

1. Climb Stairs with Min Moves

2. Goldmine

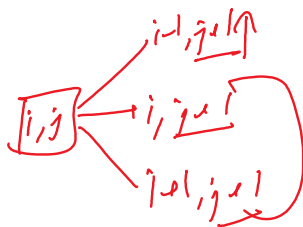
- 3. Target Sum Subset
 - 4. CC Comb
 - 5. CC Perm
 - 6. zero-one KS
 - 7. Unbounded KS
 - 6. Fractional KS
- DP

```
for(int j = dp[0].length - 1; j >= 0; j--){
    for(int i = 0; i < dp.length; i++){
        // ...
    }
}
```

0	1	4	2	8	2
4	3	6	5	0	4
1	2	4	1	4	6
2	0	7	3	2	2
3	1	5	0	2	4
2	7	0	8	5	1

0	26	24	21	14	12	2
1	31	26	23	17	6	4
2	28	27	20	11	10	6
3	29	25	25	13	8	2
4	33	26	28	18	6	4
5	32	30	18	17	9	4
	0	1	2	3	4	5

```
if(j == dp[0].length - 1){
    // ...
} else if(i == 0){
    // ...
} else if(i == dp.length - 1){
    // ...
} else {
    // ...
}
```



4

d1
d2
d3

9:45 to 9:55
table

```

int[][] dp = new int[costs.length][costs[0].length];
for(int j = dp[0].length - 1; j >= 0; j--){
    for(int i = 0; i < dp.length; i++){
        if(j == dp[0].length - 1){
            dp[i][j] = costs[i][j];
        } else if(i == 0){
            dp[i][j] = costs[i][j] + Math.max(dp[i][j + 1], dp[i + 1][j + 1]);
        } else if(i == dp.length - 1){
            dp[i][j] = costs[i][j] + Math.max(dp[i][j + 1], dp[i - 1][j + 1]);
        } else {
            dp[i][j] = costs[i][j] + Math.max(dp[i][j + 1], Math.max(dp[i - 1][j + 1], dp[i + 1][j + 1]));
        }
    }
}

int ans = dp[0][0];
for(int i = 1; i < dp.length; i++){
    if(dp[i][0] > ans){
        ans = dp[i][0];
    }
}

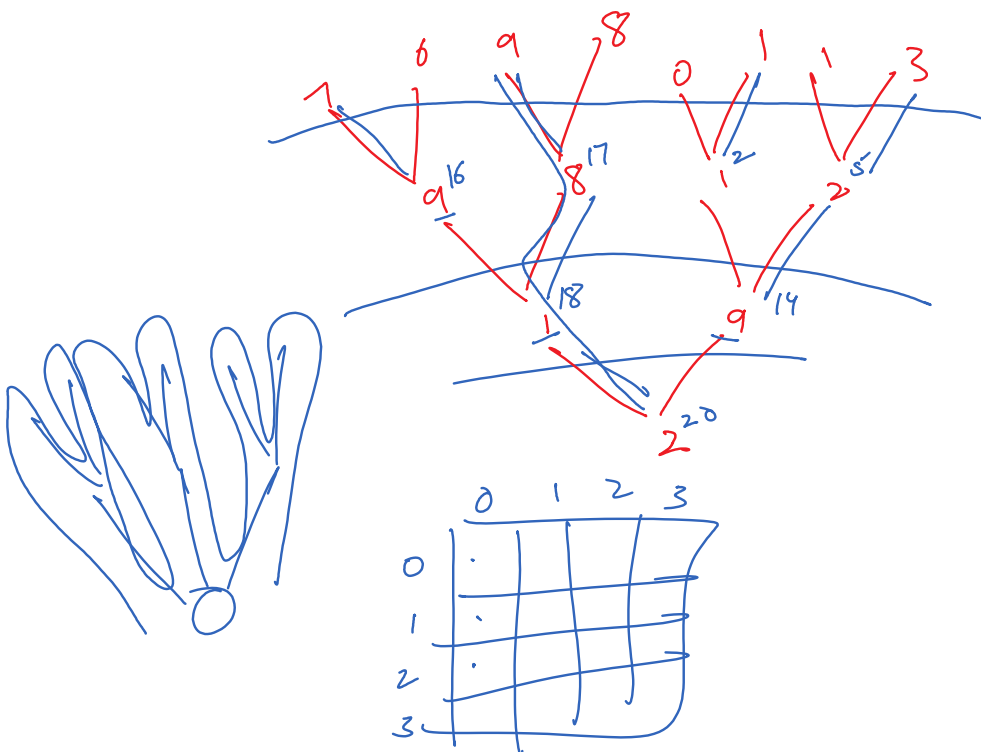
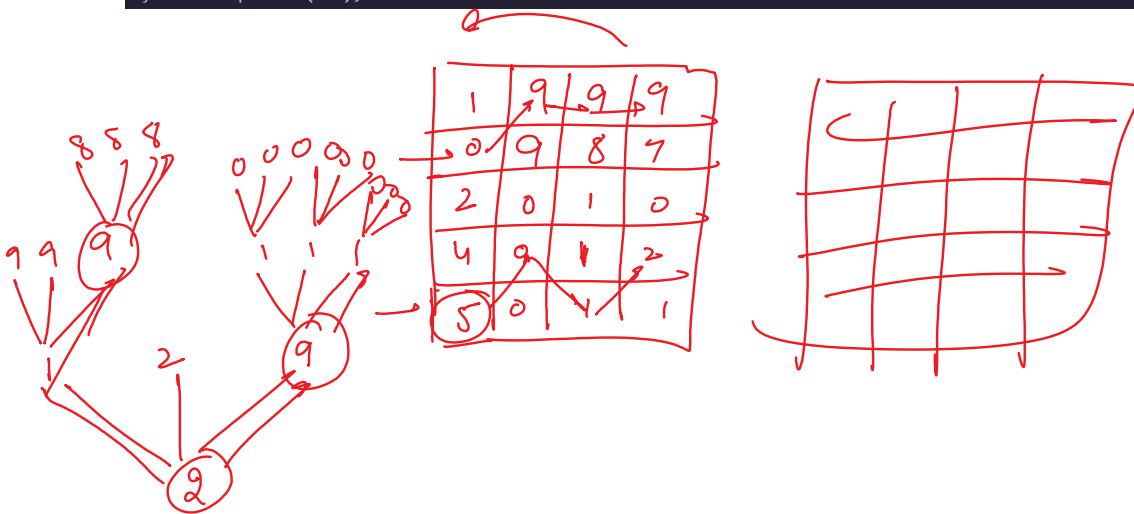
System.out.println(ans);

```

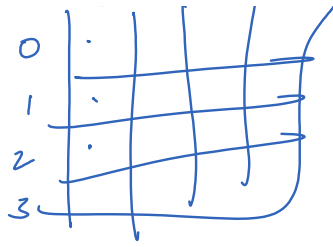
9:45 to 9:55

1. Fill table for code

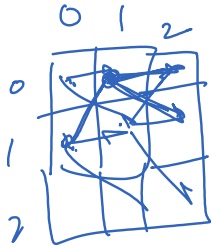
2. Try to print path



03/



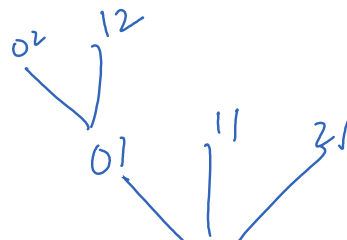
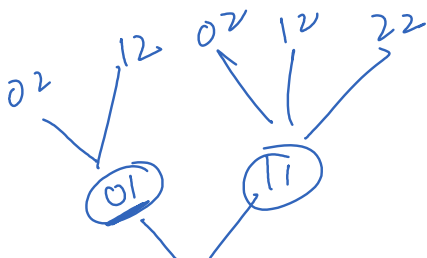
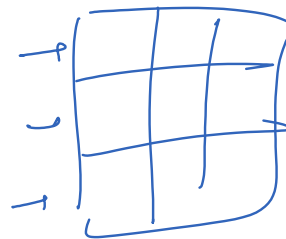
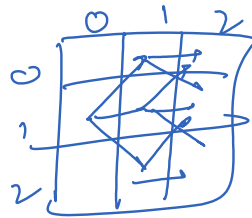
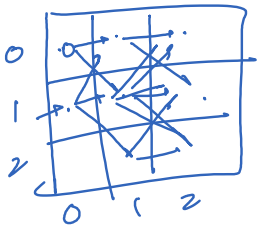
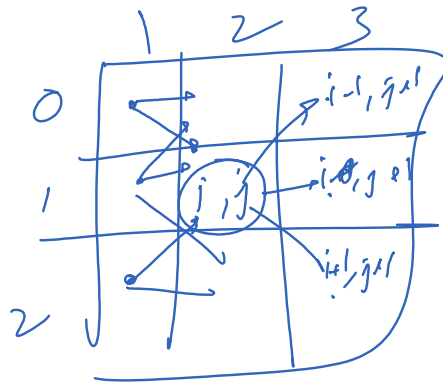
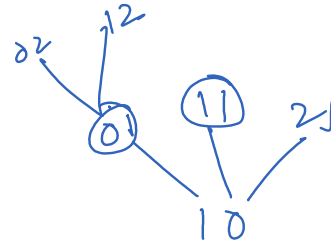
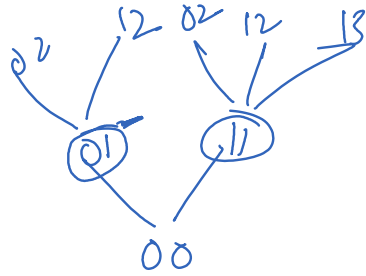
03

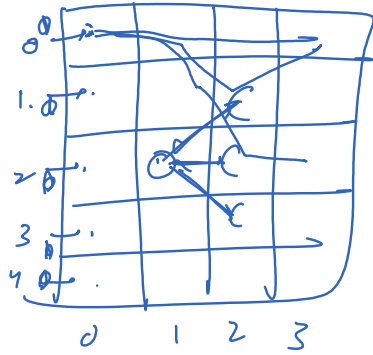
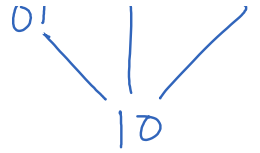
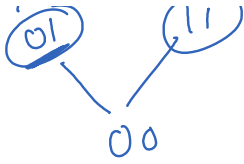


00

01

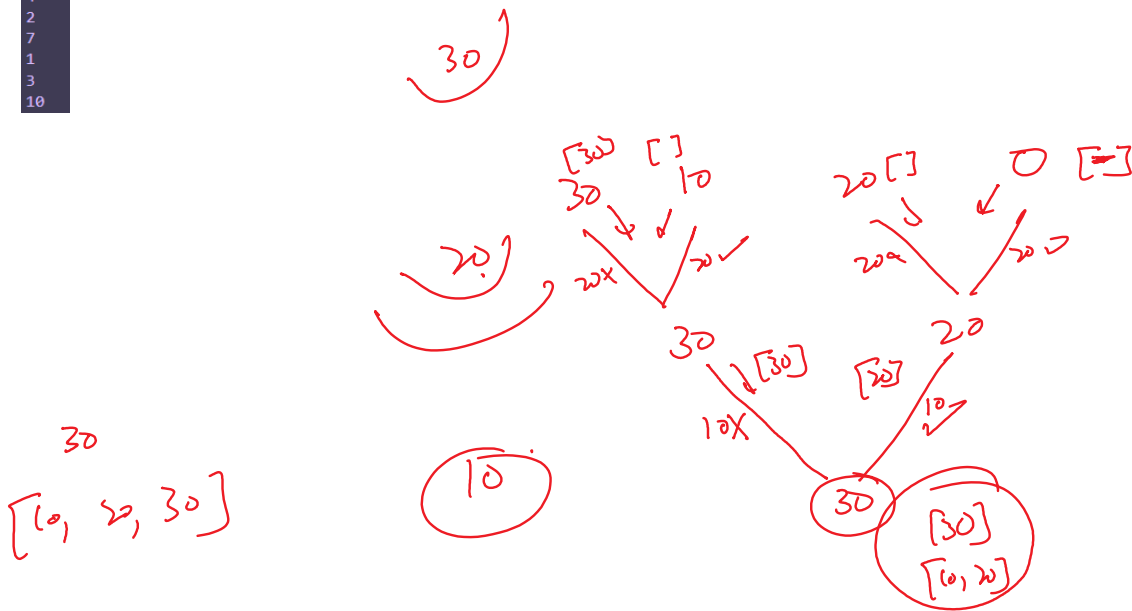
02





00

5
4
2
7
1
3
10



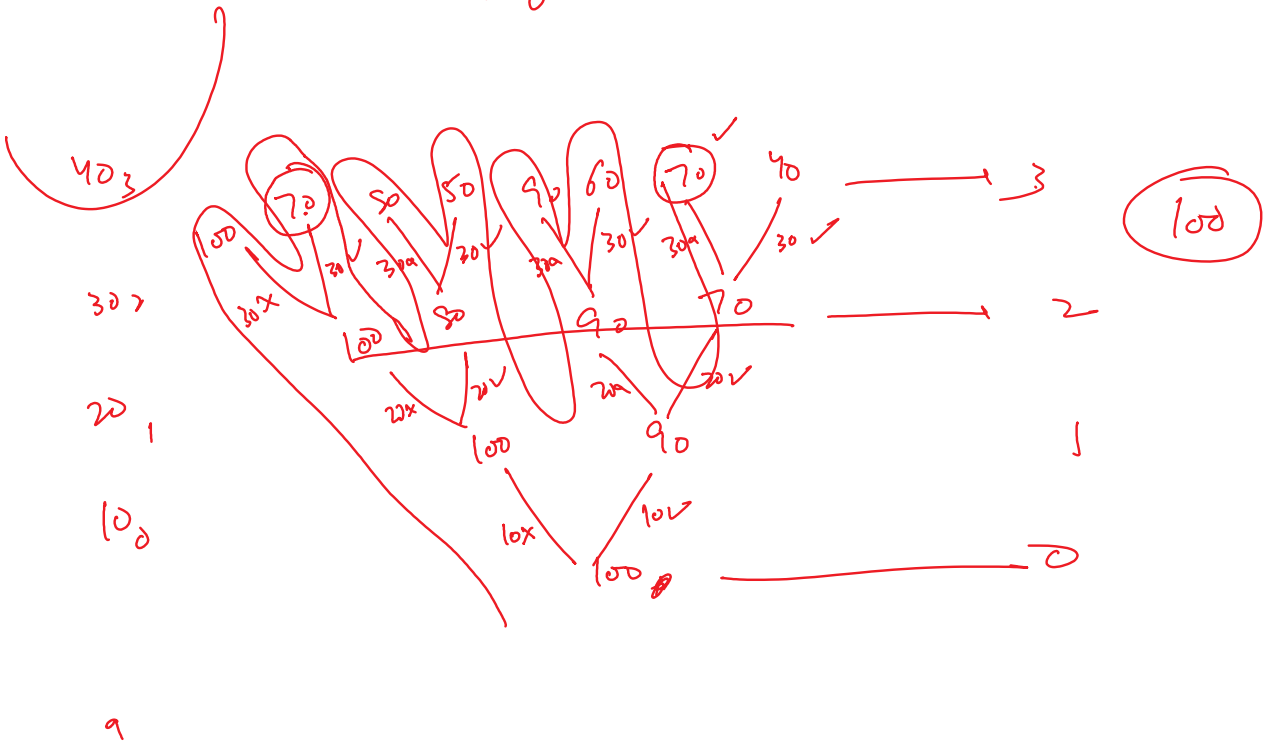
```
ArrayList<String> excluders = giveAllSubsetsOfSumX(arr, idx + 1, tar);
```

```
ArrayList<String> excluders = giveAllSubsetsOfSumX(arr, idx + 1, tar);
ArrayList<String> includers = giveAllSubsetsOfSumX(arr, idx + 1, tar - arr[idx]);

ArrayList<String> myRes = new ArrayList<String>();
myRes.addAll(excluders);
for(String set: includers){
    myRes.add(set + " " + arr[idx]);
}
```

3
1
7
2
4

$[7, 3]$
 $[2, 7, 1]$
 $[4, 2, 1, 3]$

$$\text{sty}[70][3] = \text{---}$$


Handwritten notes on a grid:

Grid columns: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

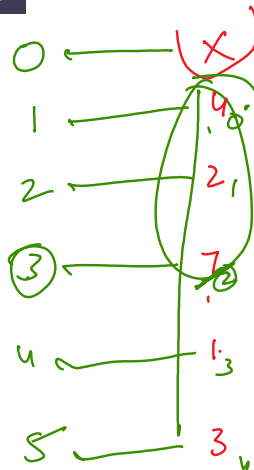
Grid rows: 0, 1, 2, 3, 4

Handwritten marks:

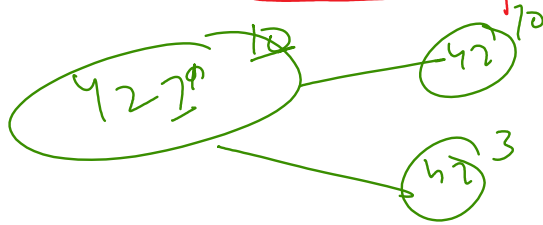
- Row 0: Column 0 has a circled 'v' and a green arrow pointing to it from the left. Column 10 is crossed out with a green line.
- Row 1: Column 1 has a circled 'x' and a red arrow pointing to it from the left. Column 10 is crossed out with a green line.
- Row 2: Column 2 has a circled 'x' and a red arrow pointing to it from the left. Column 10 is crossed out with a green line.
- Row 3: Column 3 has a circled 'x' and a red arrow pointing to it from the left. Column 10 is crossed out with a green line.
- Row 4: Column 4 has a circled 'x' and a red arrow pointing to it from the left. Column 10 is crossed out with a green line.

Additional notes:

- A green arrow points from the left towards the grid, specifically towards the circled 'v' in row 0, column 0.
- A red arrow points from the left towards the grid, specifically towards the circled 'x' in row 1, column 1.
- A red arrow points from the left towards the grid, specifically towards the circled 'x' in row 2, column 2.
- A red arrow points from the left towards the grid, specifically towards the circled 'x' in row 3, column 3.
- A red arrow points from the left towards the grid, specifically towards the circled 'x' in row 4, column 4.



0	✓	X	X	X	X	X	X	X	X	X	X
1	✓	X	X	X	✓	X	X	X	X	X	X
2	✓	X	X	X	✓	X	X	X	X	X	X
3	✓	X	X	X	✓	X	X	X	X	X	X
4	✓	X	X	X	✓	X	X	X	X	X	X
5	✓	X	X	X	✓	X	X	X	X	X	X



```

strg[i][j] = strg[i - 1][i];
int val = arr[i - 1];

if(j >= val && strg[i - 1][j - val]){
    strg[i][j] = true;
}

```

7 DP = R, M, T

150 → Tabulation

Videos Watchy is hw

9-12

1000+ times

Recursion

1. TSS	4. OIRS
2. CCC	5. UBRs
3. CCP	6. FRS

Direction

Tabulation → Topological Sort