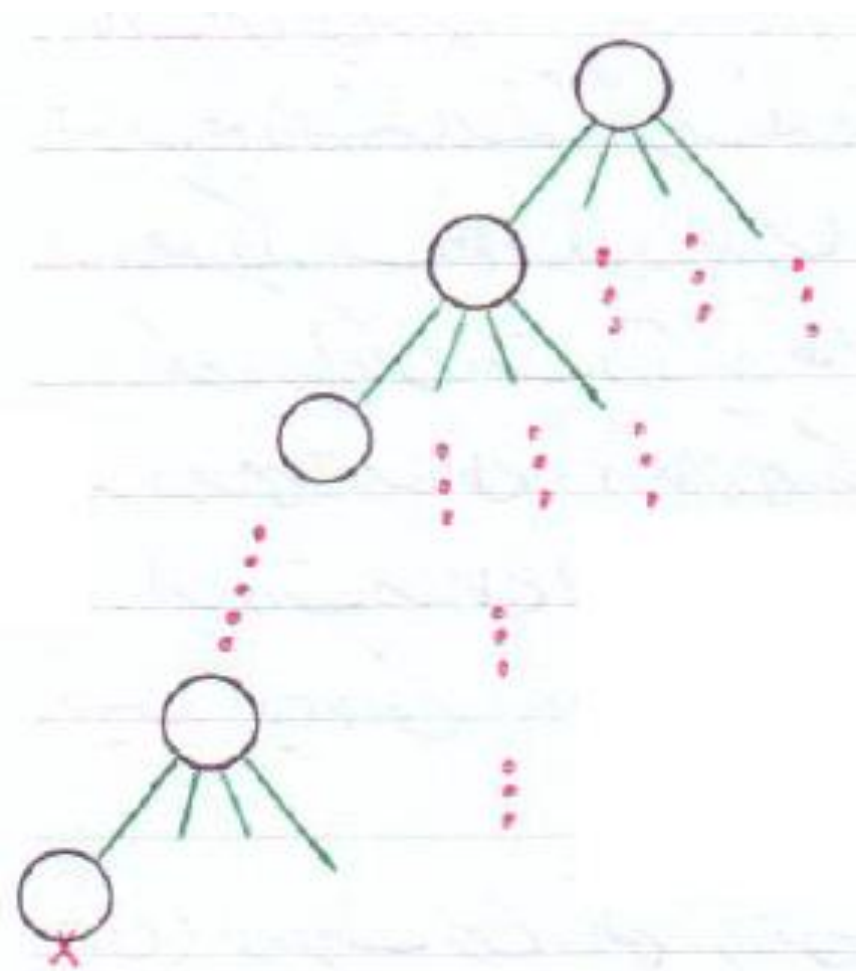


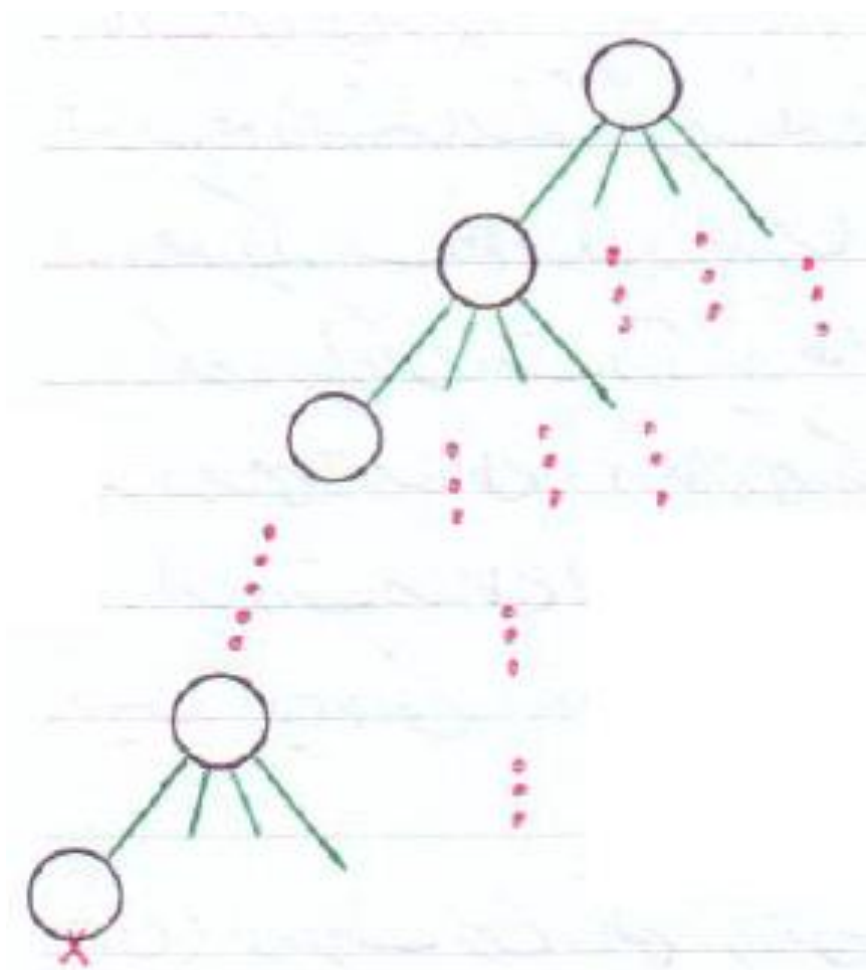
طراحی الگوریتم ها

روش عقبگرد

استاد درس: مهدی جبل عاملی

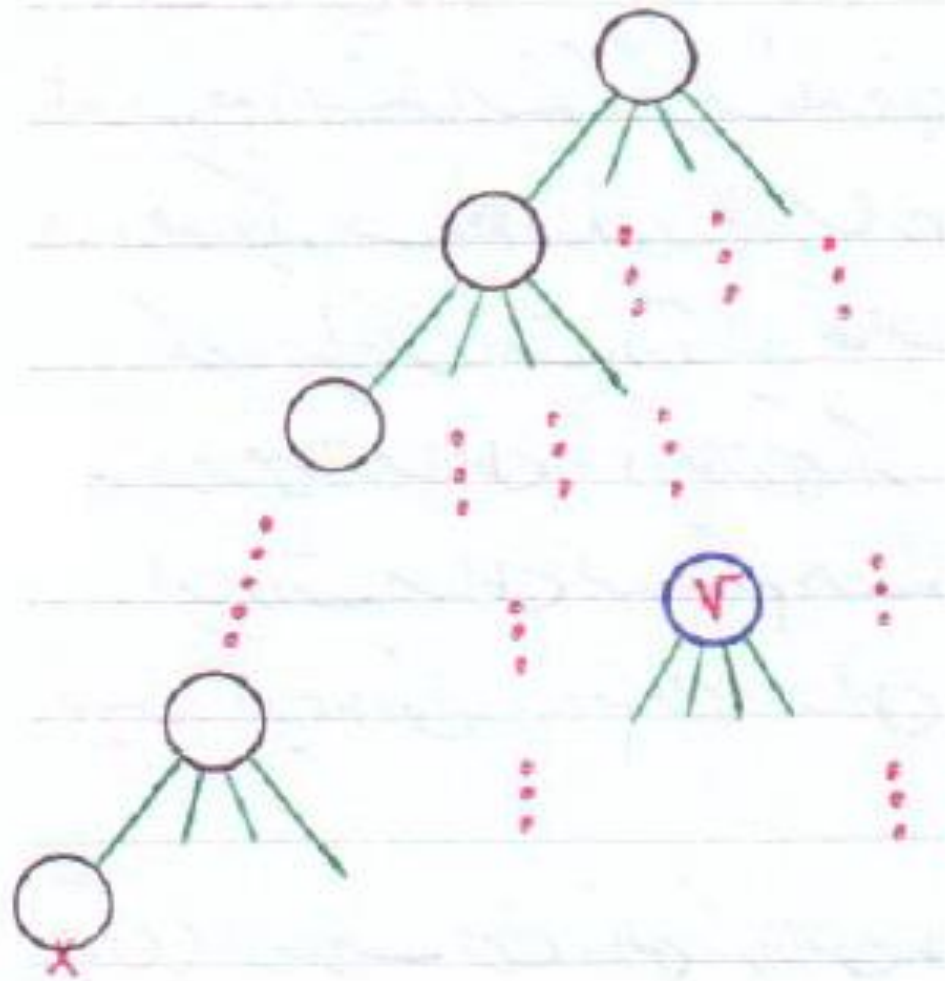






• درخت فضای حالت ←

# قالب کلی روش عقبگرد



Backtracking (node  $v$ ) {

if promising ( $v$ )

if (در گره  $v$  جوابی وجود دارد)

write solution;

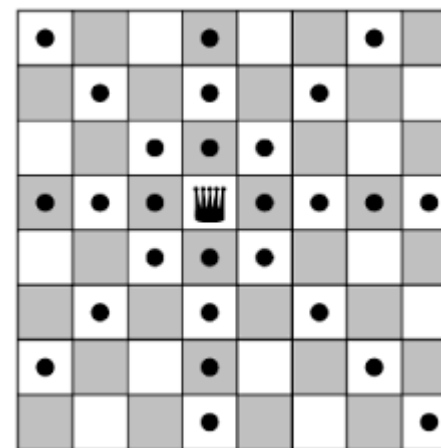
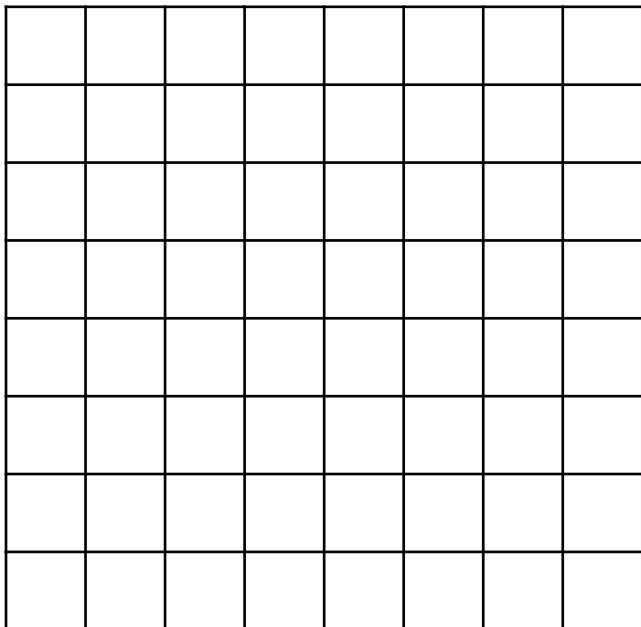
else

foreach child  $u$  of  $v$

Backtracking( $u$ );

}

# مساله وزیرها



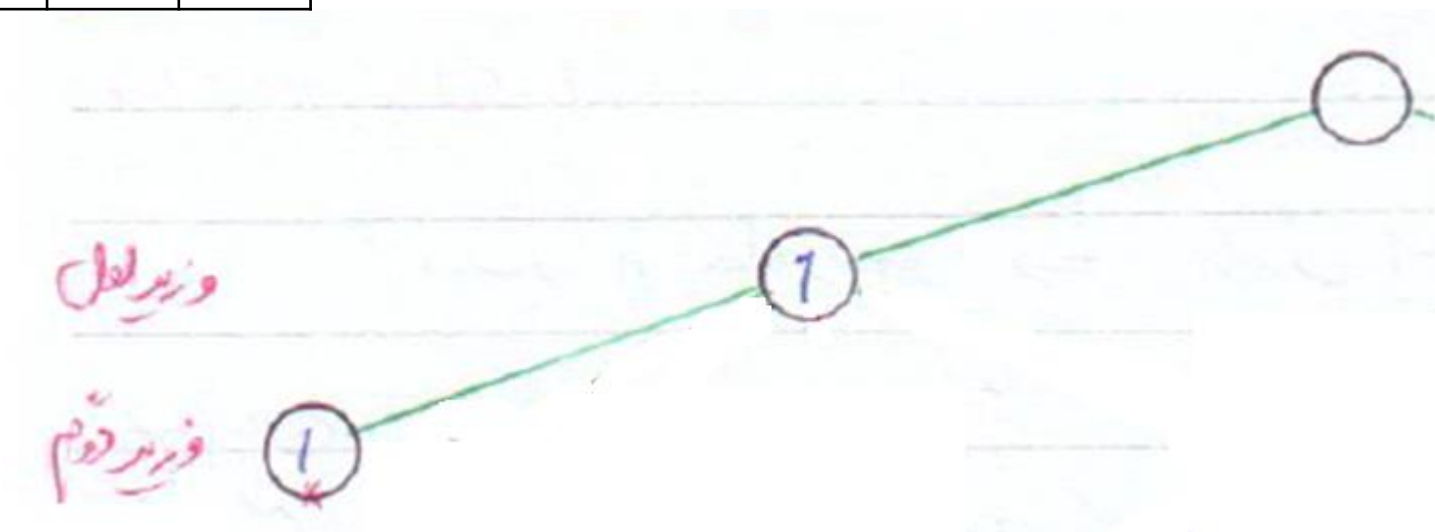
x			

حل مساله وزيرها ( $n=4$ )



x			

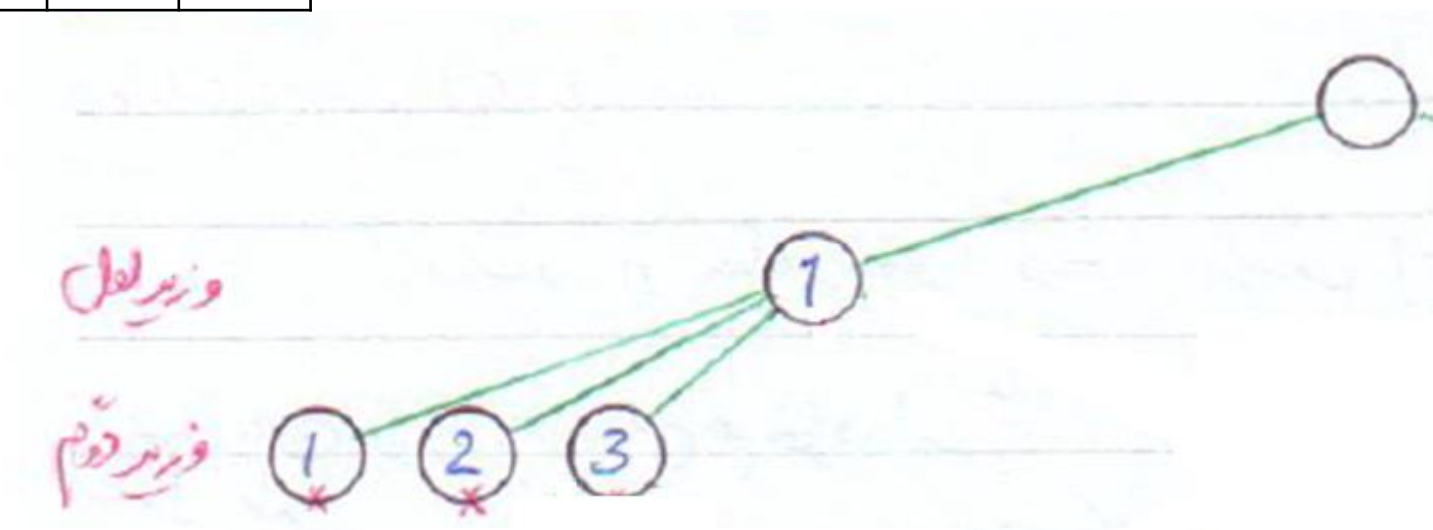
حل مساله وزیرها ( $n=4$ )





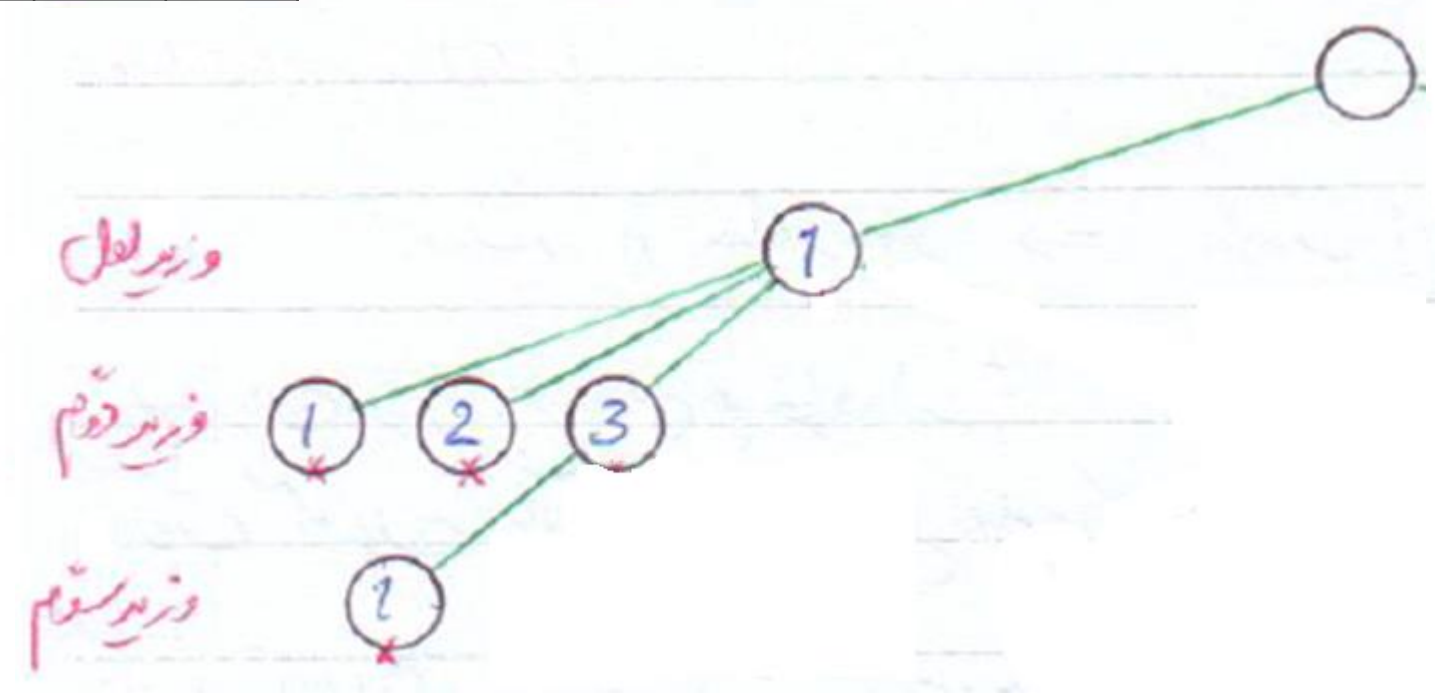
x			
		x	

حل مساله وزیرها ( $n=4$ )



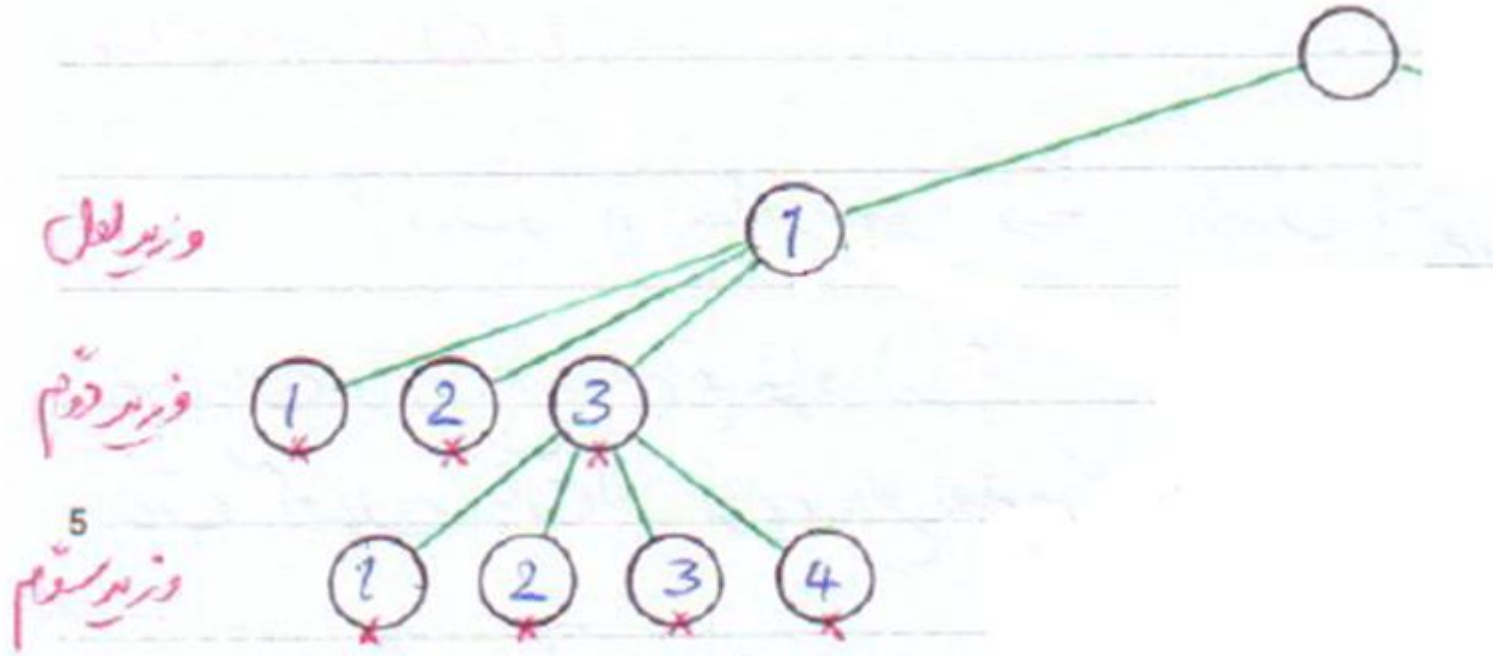
X			
		X	

حل مساله وزیرها (n=4)



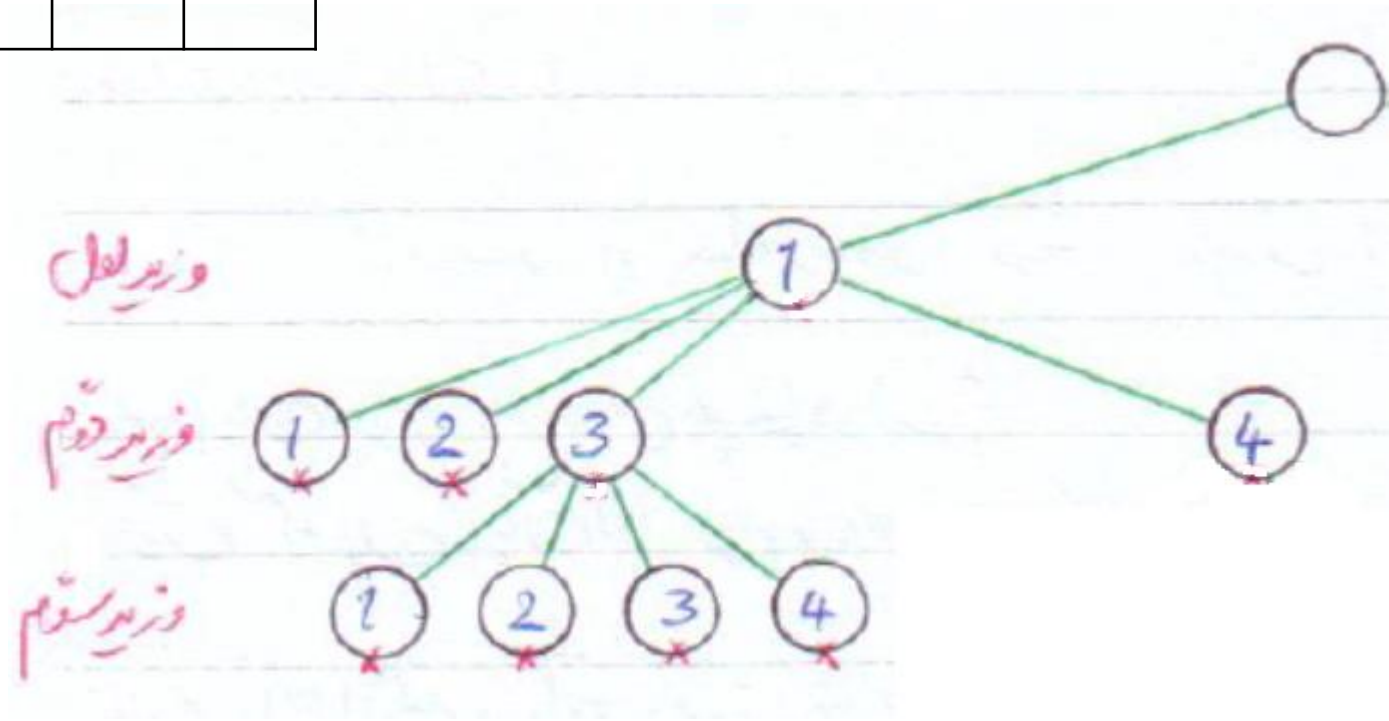
X			
		X	

حل مساله وزیرها (n=4)



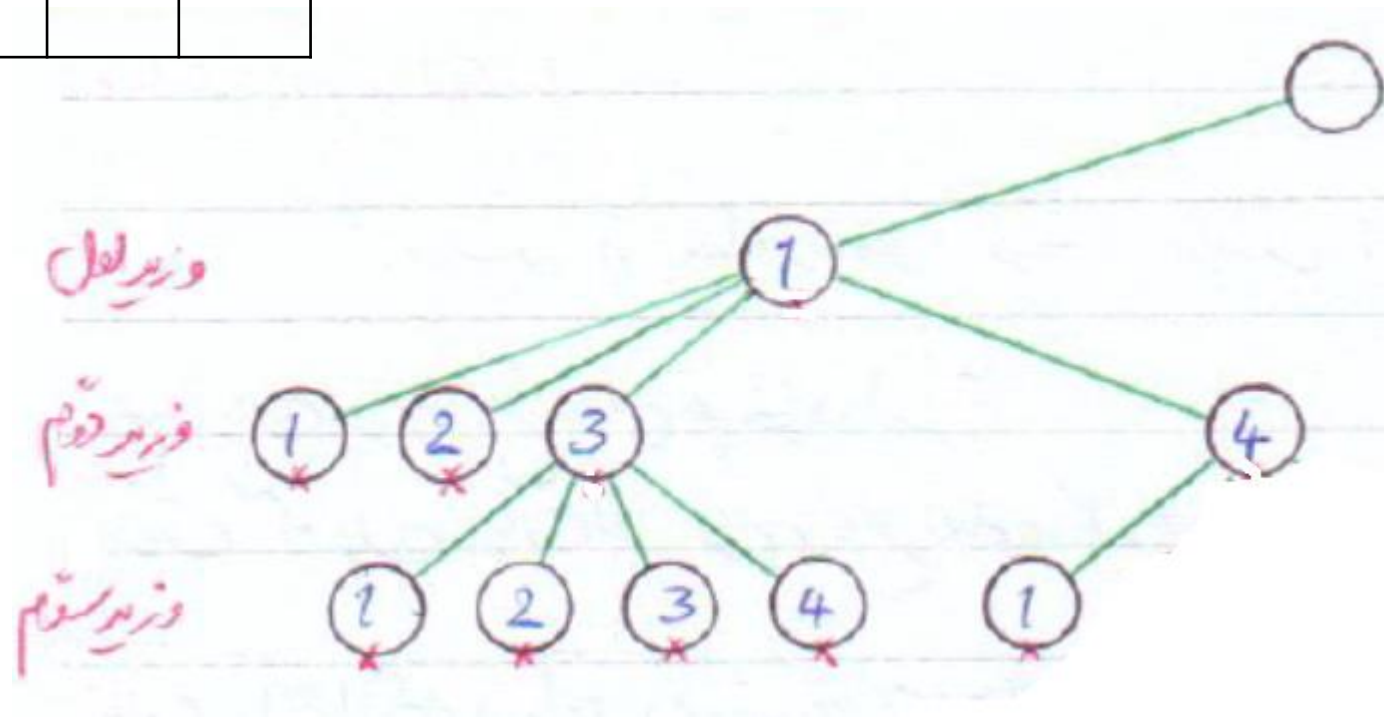
X			
			X

حل مساله وزیرها (n=4)



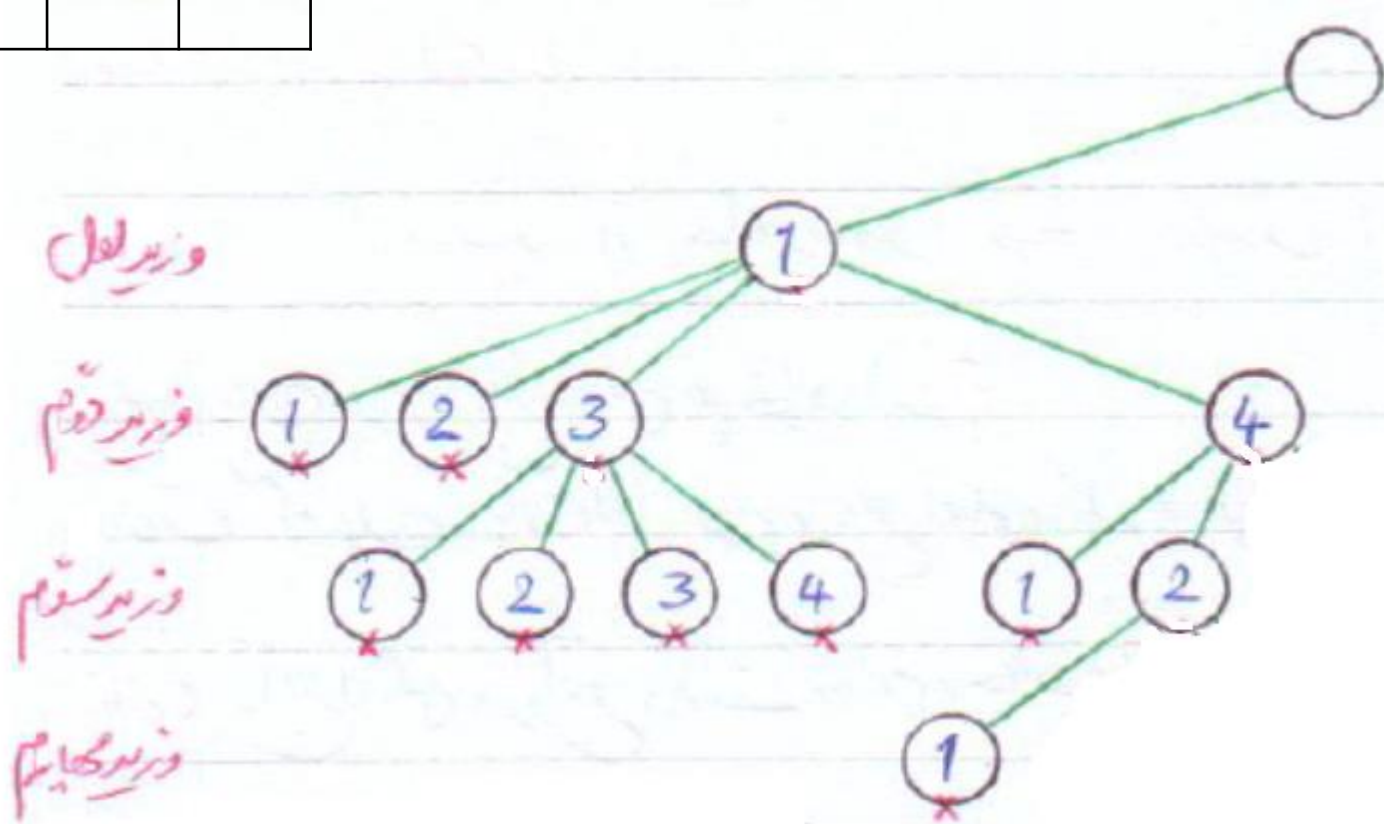
X			
			X

حل مساله وزیرها (n=4)



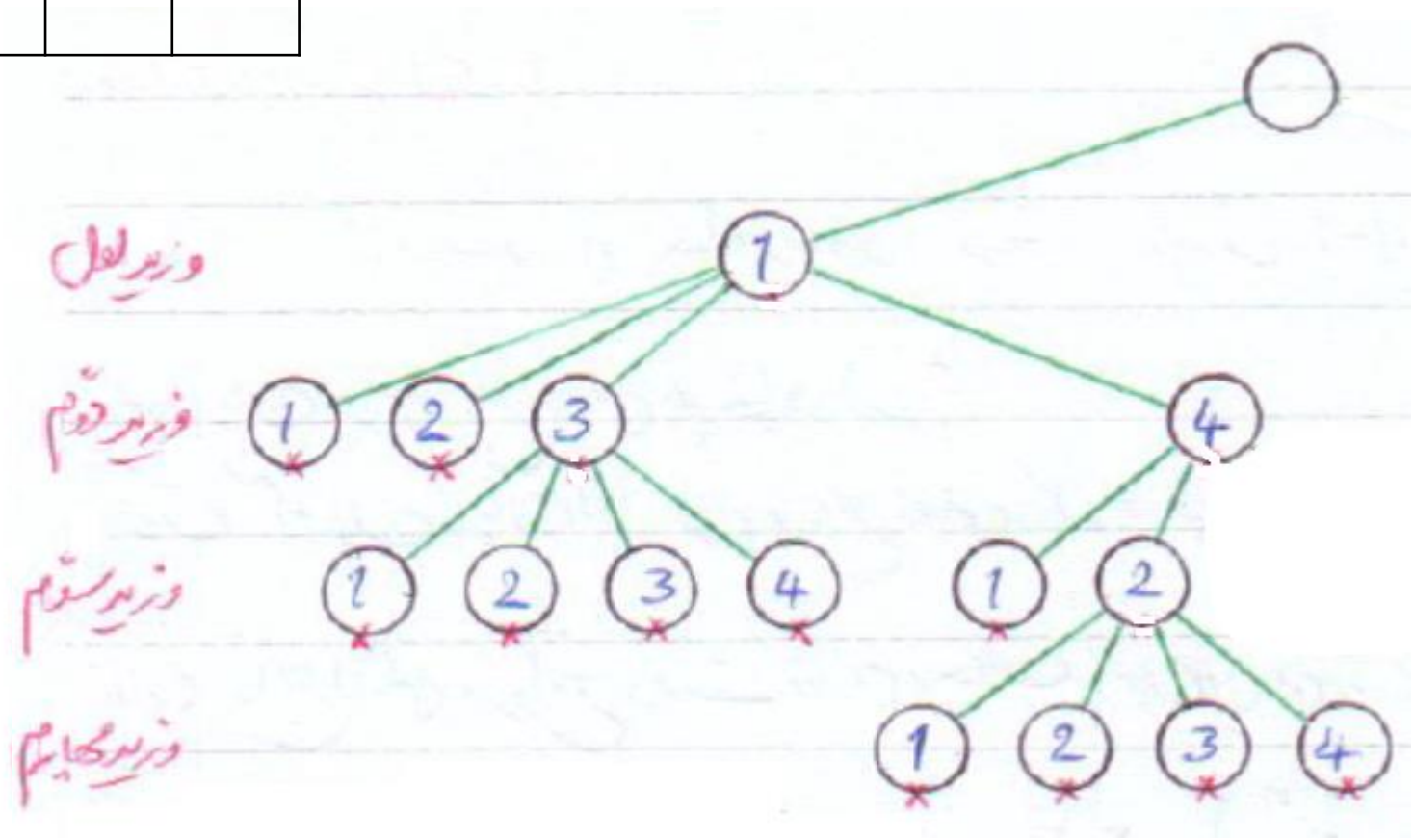
X			
			X
	X		

حل مساله وزیرها (n=4)



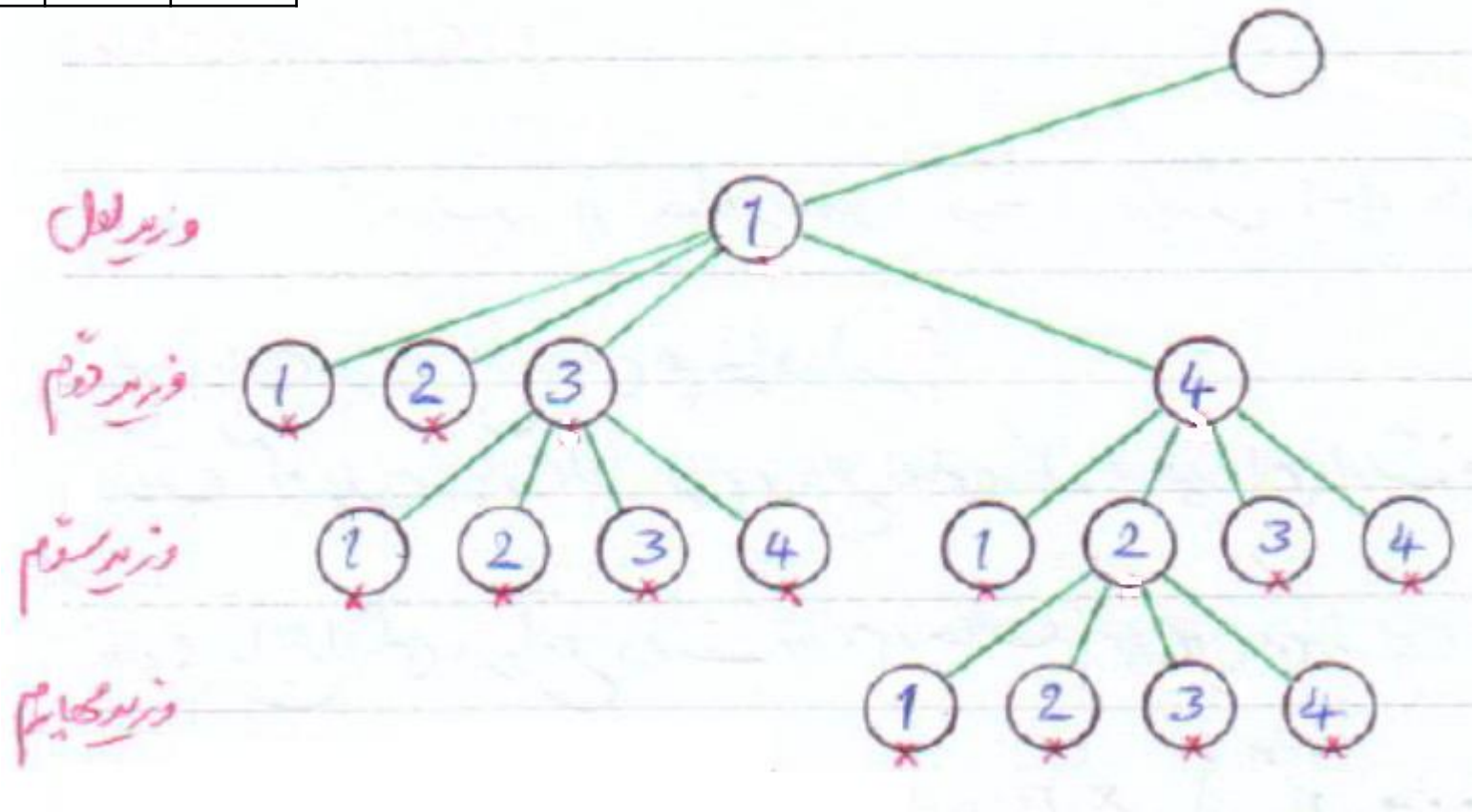
X			
			X
	X		

حل مساله وزیرها (n=4)



X			
			X

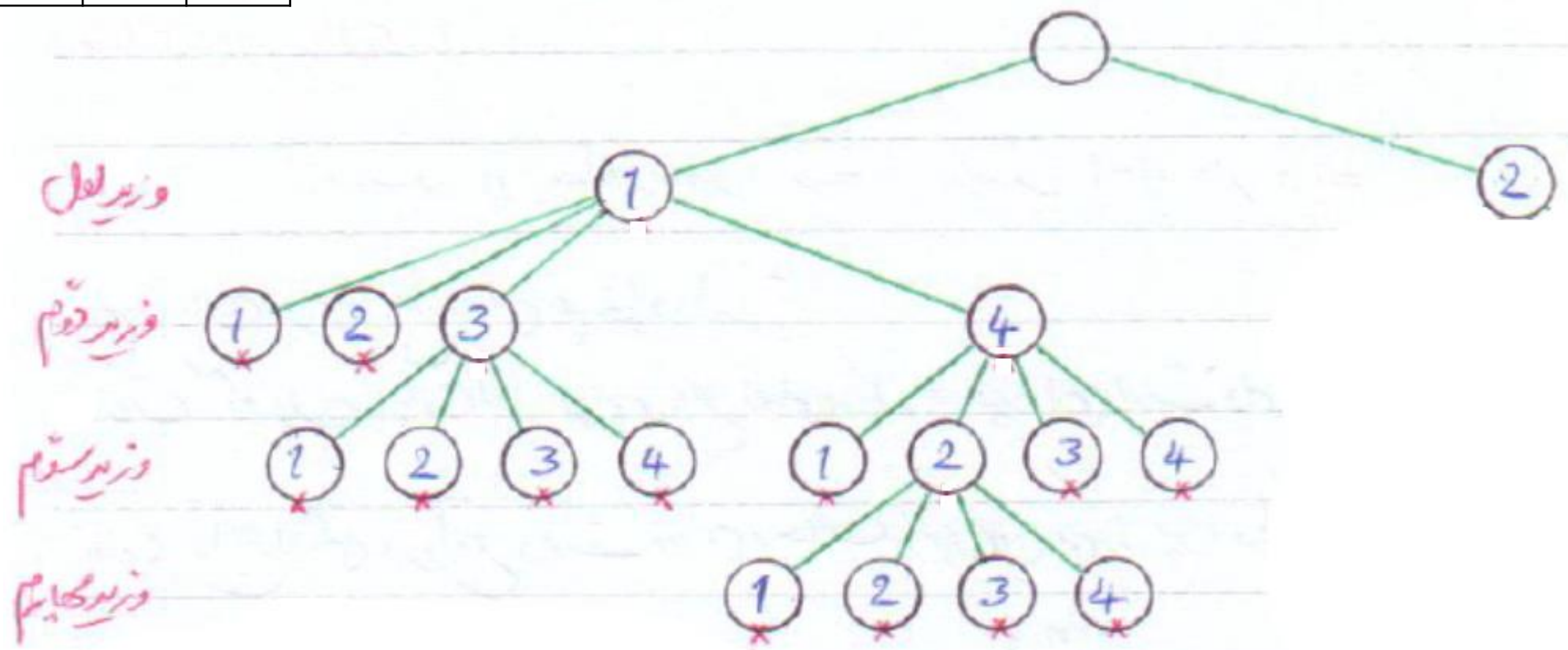
حل مساله وزیرها (n=4)





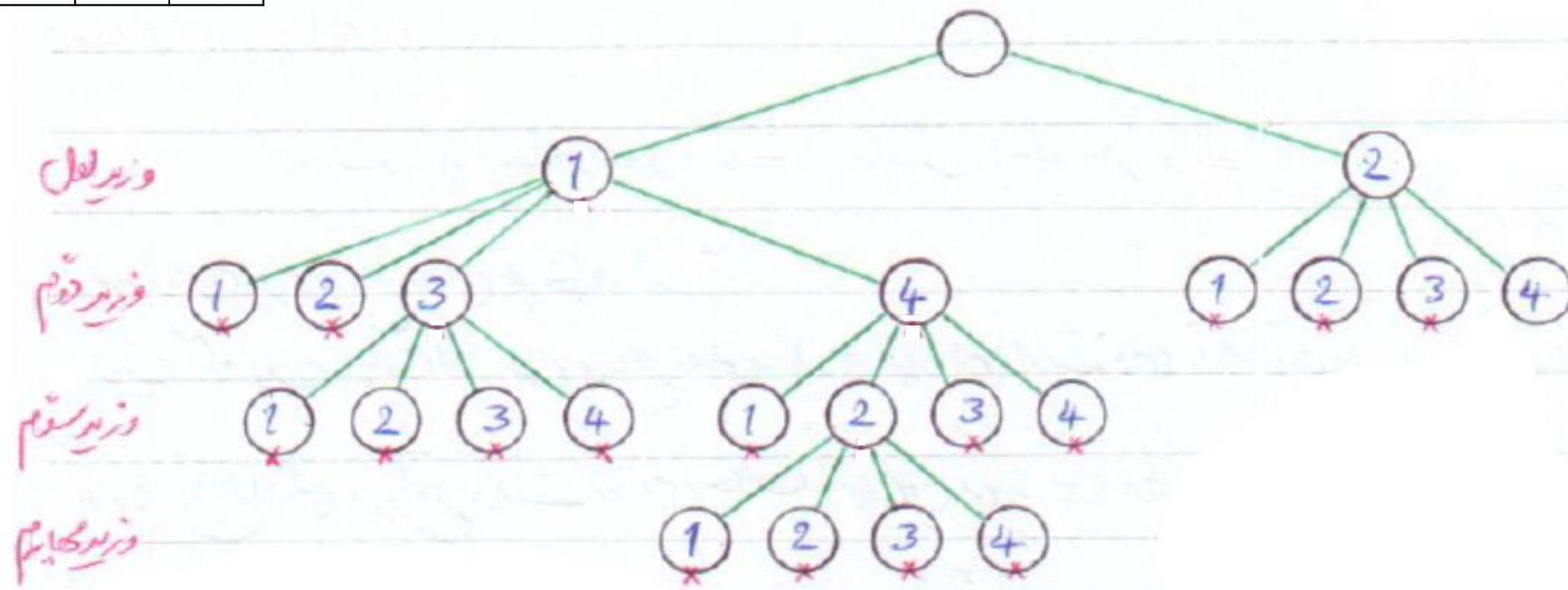
	X		

حل مساله وزیرها ( $n=4$ )



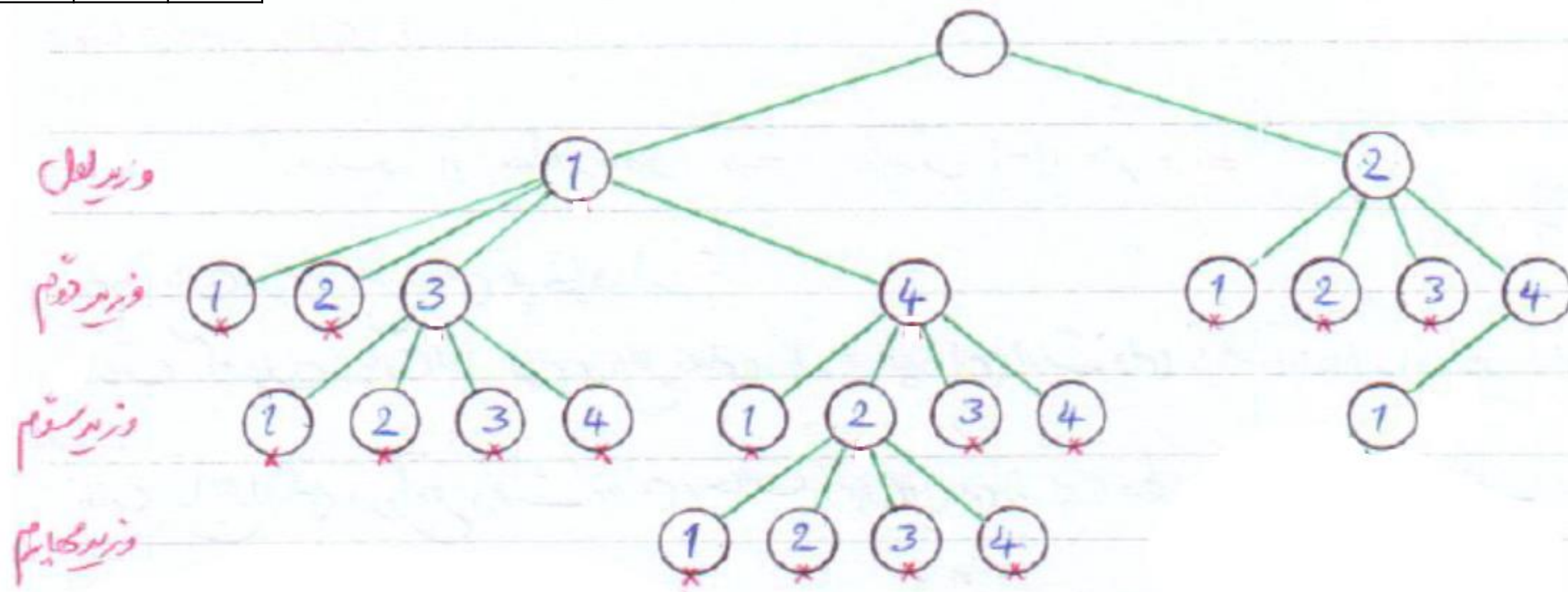
	X		
			X

حل مساله وزیرها ( $n=4$ )



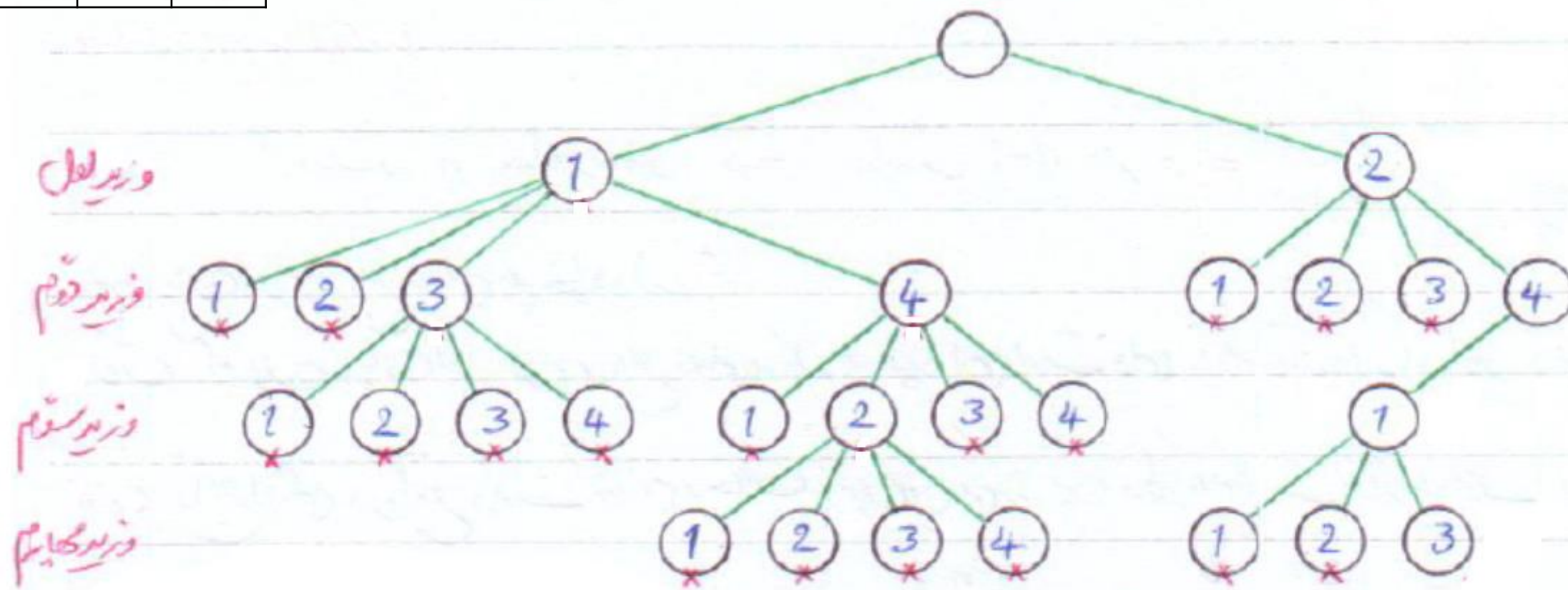
	X		
			X
X			

حل مساله وزیرها ( $n=4$ )



	X		
			X
X			
		X	

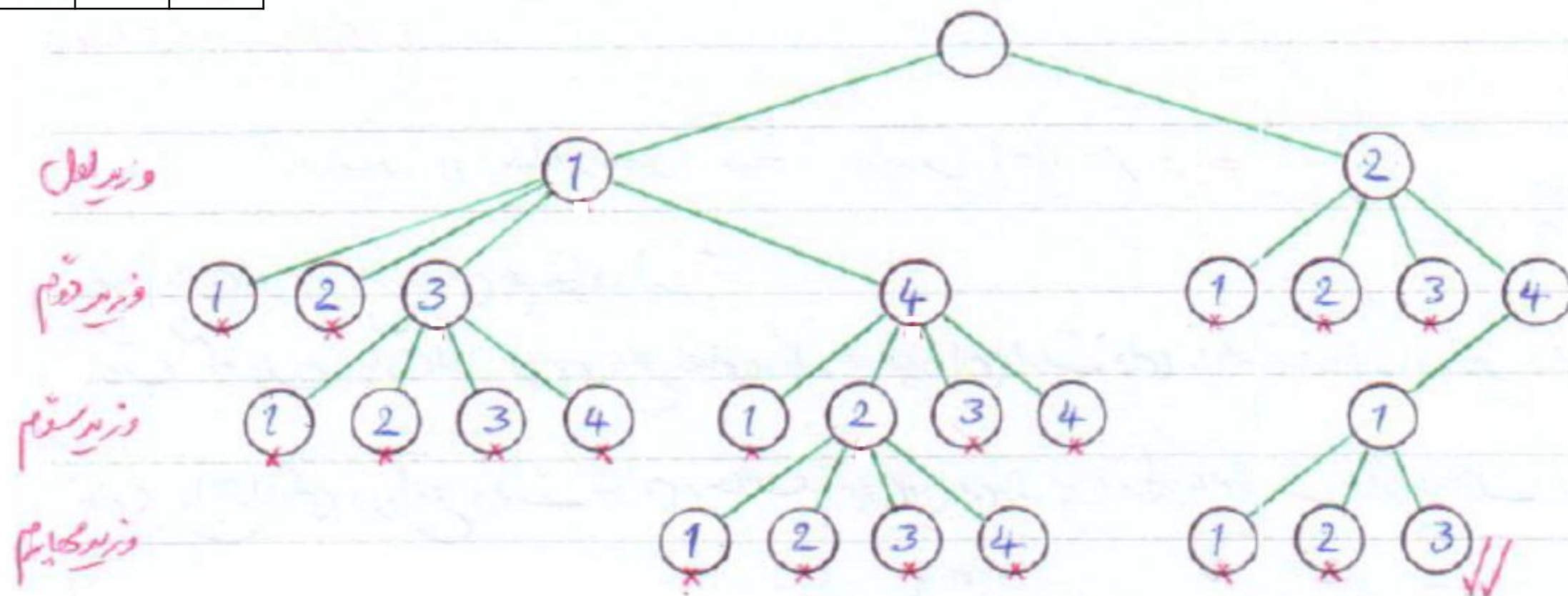
حل مساله وزیرها ( $n=4$ )





	X		
			X
X			
		X	

حل مساله وزیرها ( $n=4$ )



Queens (i) {

if Promising(i)

if (i = n)

write Col[1]..Col[n]

else

for j = 1 to n do

{ Col[i+1] = j;

Queens(i+1); }

}

Boolean Promising (i) {

for  $k=1$  to  $i-1$  do

if  $(col[i] = col[k] \text{ or}$

$i-k = |col[i] - col[k]|)$

return False;

return True;

}


سوال لکھا:

For promising  $\Rightarrow$   $n-1$  بار  $\Rightarrow$   $n$  بار



زمان اجرا:

for promising  $\Rightarrow$  در هر بار  $n-1$  بار  $\Rightarrow$  در مجموع  $n$  بار

تابع اصلی

$$(1 + n^1 + n^2 + n^3 + \dots + n^n) \times n$$

زمان اجرا:

for promising  $\Rightarrow$  مرتبه  $n-1$  بار  $\Rightarrow$  تعداد تکرار مرتبه  $n$

تابع اصلی

$$(1 + n^1 + n^2 + n^3 + \dots + n^n) \times n$$

$$= \left( \frac{n^{n+1} - 1}{n - 1} \right) \times n$$

زمان اجرا:

for promising  $\Rightarrow$  مرتبه  $n-1$  بار  $\Rightarrow$  مرتبه  $n$

تابع اصلی

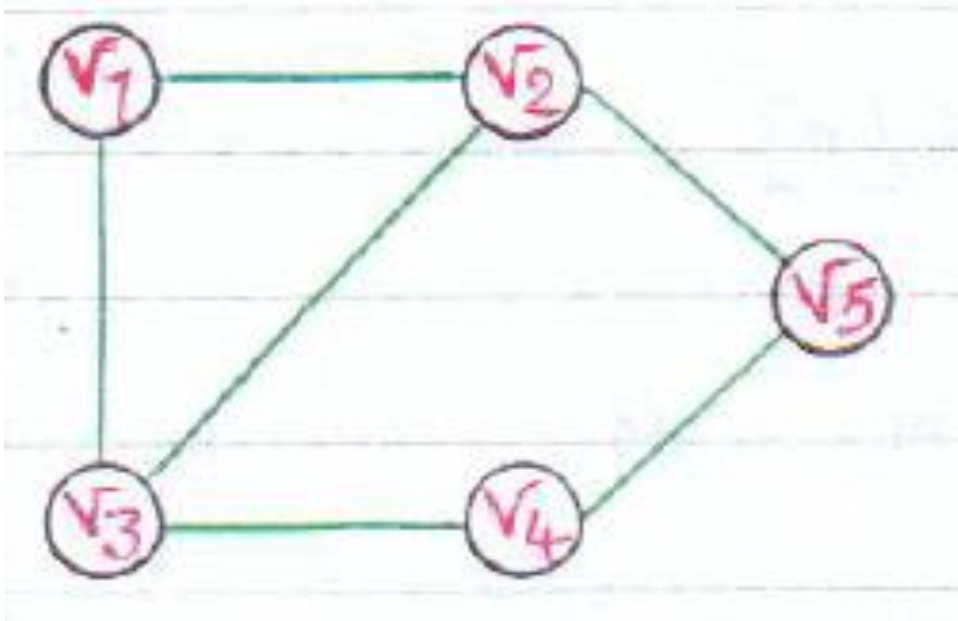
$$(1 + n^1 + n^2 + n^3 + \dots + n^n) \times n$$

$$= \left( \frac{n^{n+1} - 1}{n - 1} \right) \times n \in O(n^{n+1})$$

## تمرین

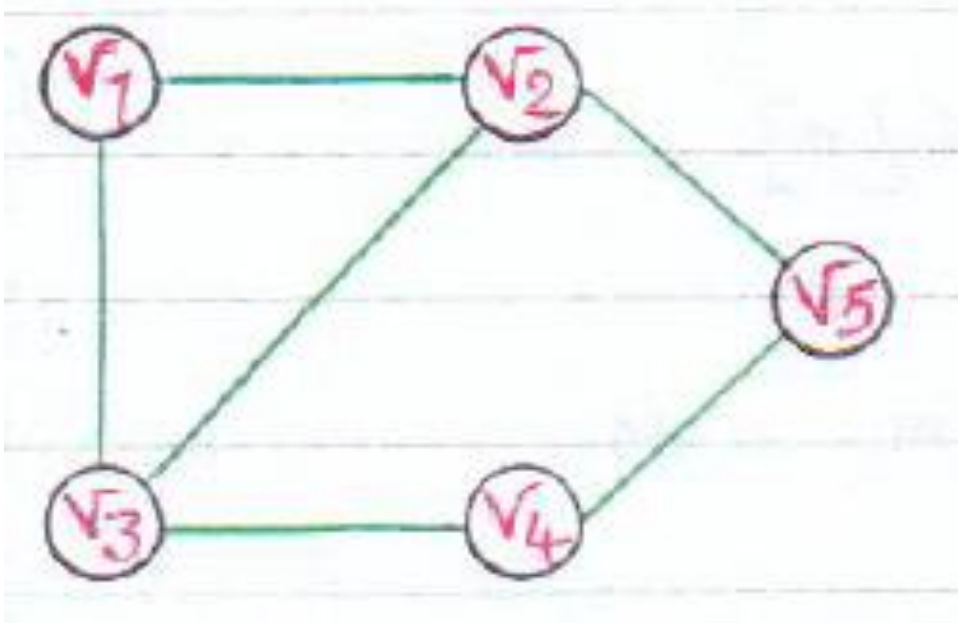
- الگوریتم وزیرها را به گونه ای پیاده سازی کنید که تعداد کل جواب های ممکنه و همچنین تعداد گره های بررسی شده از درخت فضای حالت را چاپ نماید.

# رنگ آمیزی گراف



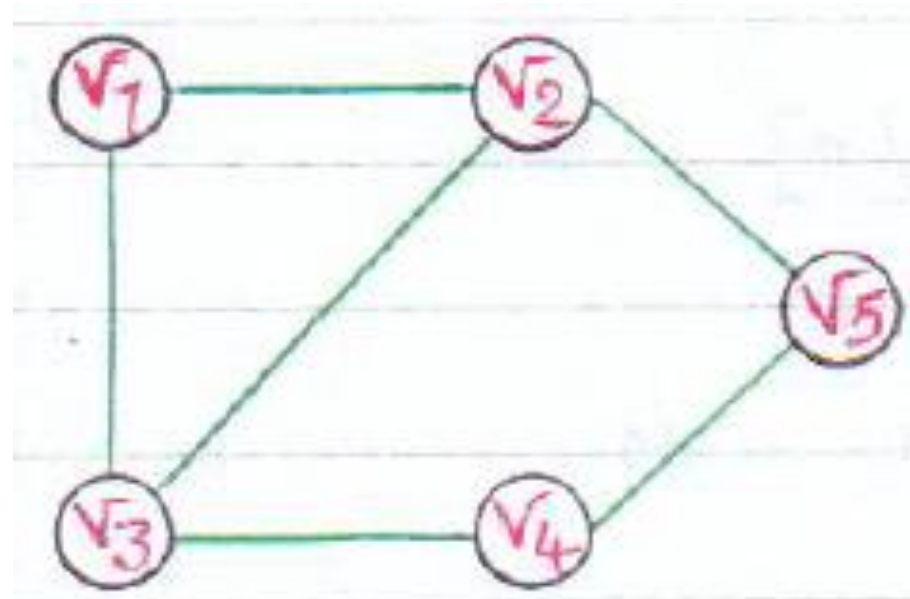
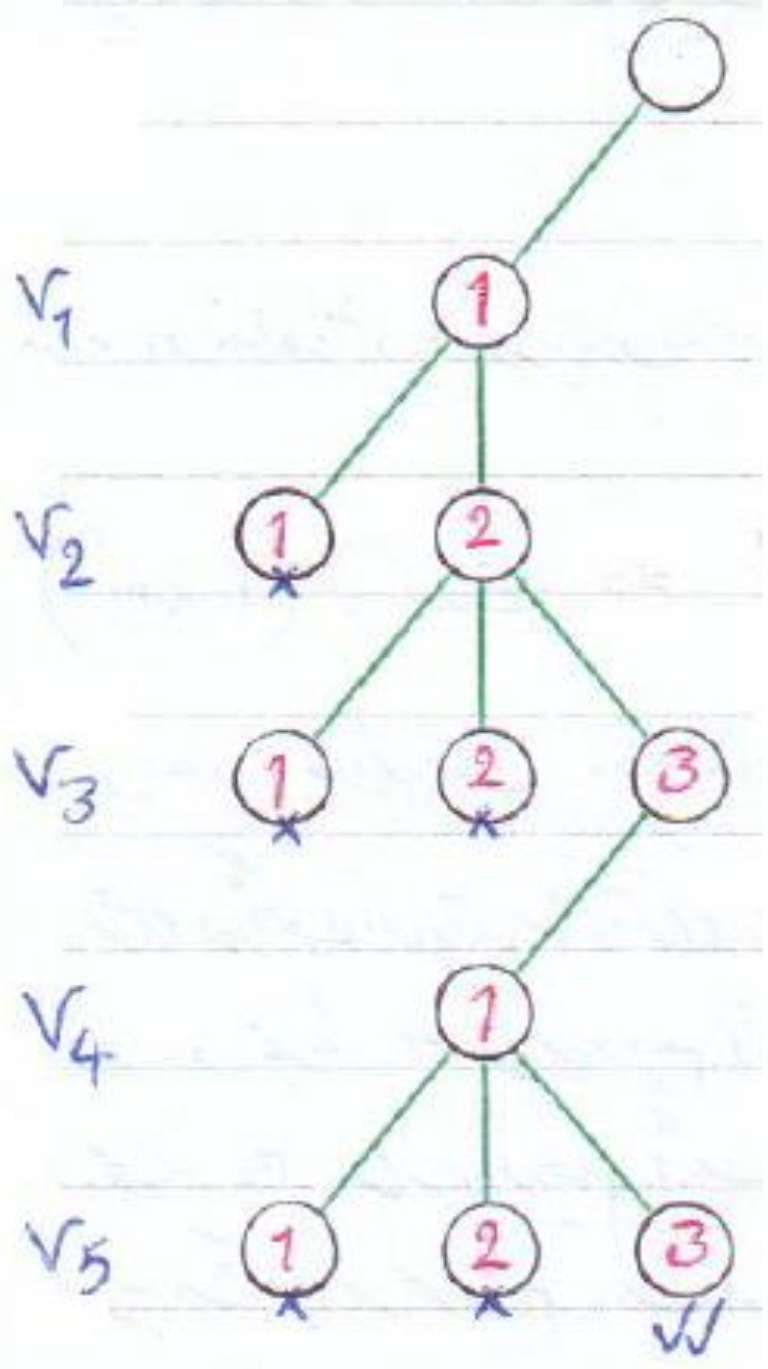
# رنگ آمیزی گراف

$$m=3$$



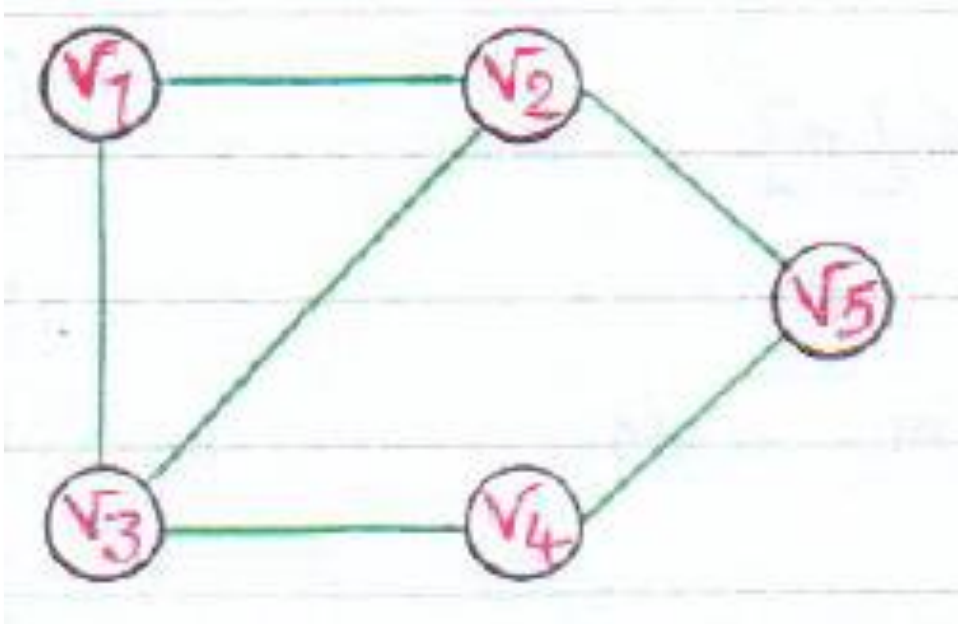
# رنگ آمیزی گراف

$m=3$



# رنگ آمیزی گراف

ورودی مساله:



0	1	1	0	0
1	0	1	0	1
1	1	0	1	0
0	0	1	0	1
0	1	0	1	0

$$L[i, j] = \begin{cases} 1 & \text{اگر دو رأس مجاور باشند} \\ 0 & \text{اگر دو رأس مجاور نباشند} \end{cases}$$



Coloring( $c$ ) {

  if promising( $c$ )

    if  $i = n$

      write  $c[1] \dots c[n]$

  else

    for  $j = 1$  to  $m$  do  
      {

$c[i+1] = j$  ;

        Coloring( $c+1$ ) ;

      }

}

```
Boolean Promising (i) {  
    for k=1 to i-1 do  
        if L[i,k]=1 And c[i]=c[k]  
            return False;  
    return True;  
}
```

```

Boolean Promising (i) {
    for k=1 to i-1 do
        if L[i,k]=1 And c[i]=c[k]
            return False;
    return True;
}

```

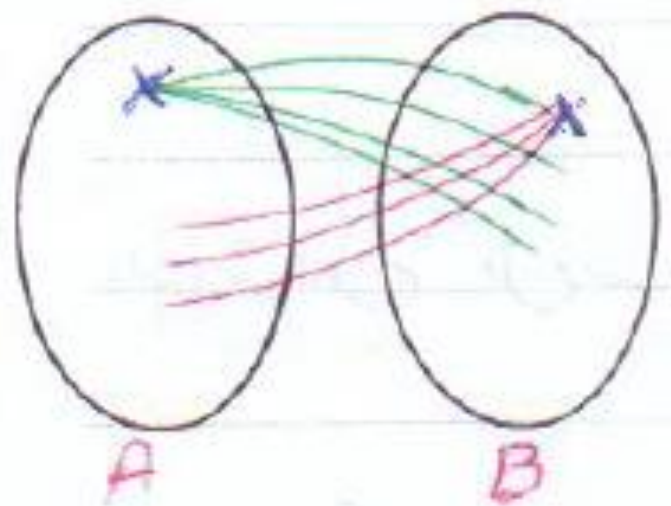
در بیشترین حالت  $n-1$  بار تکرار خواهیم شد  $\Rightarrow$  مرتبه  $n$  خواهیم بود  
 زمان اجرا  $\rightarrow$

$$(1 + m + m^2 + \dots + m^n) \times n = \frac{m^{n+1} - 1}{m - 1} \times n \xrightarrow{\text{As } n \rightarrow \infty} \in O(n \times m^n)$$

$$(1 + m + m^2 + \dots + m^n) \times n = \frac{m^{n+1} - 1}{m - 1} \times n \quad \text{تقریر} \Rightarrow \in O(n \times m^n)$$

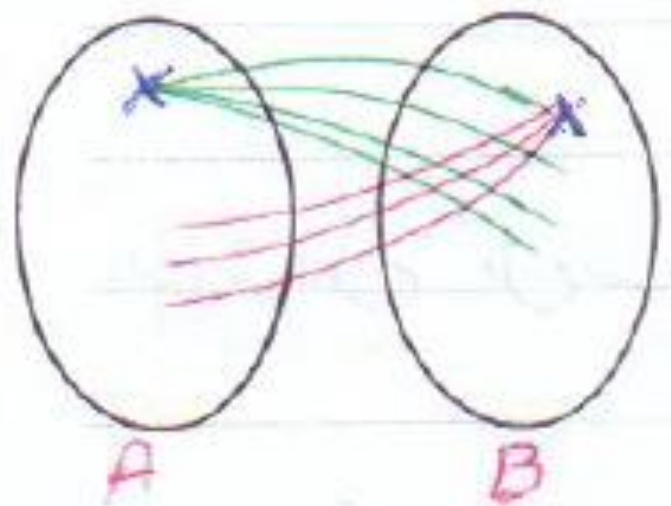
میلی درجہ اولیاء و انت  $n \times 2^n$

$$(1 + m + m^2 + \dots + m^n) \times n = \frac{m^{n+1} - 1}{m - 1} \times n \xrightarrow{\text{تقریر}} \in O(n \times m^n)$$



مبانی درخت فواصل راست  $n \times 2^n$

$$(1 + m + m^2 + \dots + m^n) \times n = \frac{m^{n+1} - 1}{m - 1} \times n \xrightarrow{\text{تقریر}} \in O(n \times m^n)$$



مبانی درخت فوایم راست  $n \times 2^n$

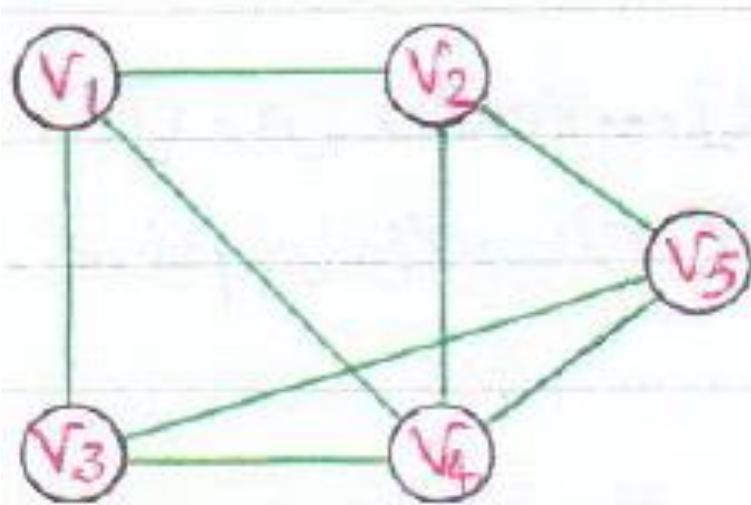
زمان اجرا مبانی درخت  $\in O(n \times n)$

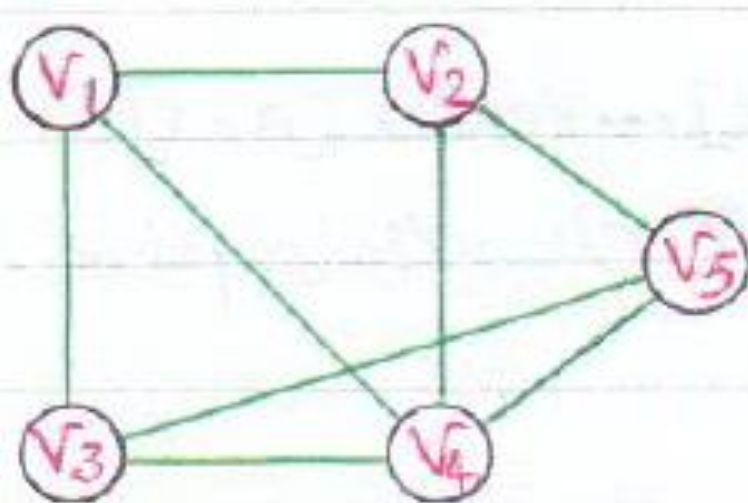
## تمرین

- الگوریتم رنگ آمیزی گراف را روی یک گراف با 6 راس اجرا نمایید. (درخت فضای حالت را تا رسیدن به اولین جواب ترسیم نمایید.)



دور همیلتونی





سطح 0



سطح 1



سطح 2



سطح 3



سطح 4



```
Hamilton(c) {  
    if promising(c)
```

```
        if  $c = n-1$ 
```

```
            write vertex[0]...vertex[n-1]
```

```
        else
```

```
            for  $j=2$  to  $n$  do  
            {
```

```
                vertex[i+1] = j;
```

```
                Hamilton(c+1);
```

```
            }
```

```
}
```

Boolean Promising (i) {

if  $i = n-1$  And  $L[\text{vertex}[n-1], \text{vertex}[0]] = 0$   
return False;

if  $i \neq 0$  And  $L[\text{vertex}[i-1], \text{vertex}[i]] = 0$   
return False;

for  $k=1$  to  $i-1$  do  
if  $\text{vertex}[k] = \text{vertex}[i]$   
return False;

return True;

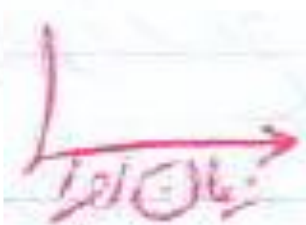
}



```

Boolean Promising (i) {
    if i == n-1 And L[vertex[n-1], vertex[0]] = 0
        return False;
    if i != 0 And L[vertex[i-1], vertex[i]] = 0
        return False;
    for k = 1 to i-1 do
        if vertex[k] = vertex[i]
            return False;
    return True;
}

```


 در بدترین حالت  $n-2$  متباین می‌شود  $\Rightarrow O(n)$

• زمان اجرای کلی:

$$\left(1 + (n-1) + (n-1)^2 + \dots + (n-1)^{n-1}\right) \times n = \frac{(n-1)^n - 1}{(n-1) - 1} \times n \in O((n-1)^n)$$

• زمان اجرای کلی:

$$(1 + (n-1) + (n-1)^2 + \dots + (n-1)^{n-1}) \times n = \frac{(n-1)^n - 1}{(n-1) - 1} \times n \in O((n-1)^n)$$

$$\theta(n \times 2^n)$$

• یادآوری: زمان اجرای فروشنده دوره گرد

## تمرین

- الگوریتم دور همیلتونی را روی یک گراف با 6 راس اجرا نمایید. (درخت فضای حالت را تا رسیدن به اولین جواب ترسیم نمایید.)



مساله كوله پشتی ۱-۰

## • مساله کوله پشتی ۱ -

• ایده حل:

• حد (Bound)

## مساله کوله پشتی ۱-۰

$i$	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

$$W = 15$$

i	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

$$W = 15$$

مجموع: ۰  
 Profit: ارزش اجناس بدست آمده  
 Weight: وزن اجناس بدست آمده  
 88\$

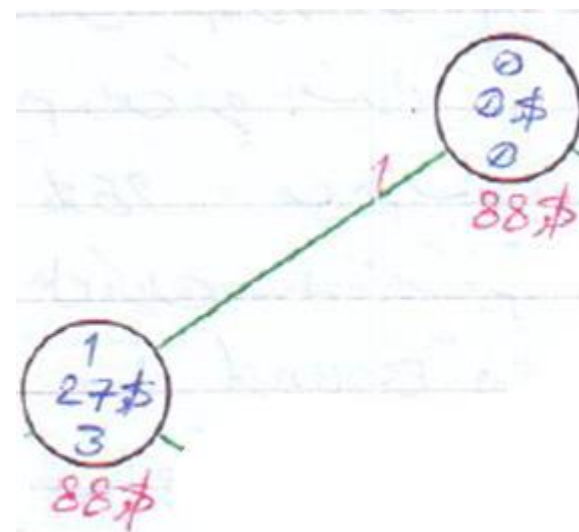
$i$	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

$$W = 15$$

Maxprofit

0\$

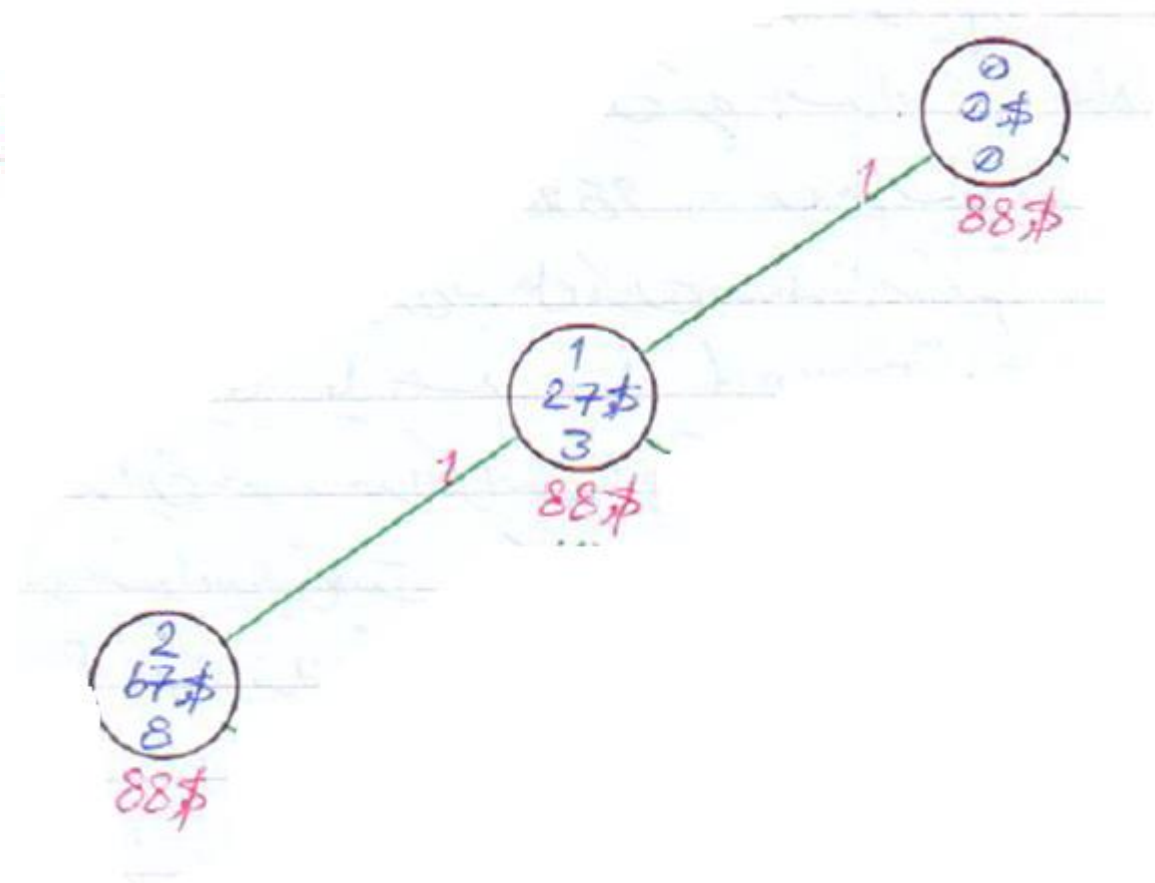
27\$



$i$	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

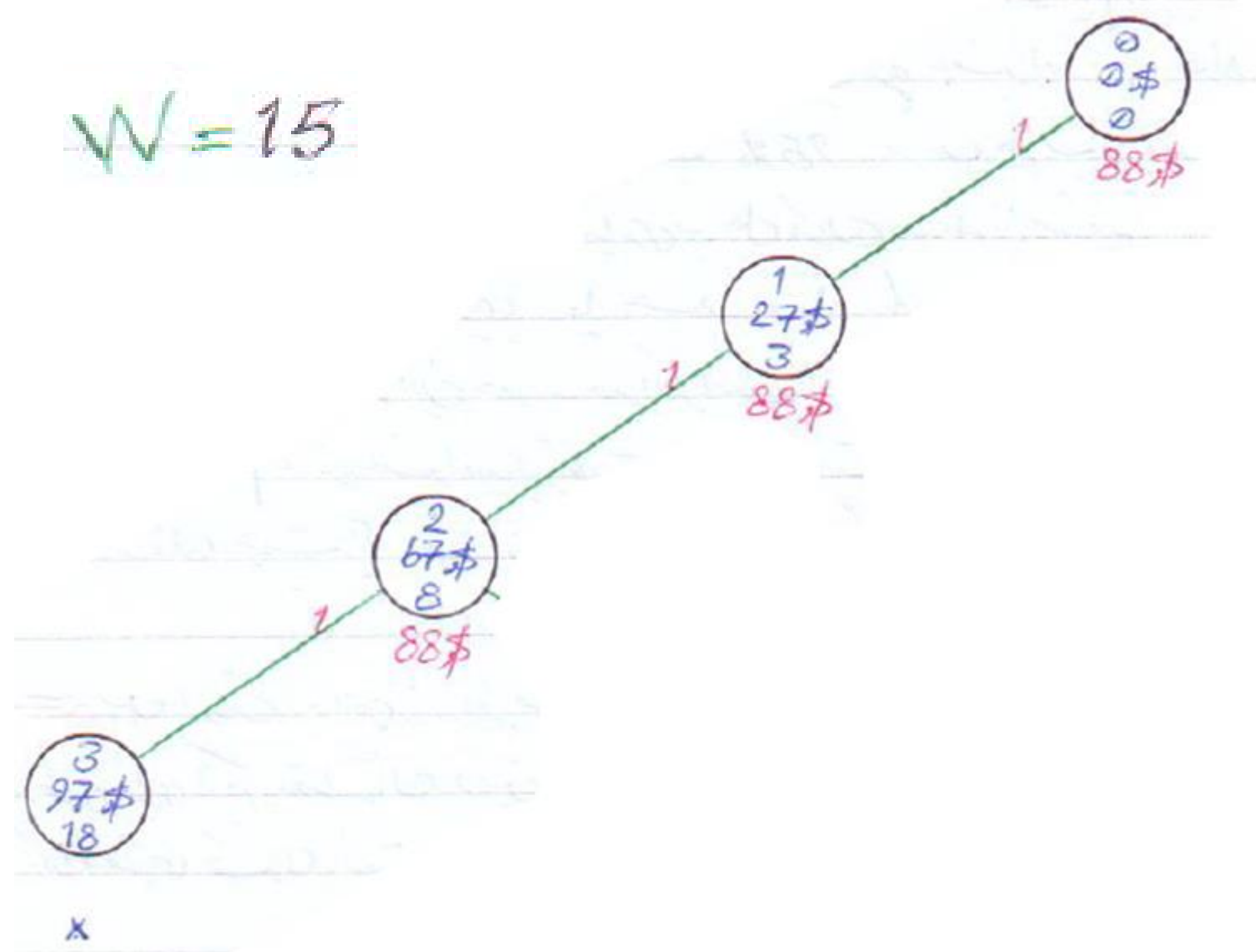
$$W = 15$$

Maxprofit  
 0\$  
 27\$  
 67\$



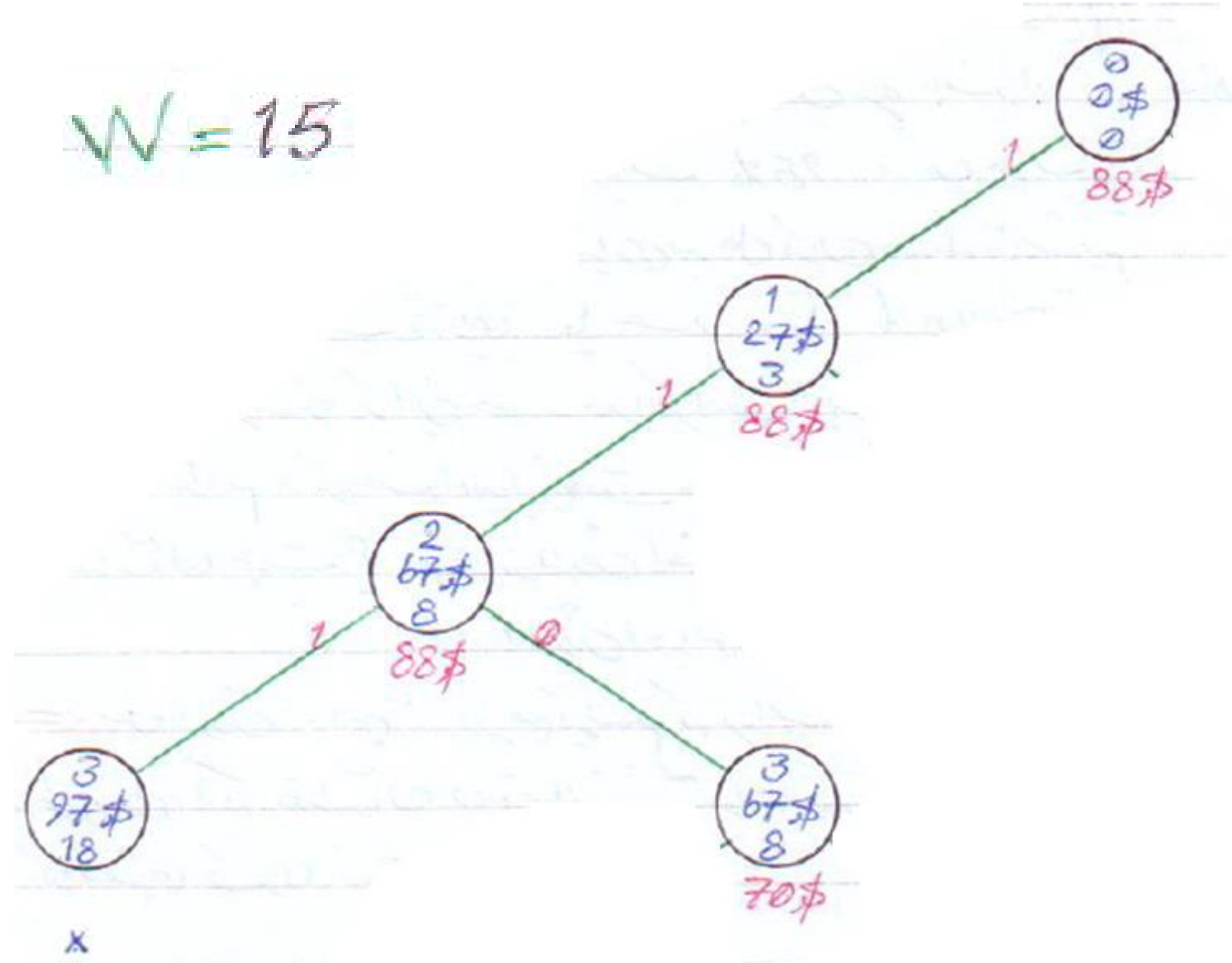
i	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

Maxprofit  
0\$  
27\$  
67\$



i	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

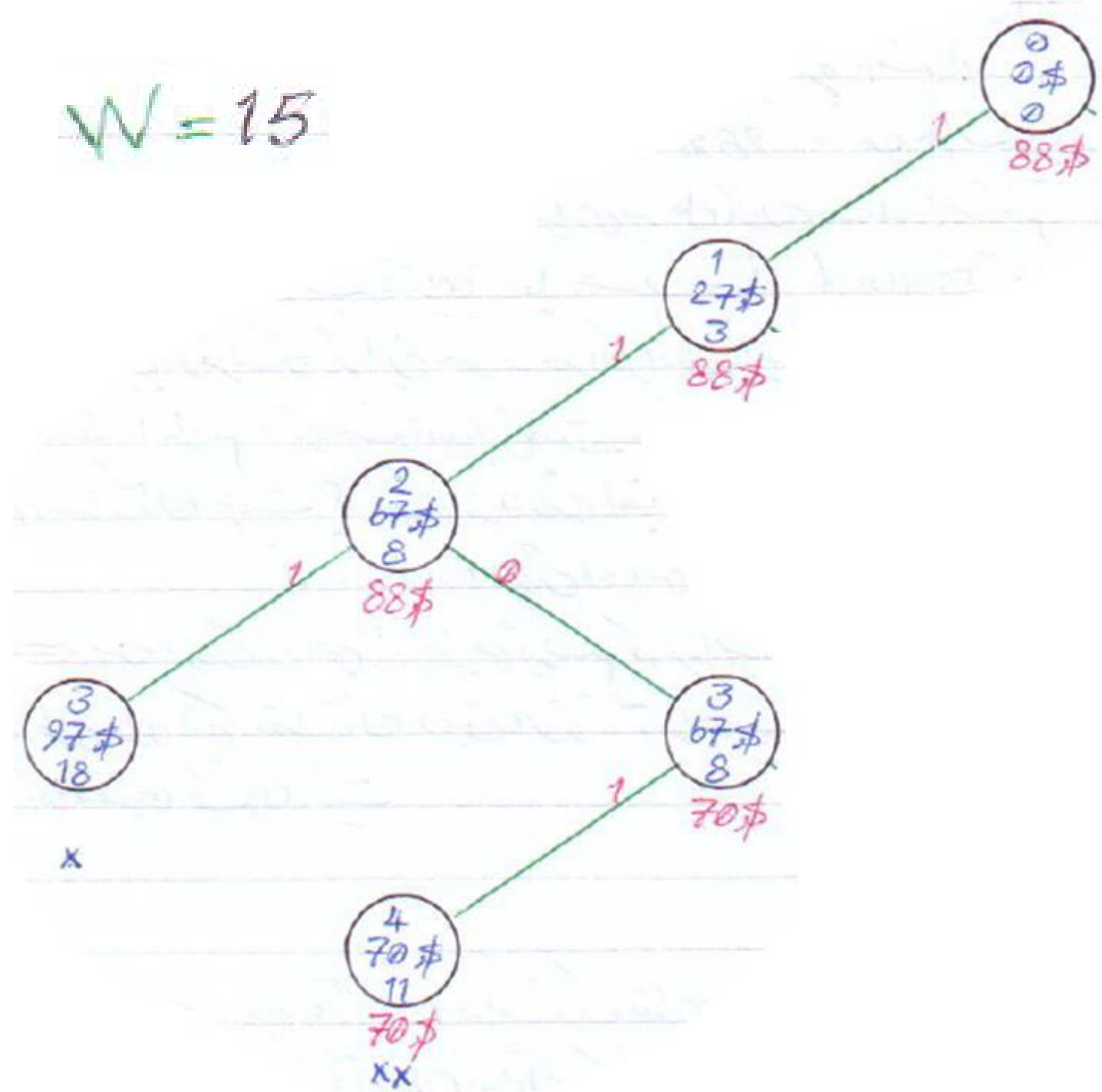
Maxprofit  
0\$  
27\$  
67\$





i	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

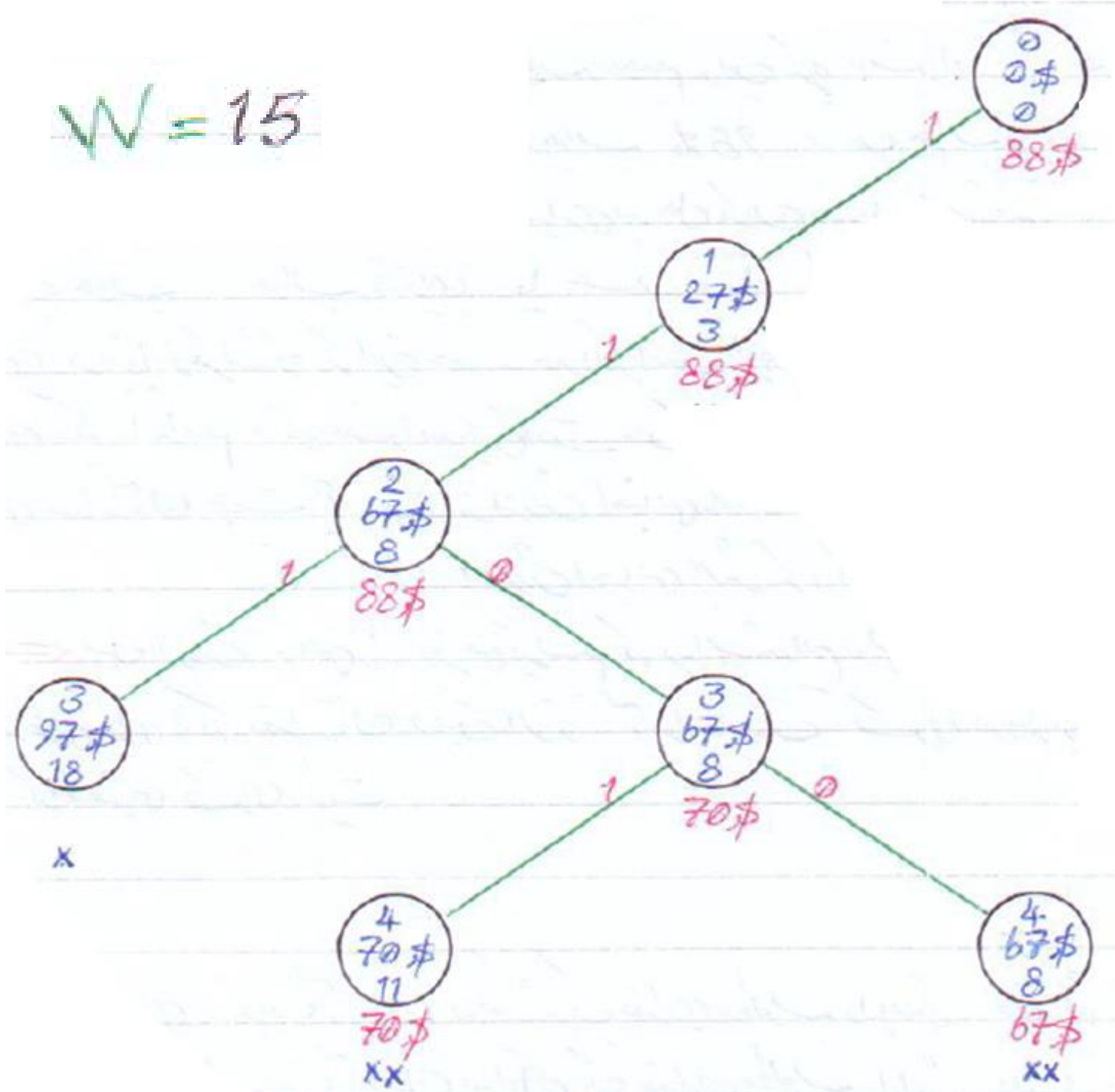
Maxprofit  
 0\$  
 27\$  
 67\$  
 70\$



i	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

$$W = 15$$

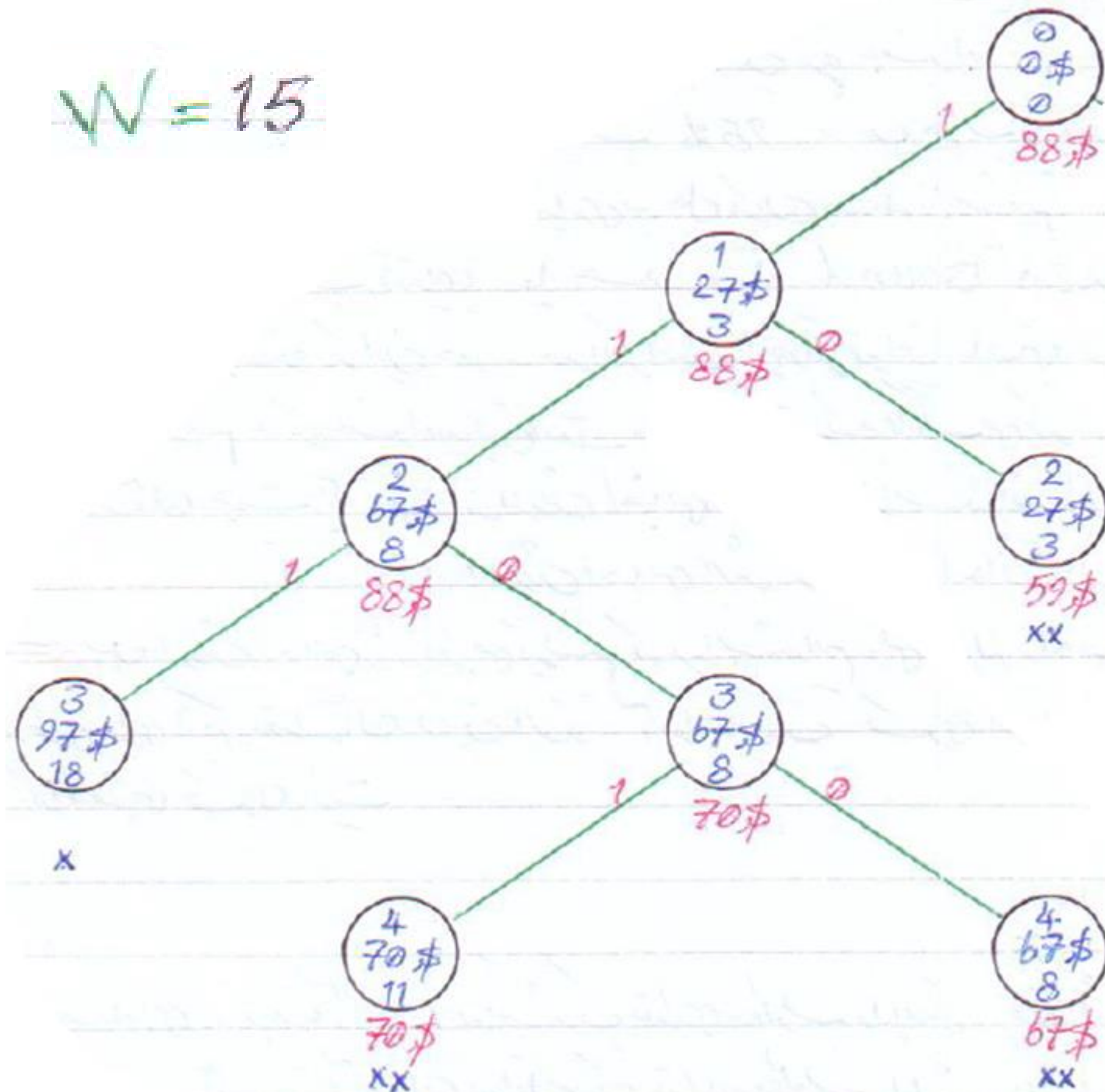
Maxprofit  
 0\$  
 27\$  
 67\$  
 70\$



i	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

Maxprofit  
0\$  
27\$  
67\$  
70\$

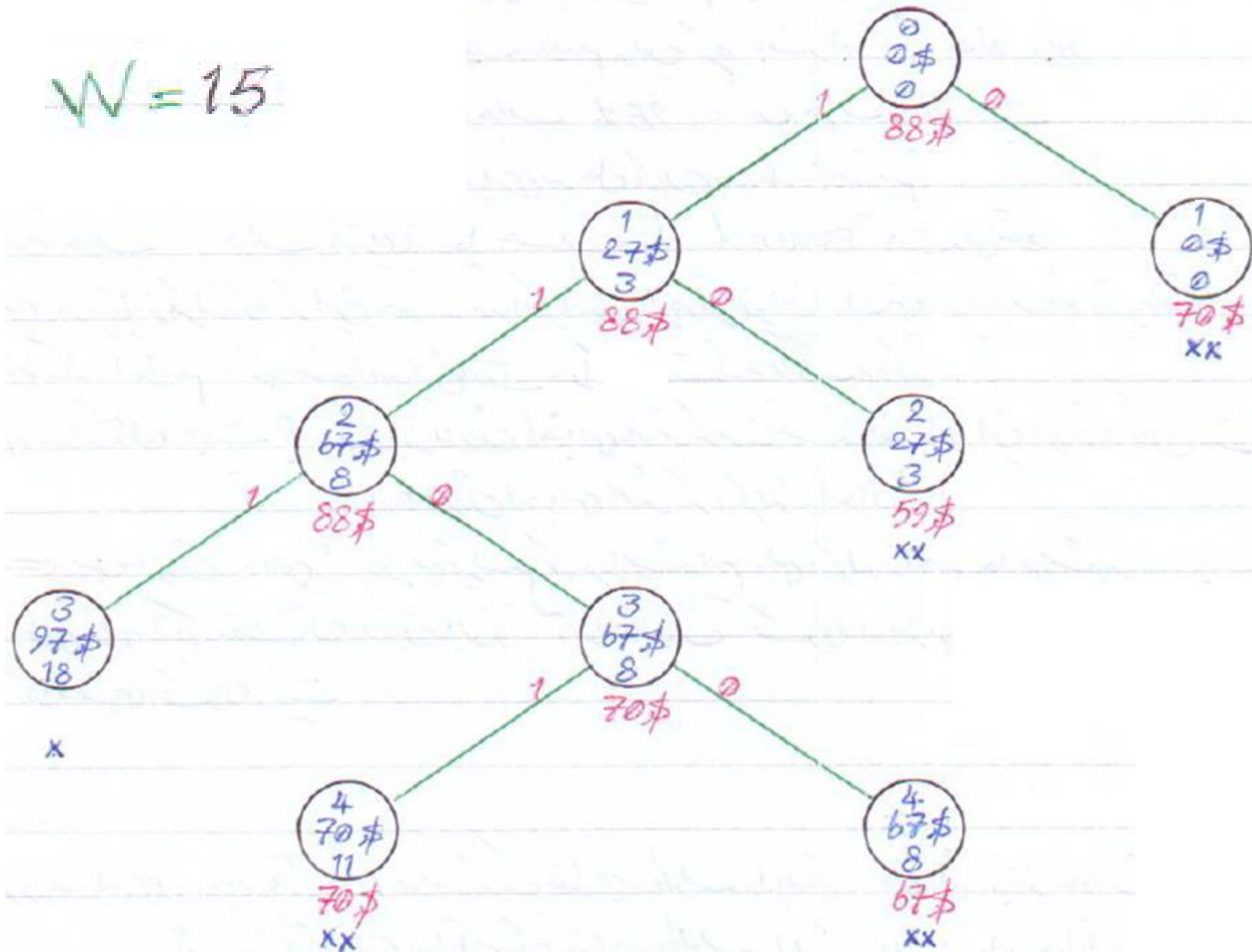
$$W = 15$$



i	$p_i$	$w_i$
1	27\$	3
2	40\$	5
3	30\$	10
4	3\$	3

$$W = 15$$

Maxprofit  
 0\$  
 27\$  
 67\$  
 70\$





```
knapsack (i, Profit, weight) {  
    if weight  $\leq$  W and Profit  $>$  MaxProfit  
    {  
        MaxProfit = Profit;
```

```
    }  
    if Promising (i, Profit, weight)  
    {  
        x[i+1] = 1;  
        knapsack (i+1, Profit + P[i+1], weight + w[i+1]);  
        x[i+1] = 0;  
        knapsack (i+1, Profit, weight);  
    }  
}
```

```
knapsack (i, Profit, weight) {  
  if weight  $\leq$  W and Profit  $>$  MaxProfit  
  {
```

```
    MaxProfit = Profit;
```

```
    Best i = i;
```

```
    Best x [1..i] = x [1..i];
```

```
  }
```

```
  if Promising (i, Profit, weight)  
  {
```

```
    x [i+1] = 1;
```

```
    knapsack (i+1, Profit + P [i+1], weight + W [i+1]);
```

```
    x [i+1] = 0;
```

```
    knapsack (i+1, Profit, weight);
```

```
  }
```

```
}
```

Boolean Promising (i, Profit, weight) {

if weight  $\geq$  W return False;

k = i + 1; Bound = Profit; total = weight;

while (k  $\leq$  n and total + w[k]  $\leq$  W)

{

total += w[k];

Bound += P[k];

k++

}

if k  $\leq$  n Bound +=  $\lceil (W - \text{total}) / w[k] \rceil \times P[k];$

if Bound  $\leq$  MaxProfit return False;

return True;

}



Boolean Promising (c, Profit, weight) {

if weight  $\geq W$  return False;

k = c + 1; Bound = Profit; total = weight;

while (k  $\leq$  n and total + w[k]  $\leq$  W)

{

total += w[k];

Bound += P[k];

k++

}

if k  $\leq$  n Bound +=  $\left[ (W - \text{total}) / w[k] \right] \times P[k];$

if Bound  $\leq$  MaxProfit return False;

return True;

}

مقدار n را در نظر بگیرید

→  $\in O(n)$   
زمان اجرا



• زمان اجرای کلی:

$$(1 + 2 + 2^2 + \dots + 2^n) \times n = (2^{n+1} - 1) \times n \Rightarrow \in O(n \times 2^n)$$

• زمان اجرای کلی:

$$(1 + 2 + 2^2 + \dots + 2^n) \times n = (2^{n+1} - 1) \times n \Rightarrow \in O(n \times 2^n)$$

• یادآوری: زمان اجرای کوله پشتی ۱-۰ به روش برنامه ریزی پویا  $\Theta(nw)$

## تمرین

- الگوریتم کوله پشتی ۱-۰ را برای ۵ شیء اجرا نمایید. (درخت فضای حالت را ترسیم نمایید.)

پایان