طراحی الگوریتم

درس ششم: روش عقبگرد – روش شاخه و قید

Backtracking Algorithms

Branch and Bound Algorithms

مدرس:

فرشید شیرافکن

دانشجوی **دکتری دانشگاه تهران**

(کارشناسی و کارشناسی ارشد : کامپیوتر نرم افزار) (دکتری: بیو انفورماتیک)

مسئله "هزار توی شمشادی"

مسیری را طی کرده و اگر به بن‌بست برسیم، بر می‌گردیم و راهی دیگر را امتحان می‌کنیم.

اگر تابلویی باشد که به ما بگوید مسیری که پیش گرفته‌ایم به بن‌بست منتهی می‌شود، کارها آسان می‌شود.

اگر این تابلو در اول راه باشد، زمان صرفه‌جویی شده خیلی بیشتر می شود،زیرا همهٔ دو راهی‌های بعد از آن دیگر در نظر گرفته نمی شود.

از این تابلو ها در الگوریتم‌های عقبگردی استفاده می شود.

ساختار کلی الگوریتم های عقبگرد

```
Backtrack(A, k)
{   if A = (a1, a2, ··· , ak) is a solution, then report it;
    else
        {
            k ← k + 1;
            compute Sk;
            while(Sk ≠ ∅){
                ak ← an element of Sk;
                Sk ← Sk − {ak};
                Backtrack(A, k);
        }
    }
}
```

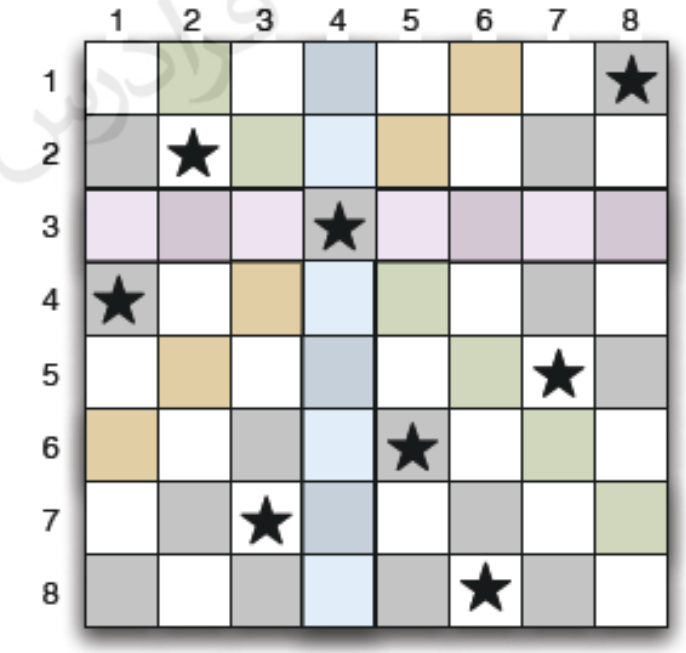
از تکنیک عقبگرد برای حـل مسـائلی استفاده می‌شود که در آن‌ها دنباله‌ای از اشیاء از یک مجموعه مشخص انتخـاب می‌شود، به طوری که این دنباله، ملاکی را در بر می‌گیرد.

مسئله n وزیر

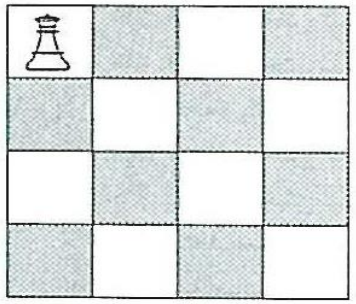
هدف چیدن n مهره وزیر در یک صفحهٔ شطرنج $n \times n$ است، به طوری که هیچ دو وزیری یکدیگر را **گارد** ندهند، یعنی هیچ دو وزیری نباید در یک سطر، ستون یا قطر یکسان باشند.

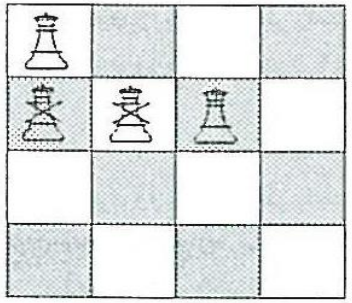
دنباله : n موقعیت که در آنها وزیرها قرار داده می‌شوند.

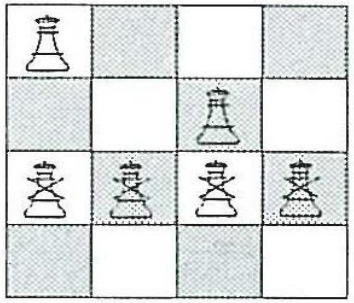
مسئله n وزیر، شکل کلی نمونه‌ای است که در آن $n = 8$ است (صفحه شطرنج استاندارد).

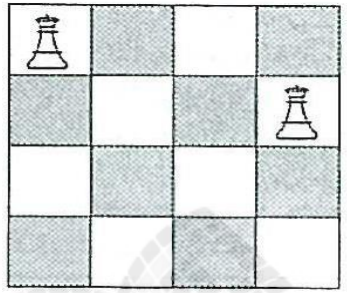


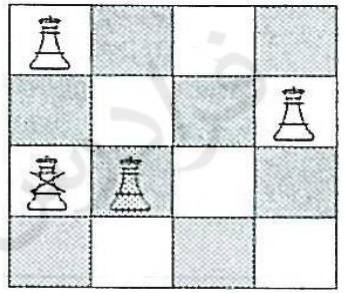
مسئله ۴ وزیر

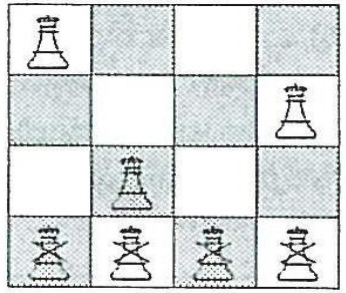

(الف)

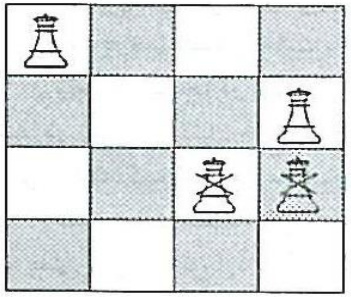

(ب)

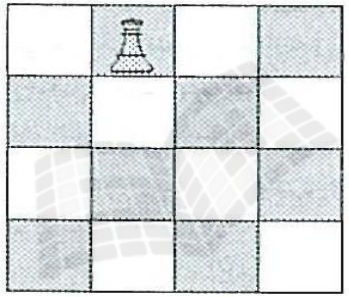

(ج)

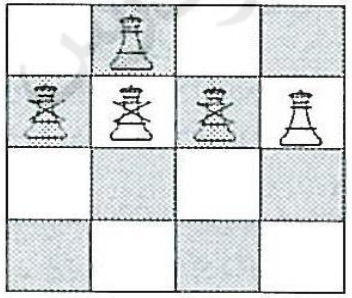

(د)

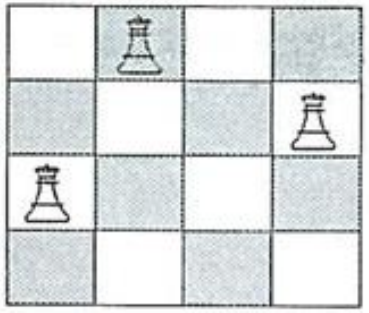

(ه)

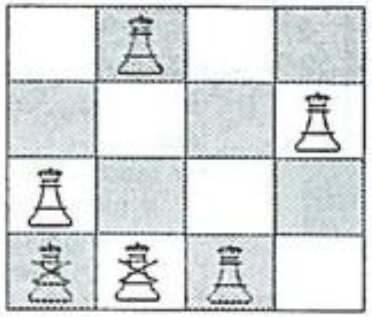

(و)


(ز)


(ح)

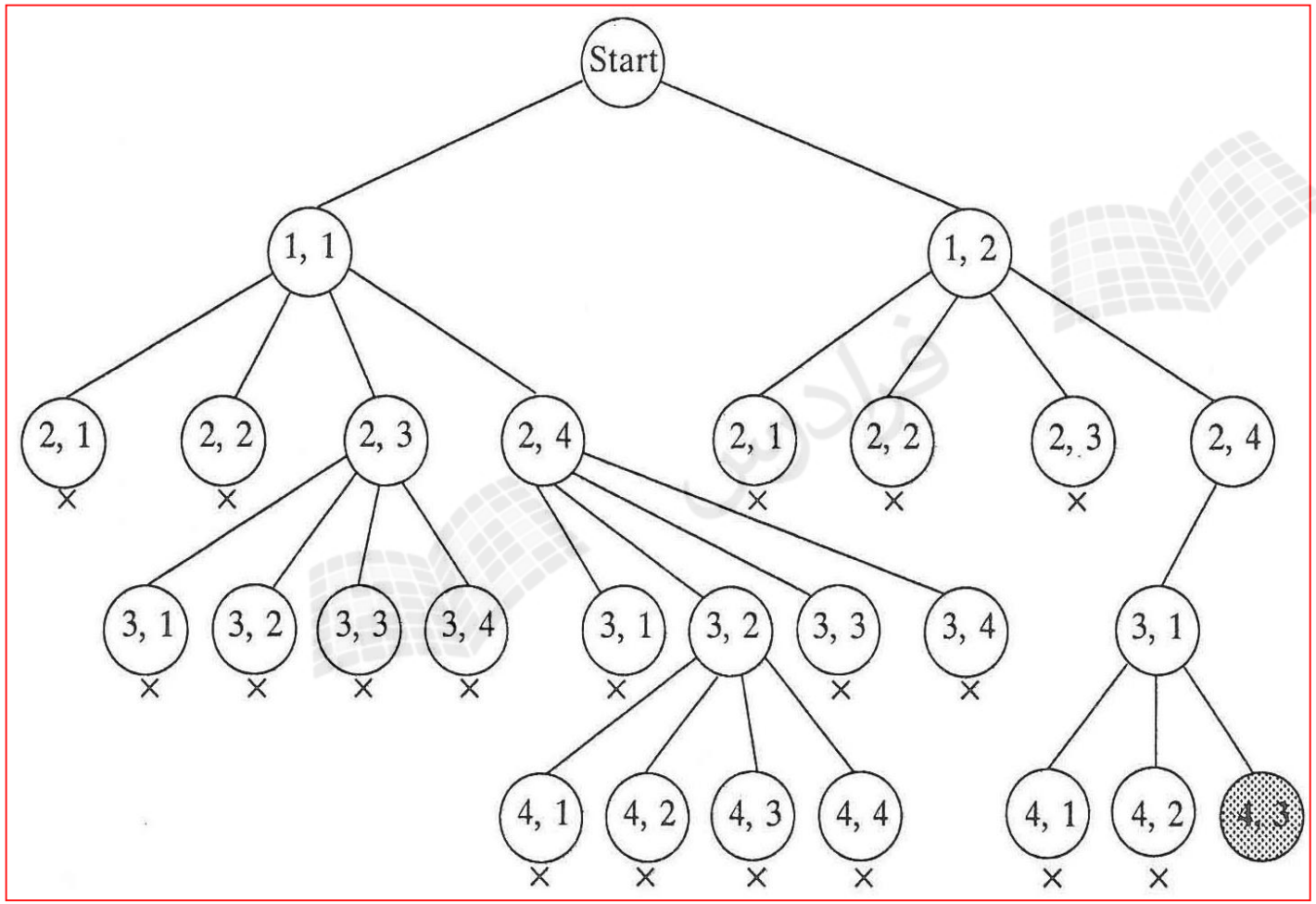

(ط)

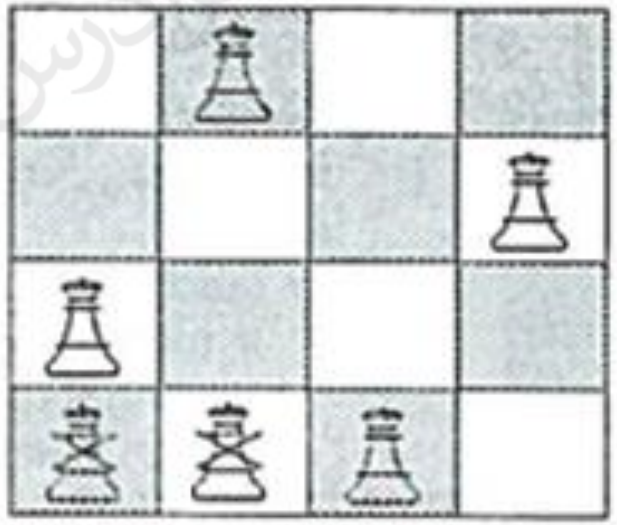

(ی)

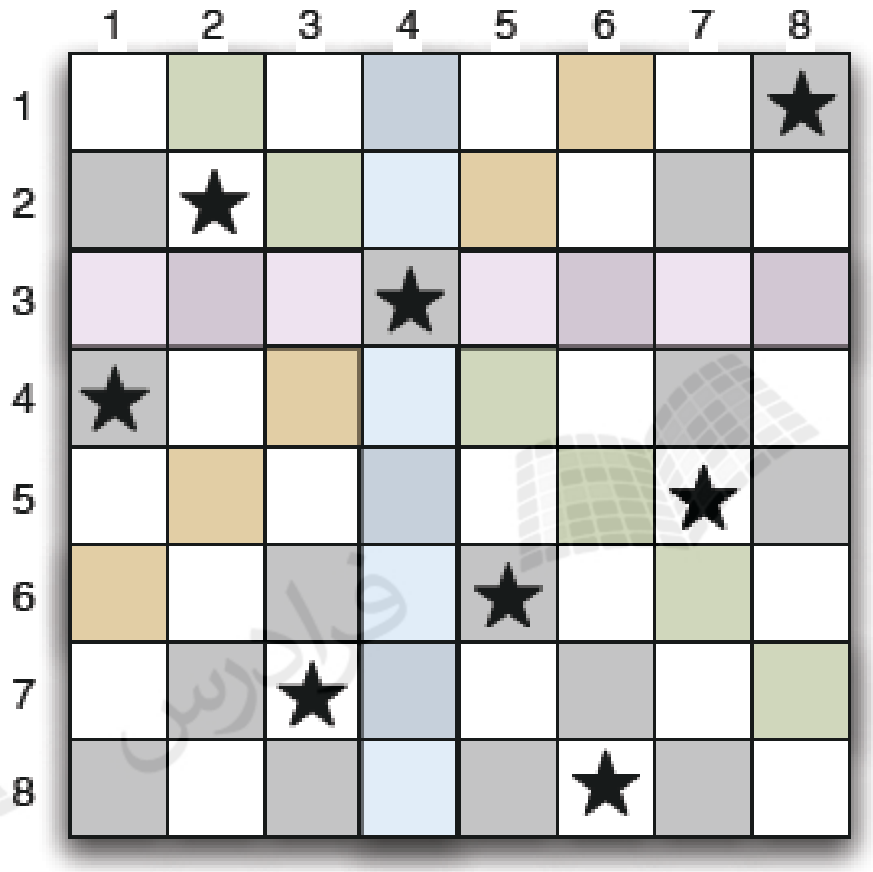

(ک)

استفاده از روش عقبگرد برای حل مسئله ۴ وزیر

بخشی از درخت فضای حالت هرس شده







Row: Queen $Q[i,k]$ conflicts with Queen $Q[j,l] \iff i = j$.

Column: Queen $Q[i,k]$ conflicts with Queen $Q[j,l] \iff k = l$.

Diagonal

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

$Q[i,k]$ conflicts with $Q[j,l]$

$$\Updownarrow$$

$$i-k=j-l$$

Back-Diagonal

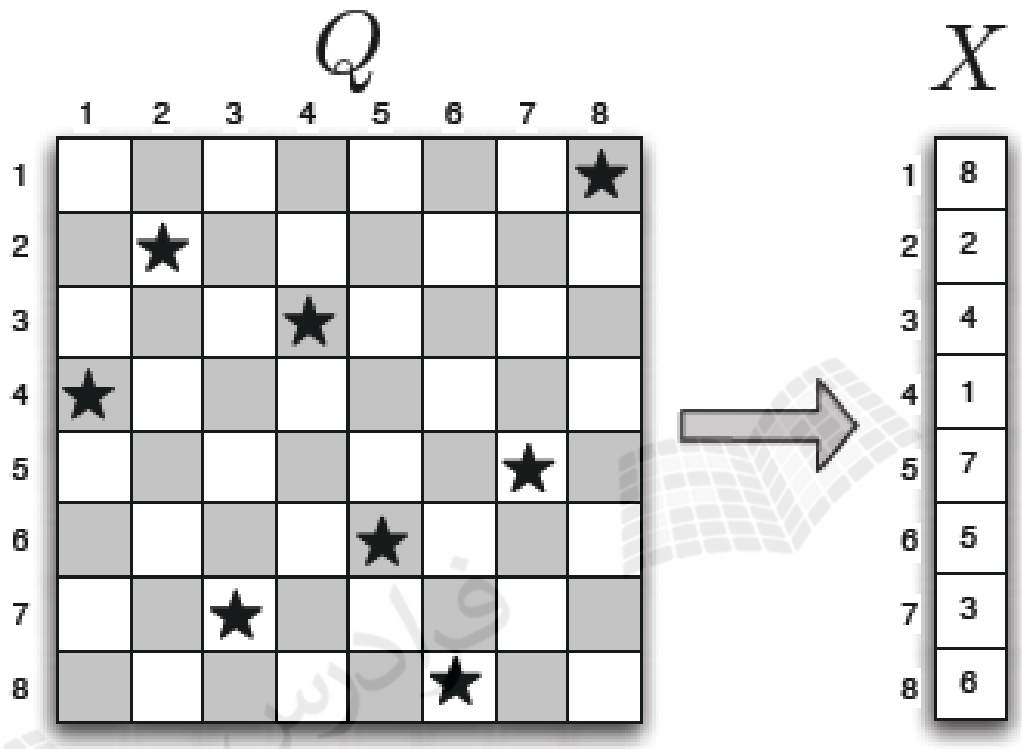
	1	2	3	4	5	6	7	8
1	2	3	4	5	6	7	8	9
2	3	4	5	6	7	8	9	10
3	4	5	6	7	8	9	10	11
4	5	6	7	8	9	10	11	12
5	6	7	8	9	10	11	12	13
6	7	8	9	10	11	12	13	14
7	8	9	10	11	12	13	14	15
8	9	10	11	12	13	14	15	16

$Q[i,k]$ conflicts with $Q[j,l]$

$$\Updownarrow$$

$$i+k=j+l$$

$$|i-j|=|k-l|$$



Suppose that $X[i] = k$ and $X[j] = l$. Queen i conflicts with Queen j if and only if:

- $X[i] = X[j]$, or
- $|i - j| = |X[i] - X[j]|$.

الگوریتم n وزیر

```
CanPlace(X,j){
    for i ← 1 to j − 1 do{
        if(X[i] = X[j] or |X[i] − X[j]| = |i − j|) then
            return false;
    }
    return true;
}
```

```
n − Queen (X,i)
{   if( CanPlace(X,i))
    {   if (i = n) then report(X);
        else{
            for k ← 1 to n do{
                X[i+1] ← k;
                n − Queen(X,i+1);
            }
        }
    }
}
```

تعداد کل گره ها در درخت فضای حالت

$$1+n+n^2+n^3+\ldots+n^n=\frac{n^{n+1}-1}{n-1}$$

$$\frac{8^{8+1}-1}{8-1}=19173961$$

1 گره در سطح صفر، n گره در سطح 1، n^2 گره در سطح 2 و ... و n^n گره در سطح n است.

تعداد کل گره ها ی امید بخش در درخت فضای حالت:

$$1+n+n(n-1)+n(n-1)(n-2)+\ldots+n!$$

$$1+8+8\times7+8\times7\times6+8\times7\times6\times5+\ldots+8!=109,601$$

مسئله

حاصل جمع زیر مجموعه‌ها

مسئله حاصل جمع زیر مجموعه‌ها

یافتن همهٔ زیر مجموعه‌هایی از n عدد به طوری که حاصل جمع آنها برابـر مقدار معین W شود.

n و W اعداد صحیح مثبت هستند.

n=5

W=21

$w_1 = 5$ $w_2 = 6$ $w_3 = 10$ $w_4 = 11$ $w_5 = 16$

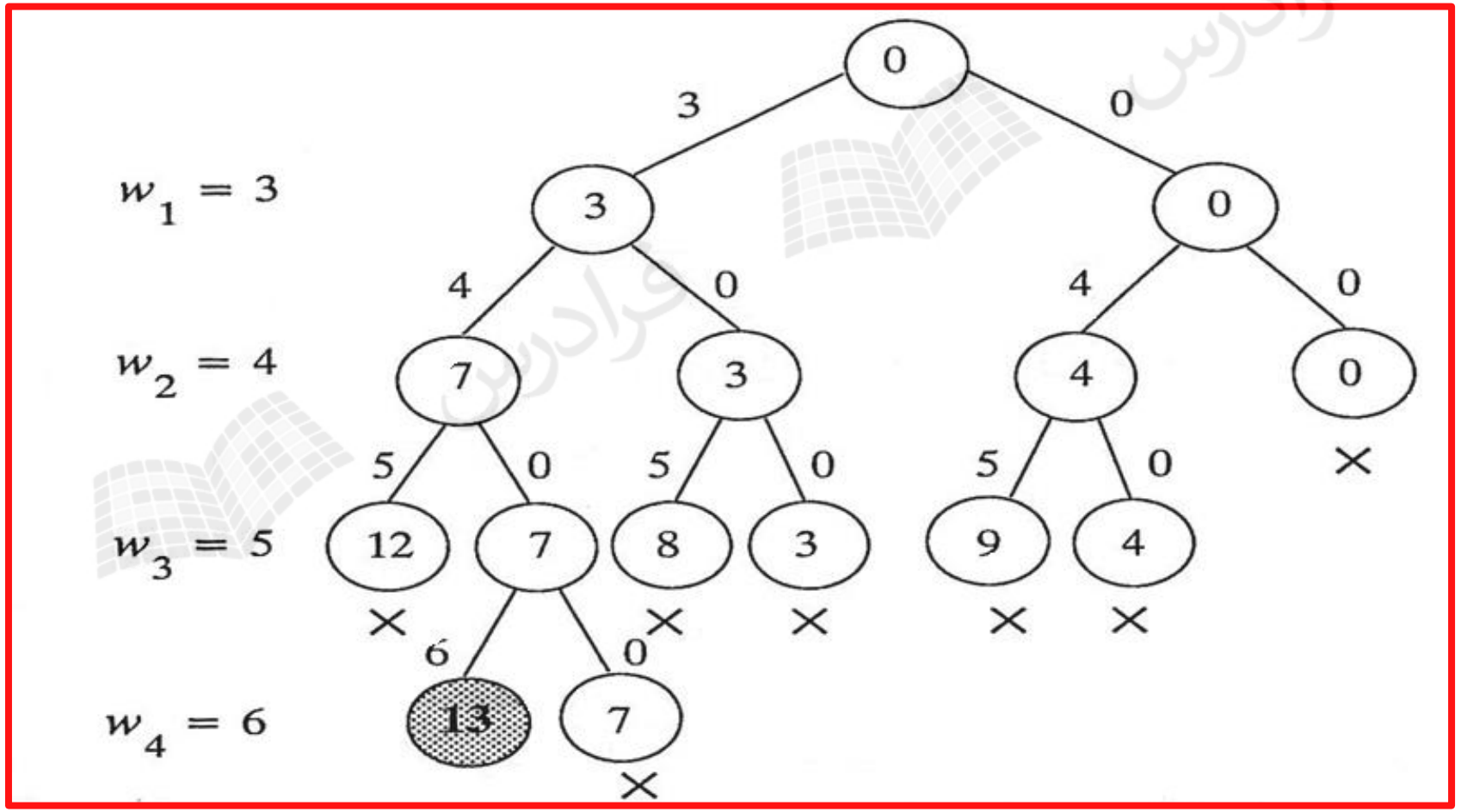
$\{w_1, w_5\}, \{w_3, w_4\}, \{w_1, w_2, w_3\}$

مثال

$w = 13$

$w_1 = 3, w_2 = 4, w_3 = 5, w_4 = 6$

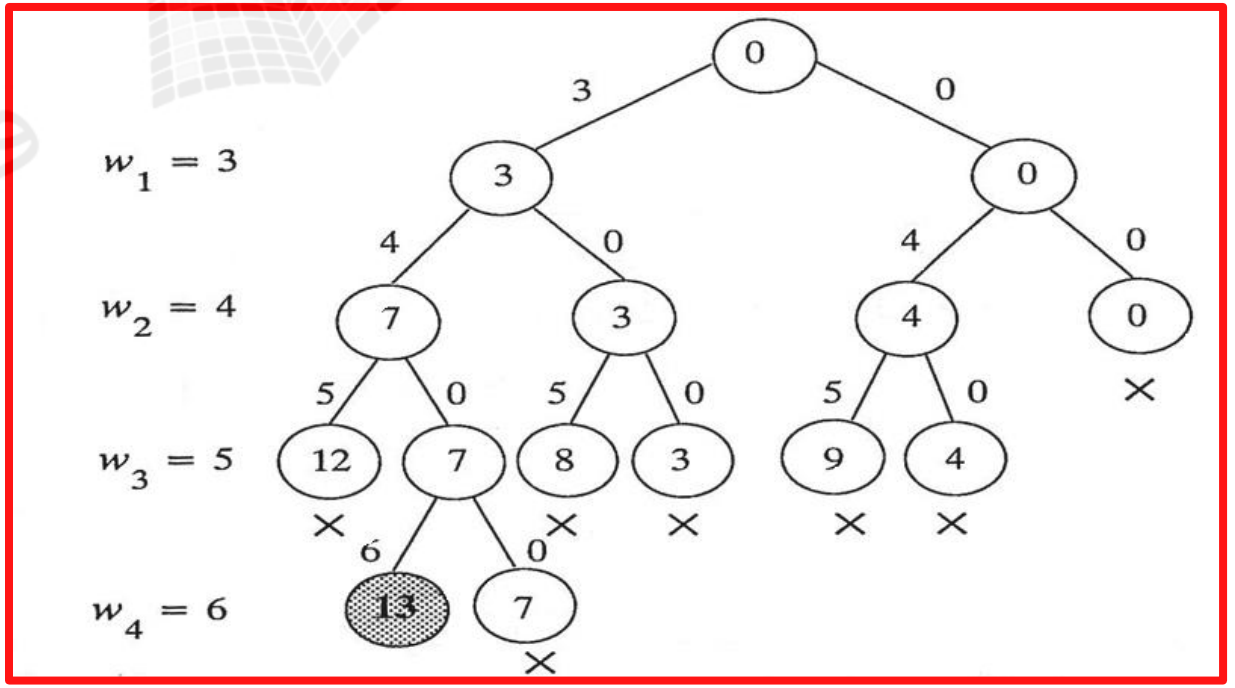
درخت فضای حالت



تعداد گره های درخت فضای حالت

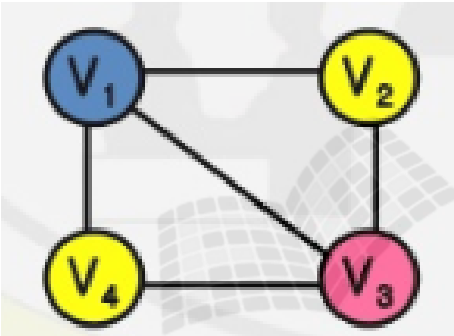
تعداد گره هایی از درخت فضای حالت که توسط الگوریتم حاصل جمع زیر مجموعه ها جستجو می شوند :

$$1+2+2^2+\ldots+2^n=2^{n+1}-1$$



رنگ آمیزی گراف

پیدا کردن همهٔ راه های ممکن برای رنگ آمیزی یک گراف بدون جهت، با استفاده از حداکثر m رنگ متفاوت، به طوری که هیچ دو رأس مجاوری هم رنگ نباشند.

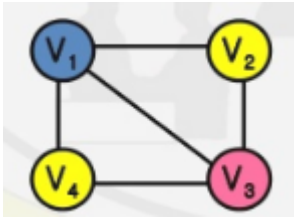


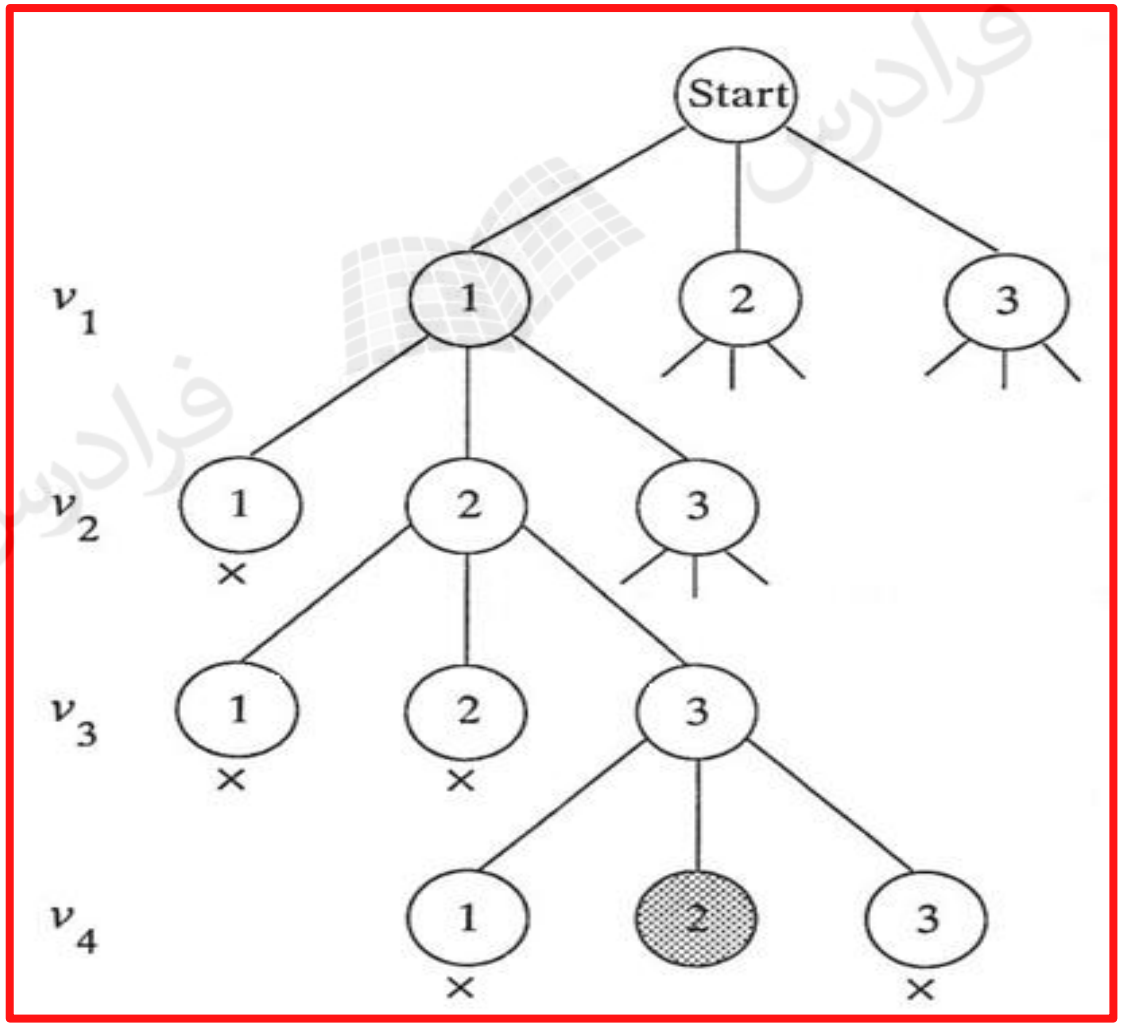
حلی برای مسئله رنگ آمیزی 2 وجود ندارد.

می توان گراف را با ۳ رنگ، رنگ آمیزی کرد.

مثال

درخت فضای حالت هرس شده برای رنگ آمیزی گراف با ۳ رنگ.



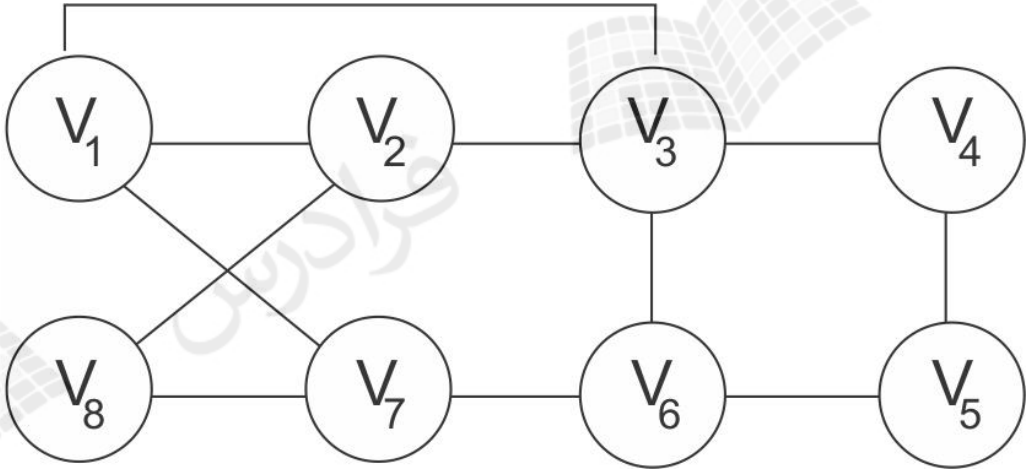


تعداد گره ها در درخت فضای

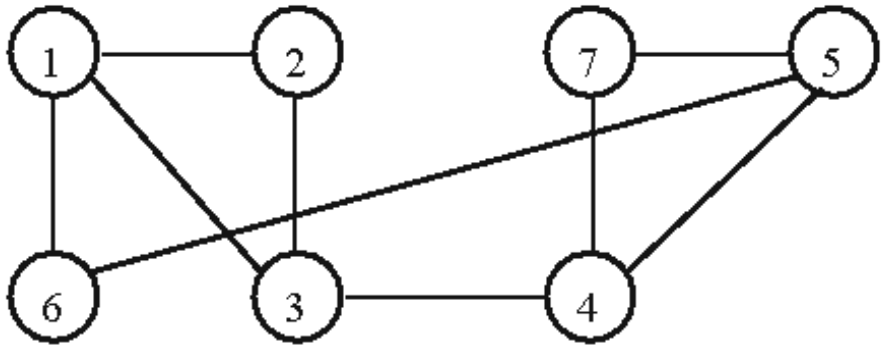
$$1+m+m^2+\ldots+m^n=\frac{m^{n+1}-1}{m-1}$$

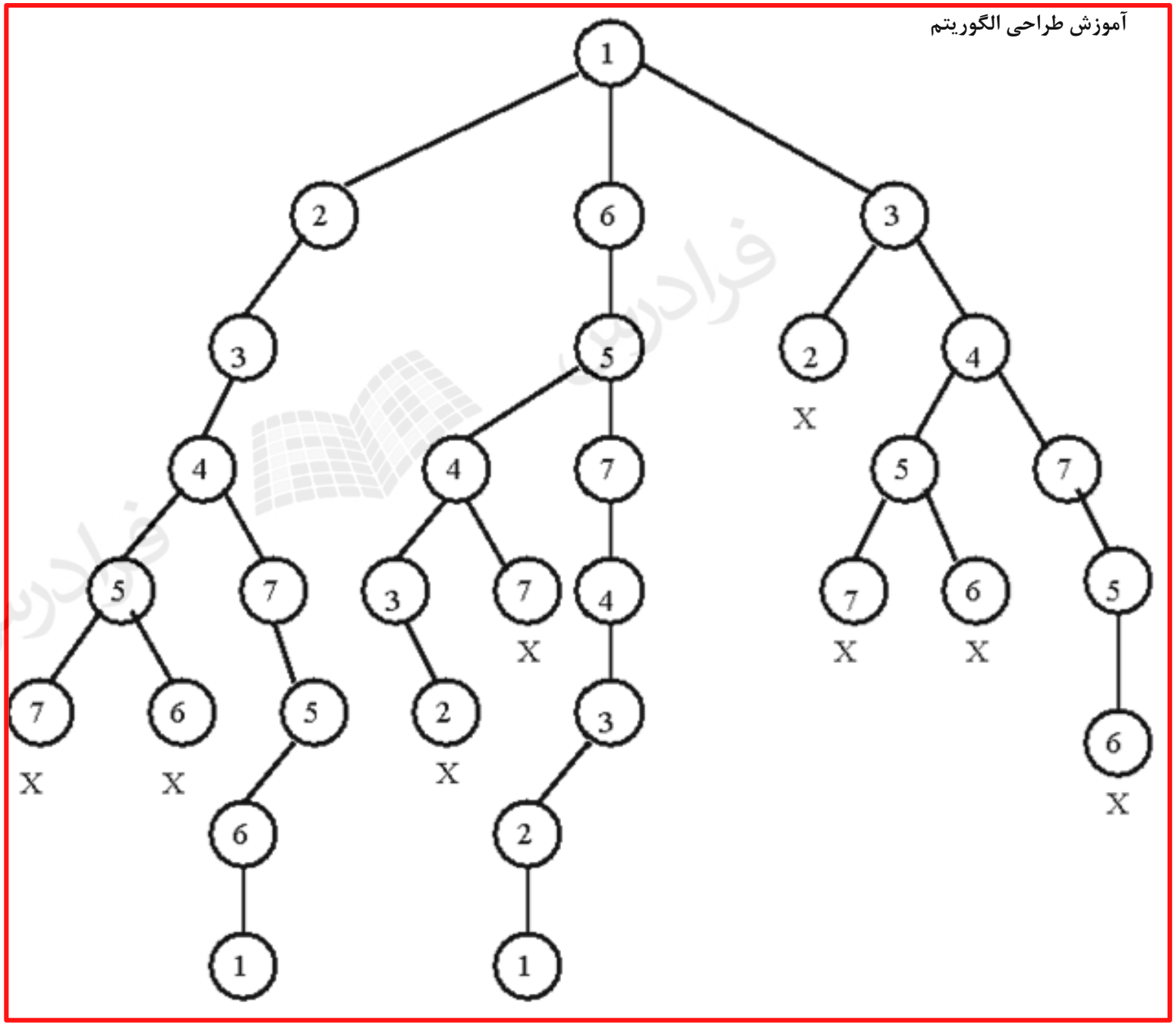
مسئله مدارهای هامیلتونی

مدار هامیلتونی (تور)، در یک گراف متصل و بدون جهت، مسیری است که از یک رأس شروع شده، هر یک از رئوس گراف را دقیقاً یکبار ملاقات می کند و در رأس شروع، به پایان می رسد.



[v1, v2, v8, v7, v6, v5, v4, v3, v1]





تعداد گره ها در درخت فضای حالت :

$$1+(n-1)+(n-1)^2+\ldots+(n-1)^{n-1}=\frac{(n-1)^n-1}{n-2}$$

خلاصه

تعداد کل گره هایی که در درخت فضای حالت بررسی می شوند:

N وزیر	$1+n+n^2+n^3+\ldots+n^n=\frac{n^{n+1}-1}{n-1}$
جمع زیر مجموعه ها	$1+2+2^2+\ldots+2^n=2^{n+1}-1$
رنگ آمیزی گراف با m رنگ	$1+m+m^2+\ldots+m^n=\frac{m^{n+1}-1}{m-1}$
دور هامیلتونی	$1+(n-1)+(n-1)^2+\ldots+(n-1)^{n-1}=\frac{(n-1)^n-1}{n-2}$

الگوریتم عقبگرد برای 0/1-Knapsack

```
Backtrack-Knapsack(X, optX, optP, ℓ){
    if ℓ = n + 1 then{
        if Σ_{i=1}^{n} x_i w_i ≤ b then{
            curP ← Σ_{i=1}^{n} x_i p_i;
            if curP ≥ optP then{
                optP ← curP;
                optX ← [x_1, x_2, ··· , x_n];
            }
        }
    }
    else{
        x_l ← 1;
        Backtrack-Knapsack(X, optX, optP, ℓ + 1);
        x_l ← 0;
        Backtrack-Knapsack(X, optX, optP, ℓ + 1);
    }
}
```

P = [23,24,15,13,16]

W = [11,12,8,7,9]

b = 26.

الگوریتم شاخه و قید

Branch-and-Bound is based on backtracking,
which is an exhaustive searching technique in the space of all feasible solutions.

الگوریتم شاخه و قید برای 0/1-Knapsack

```
B&B-Knapsack1(X, optX, optP, ℓ, curW){
    if ℓ = n + 1 then{
        if Σ_{i=1}^{n} x_i p_i ≥ optP then{
            optP ← Σ_{i=1}^{n} x_i p_i;
            optX ← [x_1, x_2, ··· , x_n];
        }
    }
    else{
        if curW + w_ℓ ≤ b then C_ℓ ← {1, 0};
        else C_ℓ ← {0};
    }
    for each x ∈ C_ℓ do {
        x_l ← x;
        Backtrack-Knapsack(X, optX, optP, ℓ + 1, curW + x_ℓ w_ℓ);
    }
}
```

الگوریتم شاخه و قید برای 0/1-Knapsack (نسخه بهتر)

```
B&B-Knapsack2(X, optX, optP, ℓ, curW){
    if ℓ = n+1 then{
        if Σ_{i=1}^{n} x_i p_i ≥ optP then{
            optP ← Σ_{i=1}^{n} x_i p_i;
            optX ← [x_1, x_2, ··· , x_n];
        }
    }
    else{
        if curW + w_ℓ ≤ b then C_ℓ ← {1,0};
        else C_ℓ ← {0};
    }
    B ← Σ_{i=1}^{l-1} x_i p_i + GFK(p_ℓ, p_{ℓ+1}, ··· , p_n, w_ℓ, w_{ℓ+1}, ··· , w_n, b − curW);
    if B ≤ optP then return;
    for each x ∈ C_ℓ do {
        x_l ← x;
        Backtrack-Knapsack(X, optX, optP, ℓ+1, curW + x_ℓ w_ℓ);
    }
}
```

مثال

P = [23,24,15,13,16]

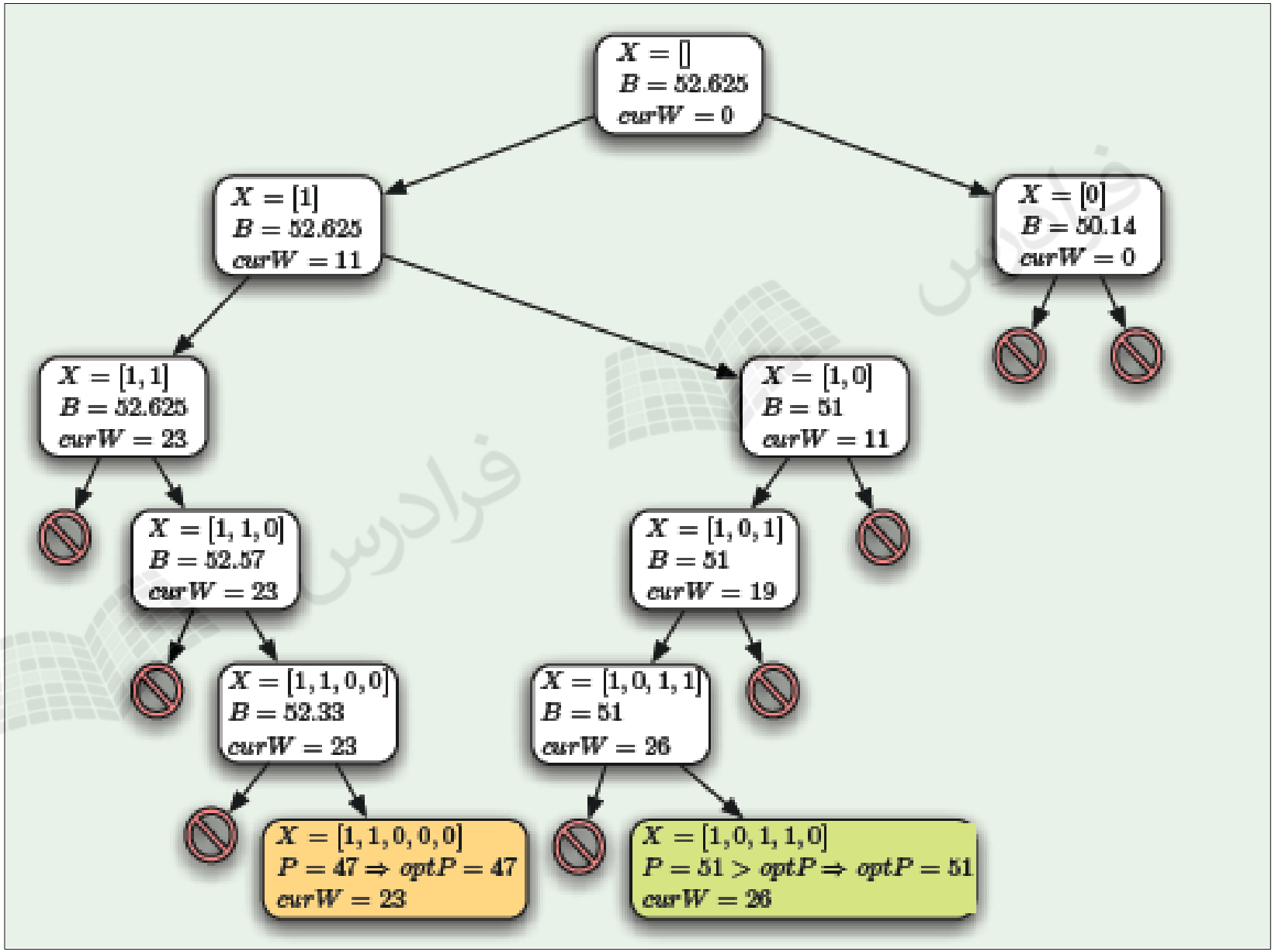
W = [11,12,8,7,9]

b = 26.

X=[1,1,3/8,0,0]

profit=

23+24+3/8(15)=52.625

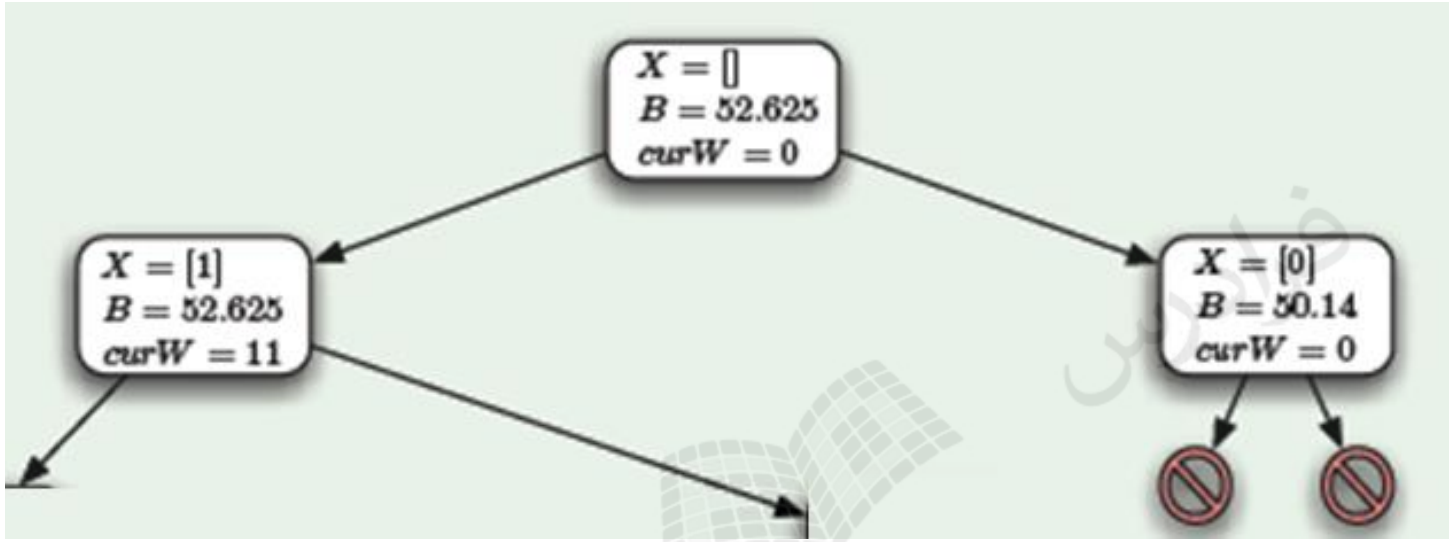


ادامه مثال

P = [23 , 24 , 15 , 13 , 16]

W = [11 , 12 , 8 , 7 , 9]

b = 26.



$$B \leftarrow \sum_{i=1}^{l-1} x_i p_i + GFK(p_\ell, p_{\ell+1}, \cdots, p_n, w_\ell, w_{\ell+1}, \cdots, w_n, b - curW);$$

L = 2

B = 0 + GFK(P2 , .. , P5 , W2 , .. , W5 , 26-0)

= 24 + 15 + 6/7(13) = 50.14

X=[0 , 1 ,1 , 6/7, 0]

مقایسه ۳ الگوریتم برای کوله پشتی 0,1

n	Backtrack-Knapsack	B&B-Knapsack1	B&B-Knapsack2
8	511	333	78
12	8191	4988	195
16	131071	78716	601

worse case size of search space

این اسلاید ها بر مبنای نکات مطرح شده در فرادرس

«آموزش طراحی الگوریتم»

تهیه شده است.

برای کسب اطلاعات بیشتر در مورد این آموزش به لینک زیر مراجعه نمایید

faradars.org/fvsft1092