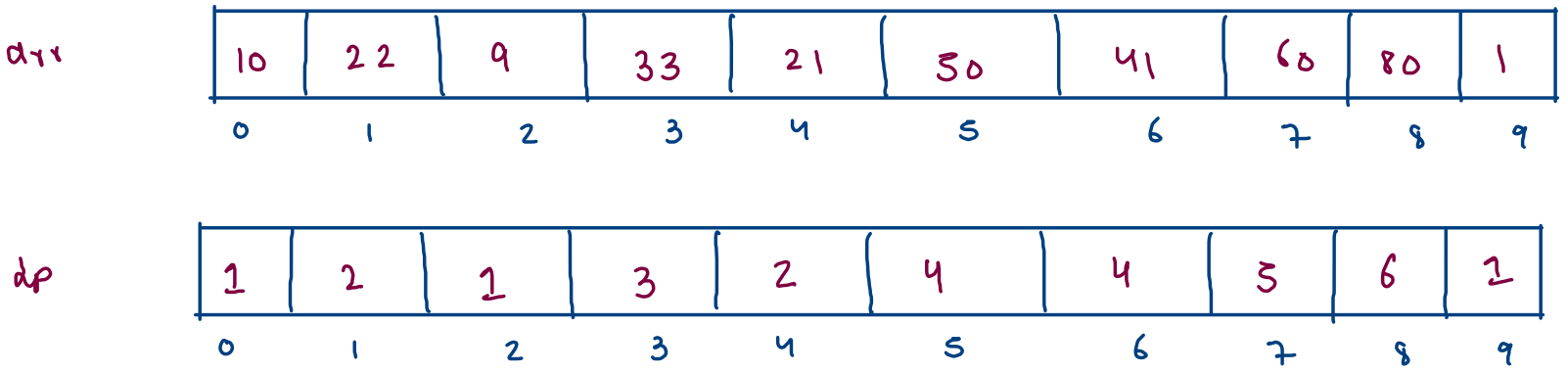
Print All Longest Increasing Subsequences



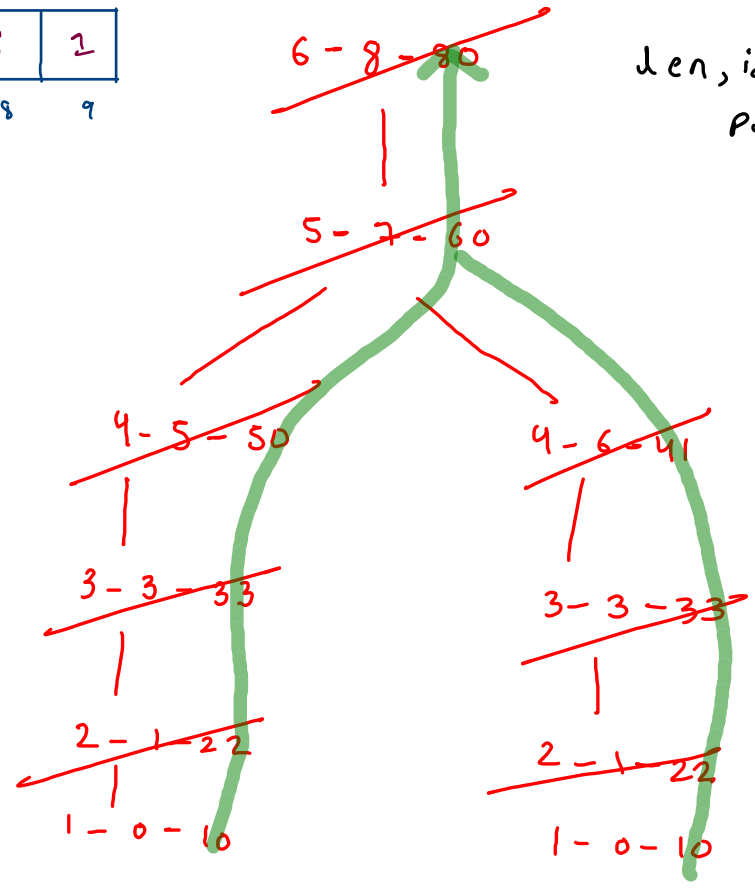

```
10
10 22 9 33 21 50 41 60 80 1
```

arr

0	1	2	3	4	5	6	7	8	9
10	22	9	33	21	50	41	60	80	1

dp

0	1	2	3	4	5	6	7	8	9
1	2	1	3	2	4	4	5	6	1



len, idx, val
psf.

7 -> 15 -> 18

9 -> 15 -> 18

7 -> 15 -> 16

9 -> 15 -> 16

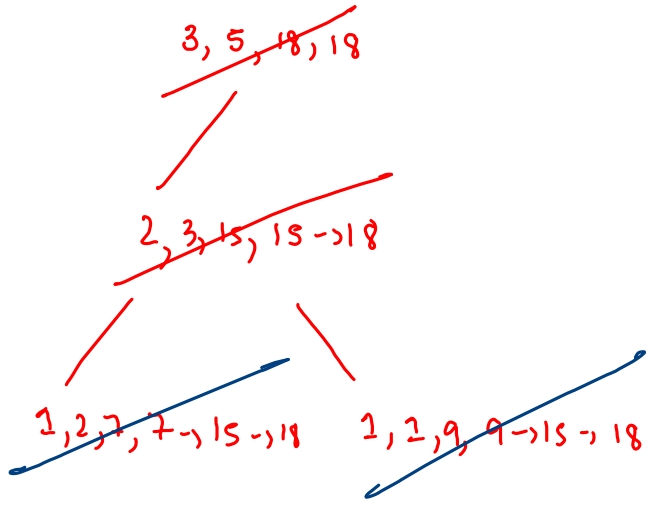
```
public static void printAllPath(int[]arr,int[]dp,int omax) {
    ArrayDeque<Pair>q = new ArrayDeque<>();

    for(int i=0; i < dp.length;i++) {
        if(dp[i] == omax) {
            q.add(new Pair(dp[i],i,arr[i],arr[i] + ""));
        }
    }

    while(q.size() > 0) {
        Pair rem = q.remove();

        if(rem.l == 1) {
            System.out.println(rem.psf);
            continue;
        }

        for(int j=rem.i-1; j >= 0;j--) {
            if(arr[j] < arr[rem.i] && dp[j] == dp[rem.i] - 1) {
                q.add(new Pair(dp[j],j,arr[j],arr[j] + " -> " + rem.psf));
            }
        }
    }
}
```

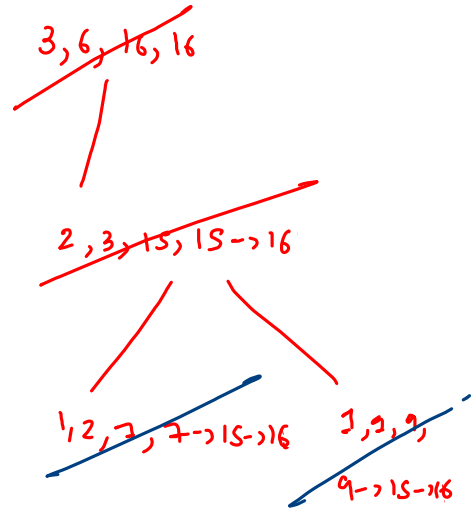
20	9	7	15	6	18	16
0	1	2	3	4	5	6

1	1	1	2	1	3	3
0	1	2	3	4	5	6

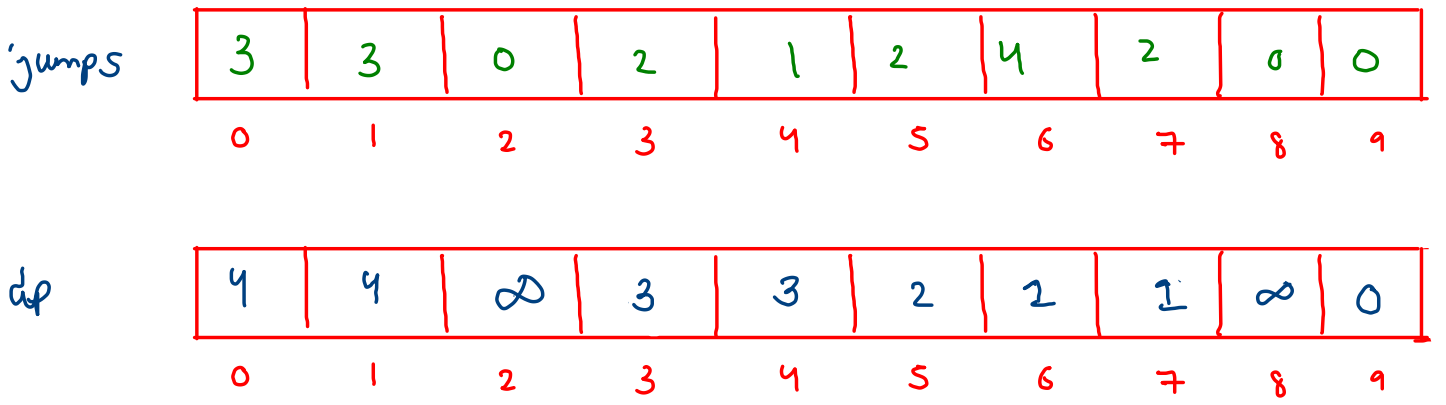
omax = 3

dem, i, val, psf





Print All Paths With Minimum Jumps

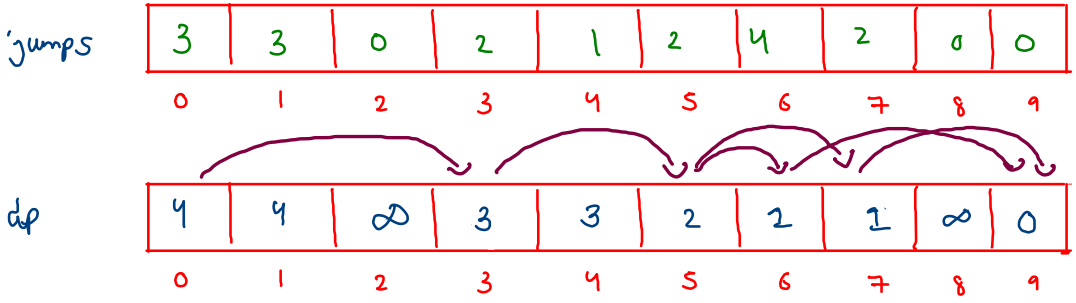


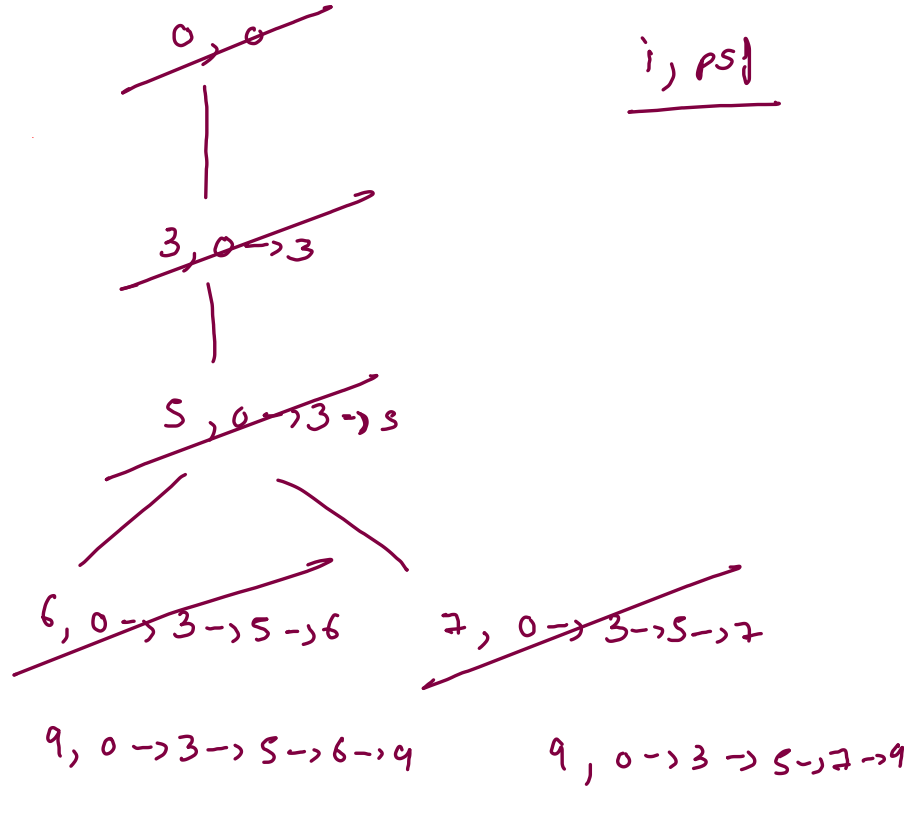
0 → 3 → 5 → 6 → 9

0 → 3 → 5 → 7 → 9

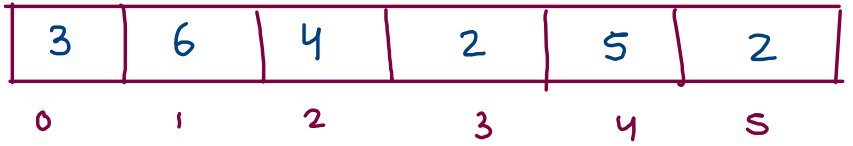
dp[i] → i to dest (n-1)

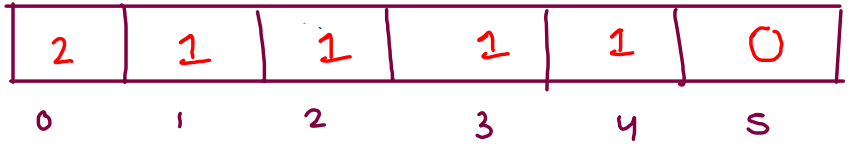
min 'jumps.


jumps
3 3 0 2 1 2 4 2 0 0
0 1 2 3 4 5 6 7 8 9
dp
4 4 ∞ 3 3 2 2 1 ∞ 0
0 1 2 3 4 5 6 7 8 9

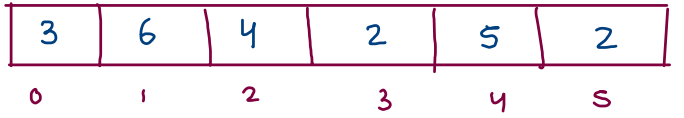

i, psf
0, 0
3, 0->3
5, 0->3->5
6, 0->3->5->6
7, 0->3->5->7
9, 0->3->5->6->9
9, 0->3->5->7->9

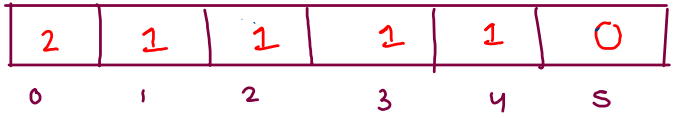
```java
for(int i = n-2; i >= 0;i--) {
    int min = Integer.MAX_VALUE;

    for(int j = 1; j <= arr[i] && i + j < n;j++) {
        if(dp[i + j] < min) {
            min = dp[i + j];
        }
    }

    if(min == Integer.MAX_VALUE) {
        dp[i] = min;
    }
    else {
        dp[i] = min + 1;
    }
}
```

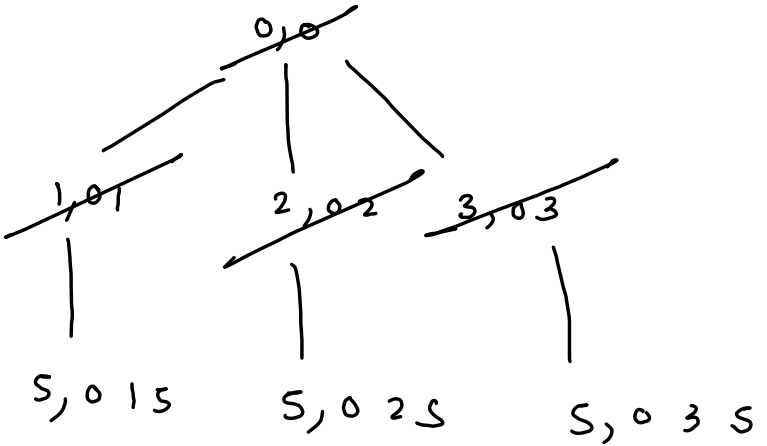




```java
public static void printAllPaths(int[]arr,int[]dp) {
    ArrayDeque<Pair>q = new ArrayDeque<>();

    int n = arr.length;

    q.add(new Pair(0,"0"));

    while(q.size() > 0) {
        Pair rem = q.remove();
        int i = rem.i;

        if(i == n-1) {
            System.out.println(rem.psf + " .");
            continue;
        }

        for(int j = 1; j <= arr[i] && i + j < n;j++) {
            if(dp[i + j] == dp[i] - 1) {
                q.add(new Pair(i + j,rem.psf + " -> " + (i + j)));
            }
        }
    }
}
```







Print All Results In 0-1 Knapsack

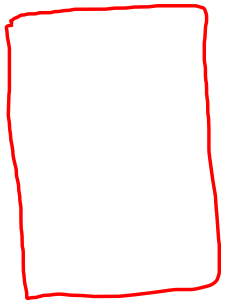
n = 5

price : 15 14 10 45 30

wt : 2 5 1 3 4

```
5
15 14 10 45 30
2 5 1 3 4
7
```

cap = 7



price : 15 14 10 45 30

wt : 2 5 1 3 4

```
else {
    int exc = dp[i-1][j];
    int inc = 0;

    int k = i-1; //item's idx
    if(j >= wt[k]) {
        inc = dp[i-1][j - wt[k]] + value[k];
    }

    dp[i][j] = Math.max(inc,exc);
}
```

		0	1	2	3	4	5	6	7
0	X	0	0	0	0	0	0	0	0
1	$(2-15)_0$	0	0	15	15	15	15	15	15
2	$(5-14)_1$	0	0	15	15	15	15	15	29
3	$(1-10)_2$	0	10	15	25	25	25	25	29
4	$(3-45)_3$	0	10	15	45	55	60	70	70
5	$(4-30)_4$	0	10	15	45	55	60	70	75

i = 4

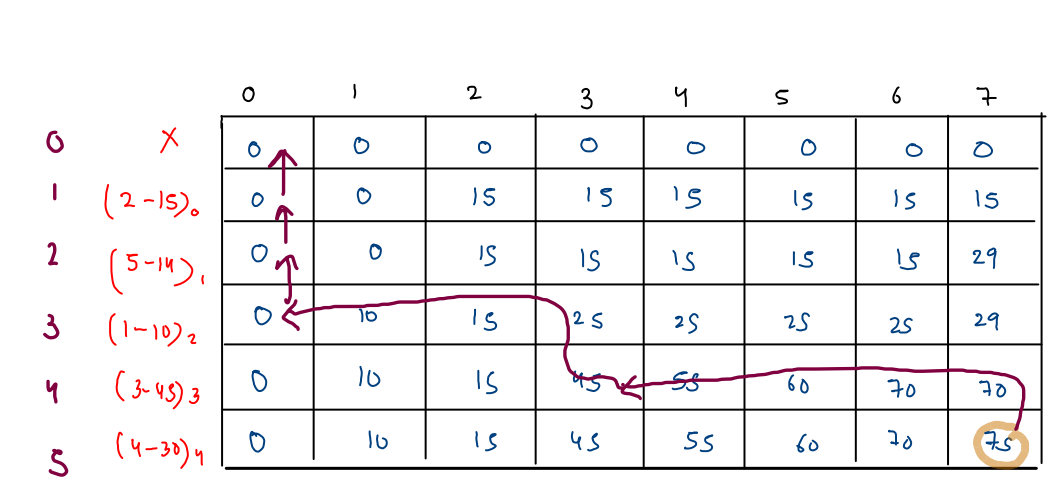
j = 5

k = 3

exc = 25

inc = 15+45 → 60

		0	1	2	3	4	5	6	7
0	X	0	0	0	0	0	0	0	0
1	$(2,15)_0$	0	0	15	15	15	15	15	15
2	$(5,14)_1$	0	0	15	15	15	15	15	29
3	$(1,10)_2$	0	10	15	25	25	25	25	29
4	$(3,45)_3$	0	10	15	45	55	60	70	70
5	$(4,30)_4$	0	10	15	45	55	60	70	75



5, 7, 0

4, 3, 4

3, 0, 4-3

i, j

row no.

col no.

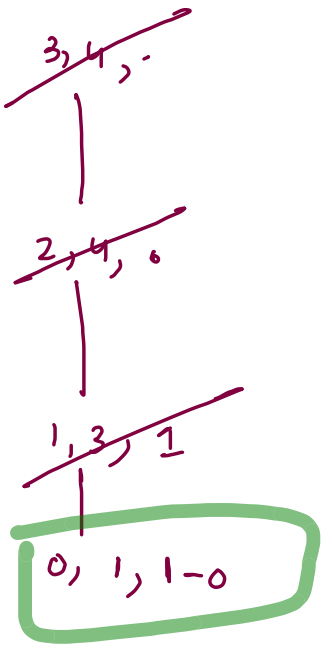
j -> col no. -> cap

i -> row no. -> (i-1) -> item no

wt	2	1	3
value	15	30	8

		0	1	2	3	4
0	X	0	0	0	0	0
1	2-15$_0$	0	0	15	15	15
2	1-30$_1$	0	30	30	45	45
3	3-8$_2$	0	38	30	45	45

cap = 4



n = 4

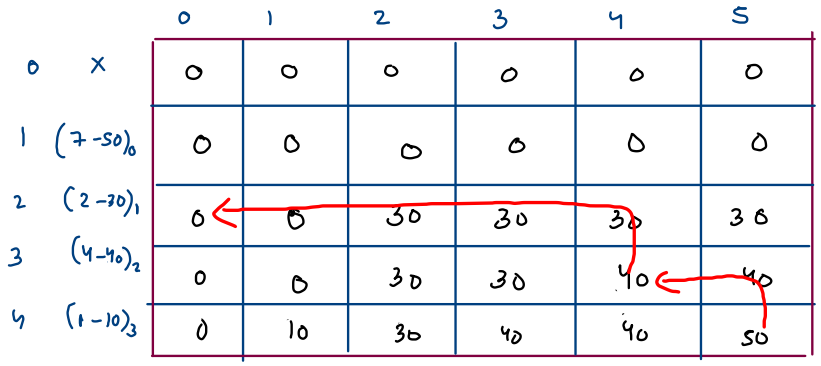
cap = 5

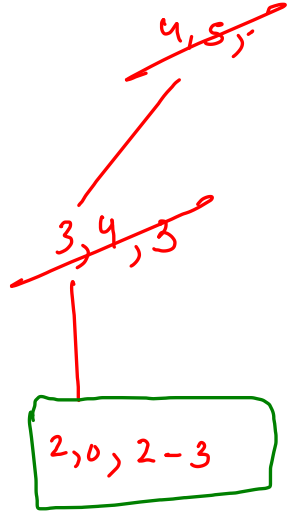
	0	1	2	3
value :	50	30	35	15
wt :	7	2	4	1

```java
for(int i=0; i < dp.length;i++) {
    for(int j=0; j < dp[0].length;j++) {
        if(i == 0) {
            //no item
            dp[i][j] = 0;
        }
        else if(j == 0) {
            //no capacity
            dp[i][j] = 0;
        }
        else {
            int exc = dp[i-1][j];
            int inc = 0;

            int k = i-1; //item's idx
            if(j >= wt[k]) {
                inc = dp[i-1][j - wt[k]] + value[k];
            }

            dp[i][j] = Math.max(inc,exc);
        }
    }
}
```

		0	1	2	3	4	5
0	X	0	0	0	0	0	0
1	$(7-50)_0$	0	0	0	0	0	0
2	$(2-30)_1$	0	0	30	30	30	30
3	$(4-40)_2$	0	0	30	30	40	40
4	$(1-10)_3$	0	10	30	40	40	50

```java
public static void printAllPaths(int[][]dp,int[]value,int[]wt) {
    int n = value.length;
    int cap = dp[0].length - 1;

    ArrayDeque<Pair>q = new ArrayDeque<>();

    q.add(new Pair(n,cap,""));

    while(q.size() > 0) {
        Pair rem = q.remove();
        int i = rem.i;
        int j = rem.j;
        String psf = rem.psf;


        if(j == 0 || i == 0) {
            System.out.println(psf);
            continue;
        }

        int exc = dp[i-1][j];
        int inc = 0;

        int k = i-1; //item's idx
        if(j >= wt[k]) {
            inc = dp[i-1][j - wt[k]] + value[k];
        }

        if(dp[i][j] == exc) {
            q.add(new Pair(i-1,j,psf));
        }
        if(dp[i][j] == inc) {
            if(j >= wt[k]) {
                q.add(new Pair(i-1,j - wt[k],k + " " + psf));
            }
        }

    }
}
```

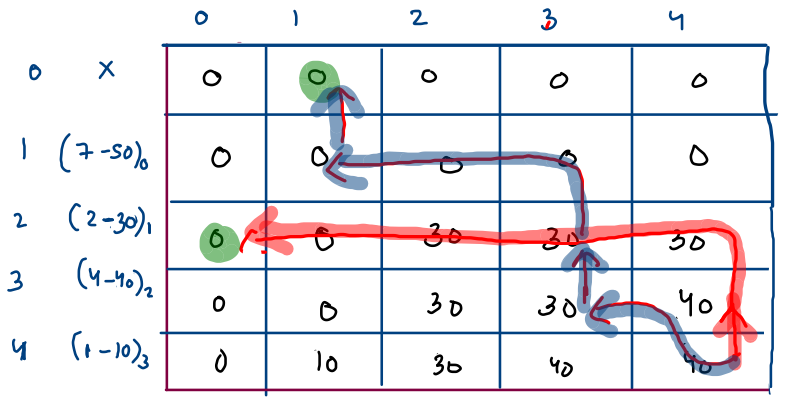


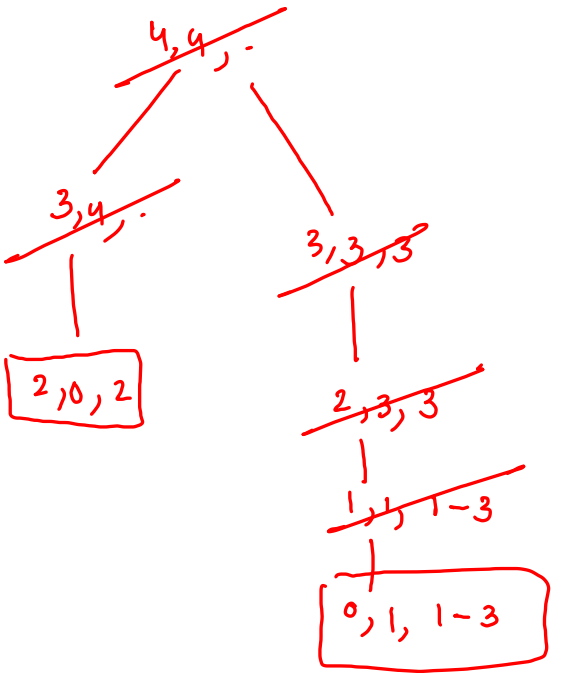


exc = 30

inc = 0+40

```java
public static void printAllPaths(int[][]dp,int[]value,int[]wt) {
    int n = value.length;
    int cap = dp[0].length - 1;

    ArrayDeque<Pair>q = new ArrayDeque<>();

    q.add(new Pair(n,cap,""));

    while(q.size() > 0) {
        Pair rem = q.remove();
        int i = rem.i;
        int j = rem.j;
        String psf = rem.psf;


        if(j == 0 || i == 0) {
            System.out.println(psf);
            continue;
        }

        int exc = dp[i-1][j];
        int inc = 0;

        int k = i-1; //item's idx
        if(j >= wt[k]) {
            inc = dp[i-1][j - wt[k]] + value[k];
        }

        if(dp[i][j] == exc) {
            q.add(new Pair(i-1,j,psf));
        }
        if(dp[i][j] == inc) {
            if(j >= wt[k]) {
                q.add(new Pair(i-1,j - wt[k],k + " " + psf));
            }
        }

    }
}
```





exc = 0

inc = 0