 +1 +1 +1 = 5

-1 +1 +1 +1 +1 = 3

+1 -1 +1 +1 +1 = 3

-+++++

+-+++

++-++

+++-+

++++-

+1 -1 +1 +1 ⇒ 1

2

```
dfs ( int[] nums , int i, int sum, int target )
{
    if ( i == nums.length )
    {
        if ( sum == target ) ans++;
        return;
    }
    dfs ( num, i+1, sum + nums[i], target )
    dfs ( num, i++, sum - nums[i], target )
}

dfs( nums, 0, 0, target )
```

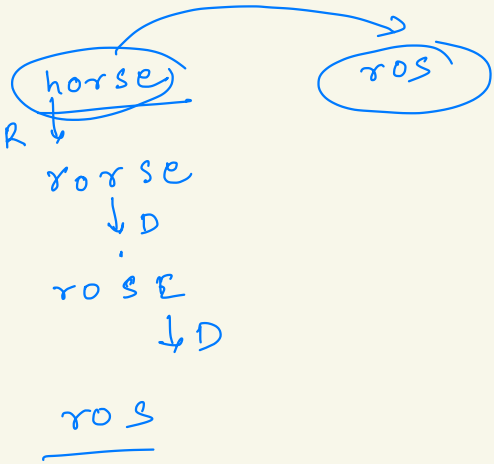
## Edit distance

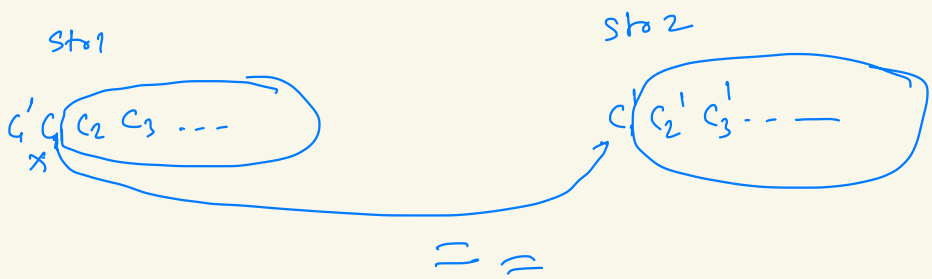
cost

Insert — 1

Delete → 1

Replace —




Sto 1
Sto 2

```
int dfs ( String p , String q, int i , int j )
{
    if ( i == p.length )
    {
        return q.length - j ;
    }
    if ( j == q.length ) {
        return p.length - i ;
    }
    if ( p.charAt(i) == q.charAt(j) )
    {
        return dfs ( p, q, i+1 j+1 );
    }
    (memo[i][j] != null) return memo[i][j]
    int cost1 = dfs( p, q, i+1, j+1 ) ;
    int cost2 = dfs ( p, q, i+1, j ) ;
    int cost3 = dfs ( p, q, i, j+1 ) ;
    memo[i][j] = 1 + min( cost1, cost2, cost3);
    return 1+min ( cost1, cost2, cost3 )
}
```

"" "abc"
3 insert

"xabc" "abc"
3 delete

"abc" "apc"

R

D

I

O ( p.length × q.length )

O ( 3 ^ min(p,q) )

p q r