بسم الله الرحمن الرحیم

طراحی و تحلیل الگوریتم‌ها

مبحث هفتم

روش‌های طراحی الگوریتم

روش جستجوی فضای حالت

Methods of Algorithm Design: State-Space Search

کاظم فولادی

دانشکده مهندسی برق و کامپیوتر

دانشگاه تهران

http://courses.fouladi.ir/algorithm

جستجوی فضای حالت

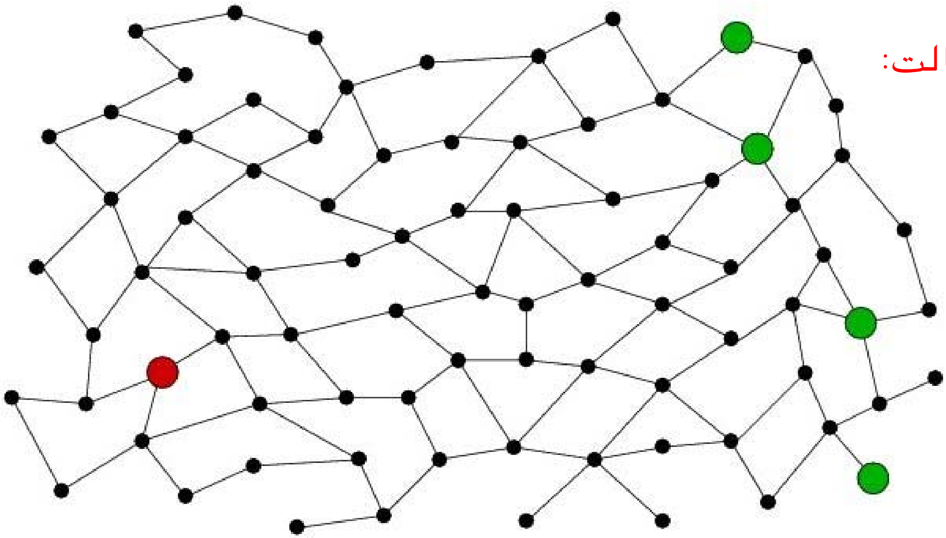
STATE-SPACE SEARCH

- هدف یافتن دنباله‌ای از عناصر از یک مجموعه با یک شرط معین است:

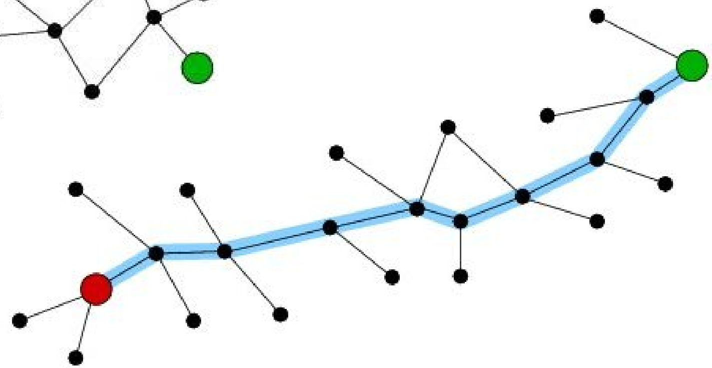
$$(x_1, x_2, x_3, \ldots, x_n)$$

- مجموعه‌ی همه‌ی دنباله‌های ممکن، **فضای حالت** نامیده می‌شود.
- **فضای حالت** را می‌توان به کمک یک **گراف** بازنمایی کرد:
  - گره‌ها: مقادیر متغیرهای مسئله (اجزای دنباله)
  - یال‌ها: رابطه‌ی بین مقادیر (معمولاً رابطه‌ی پدر ـ فرزندی)
  - مسیر: هر مسیر در گراف، یک دنباله را مشخص می‌کند.
  - راه حل: مسیری در گراف با شروع از گره‌ی آغازین به گره‌ی نهایی

گراف فضای حالت و راه حل



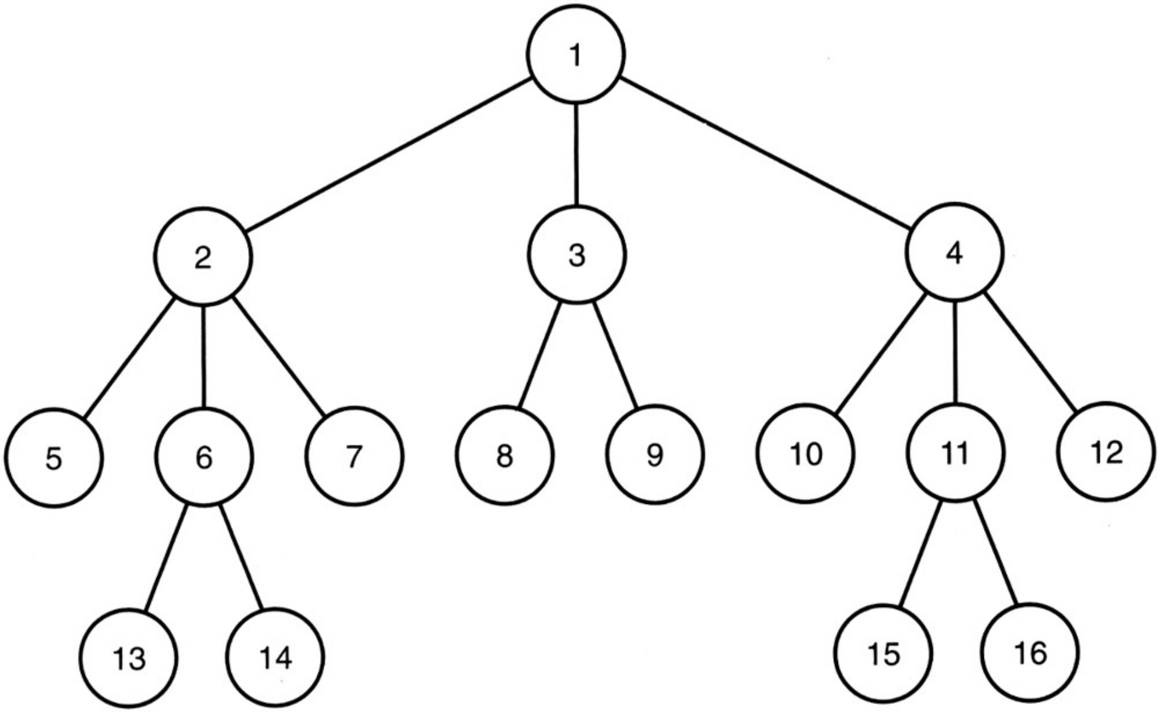
گراف فضای حالت:



راه حل:
مسیری از حالت آغازین به حالت نهایی

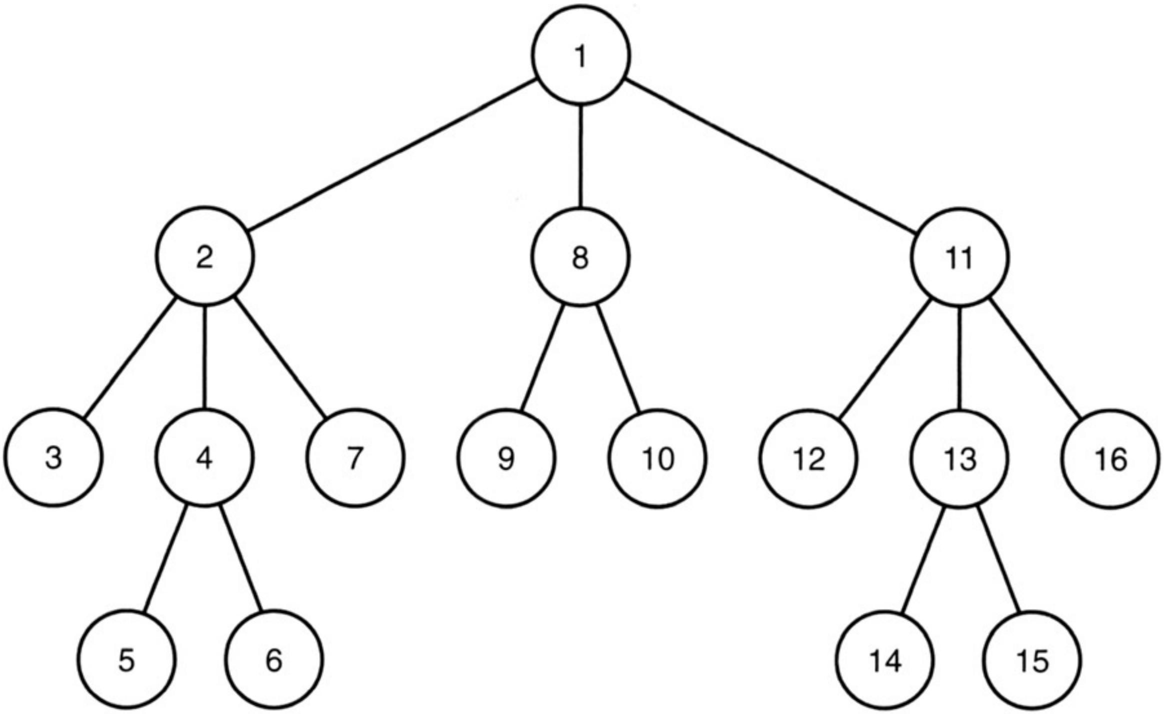
جستجو در فضای حالت با پیمایش عرض ـ اول

Breadth-First Search



جستجو در فضای حالت با پیمایش عمق ـ اول

DEPTH-FIRST SEARCH



BACKTRACKING SEARCH

جستجوی عقب‌گرد

یکی از تکنیک‌های جستجوی فضای حالت است.

- در این روش، گراف فضای حالت به صورت **عمق اول** پیمایش می‌شود.
- هر گاه به گره‌ای برسیم که امکان رسیدن به جواب از آن وجود ندارد، فرزندان آن گره دیگر بررسی نمی‌شوند.

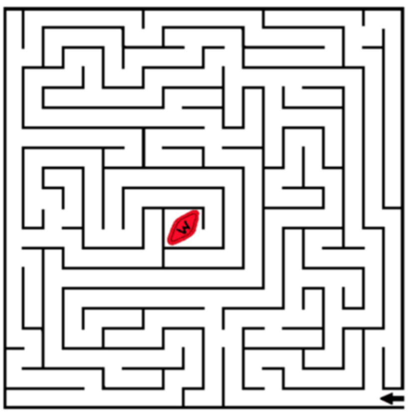
مسئله‌ی مسیر تودرتو

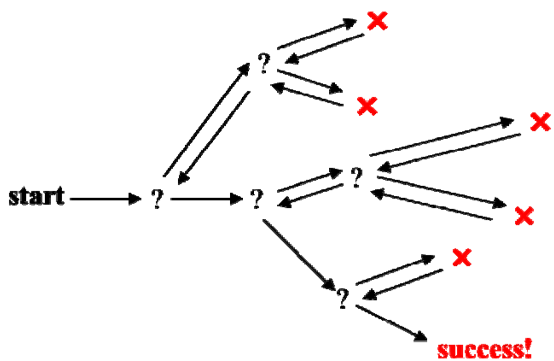
BACKTRACKING SEARCH: MAZE PROBLEM

جستجوی عقب‌گرد

مانند حرکت در یک مسیر تودرتو است

هرگاه از یک مسیر به جواب نرسیدیم، برمی‌گردیم و مسیر دیگری را آزمایش می‌کنیم.
تا جایی بر می‌گردیم که یک گزینه‌ی دیگر برای ادامه بیابیم.





تکنیک عقب‌گرد

شبه‌کد

```
BACKTRACKING-SEARCH(v)
    foreach child u of v
        if promising(u) then
            if solution(u) then
                return u
            else
                BACKTRACKING-SEARCH(u)
```

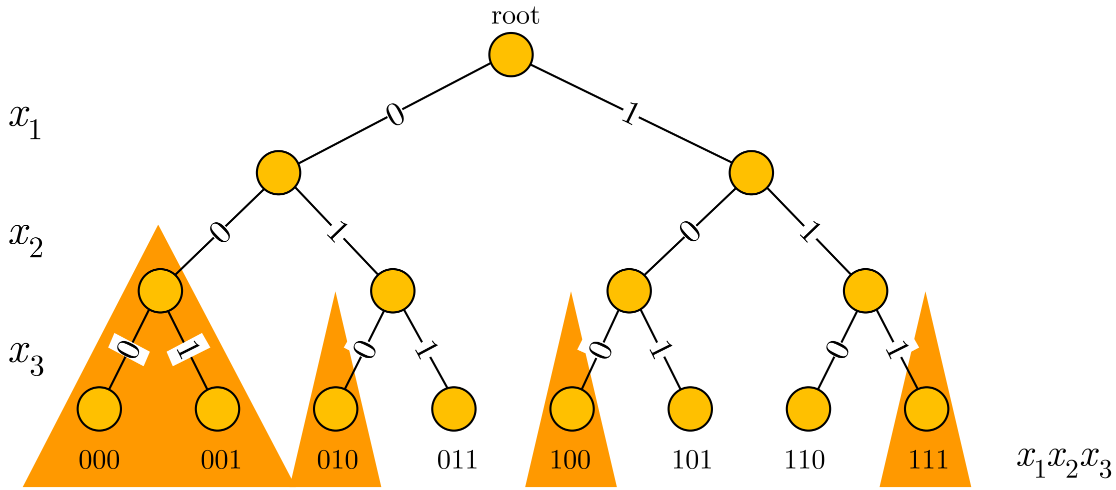
الگوریتم از گره‌ی ریشه شروع می‌شود: $v = \text{root}$

Prepared by Kazim Fouladi | Fall 2016 | 2nd Edition
کاظم فولادی

مثال: مسئله‌ی قفل رمزی

یک قفل رمزی شامل n بیت است. می‌خواهیم رمز مربوطه را پیدا کنیم.

مثلاً: $n = 3$ و می‌دانیم که این رمز شامل دو بیت 1 است.



زیردرخت‌های مشخص شده مربوط به فرزندانی هستند که جواب نمی‌رسند، پس بررسی نمی‌شوند!

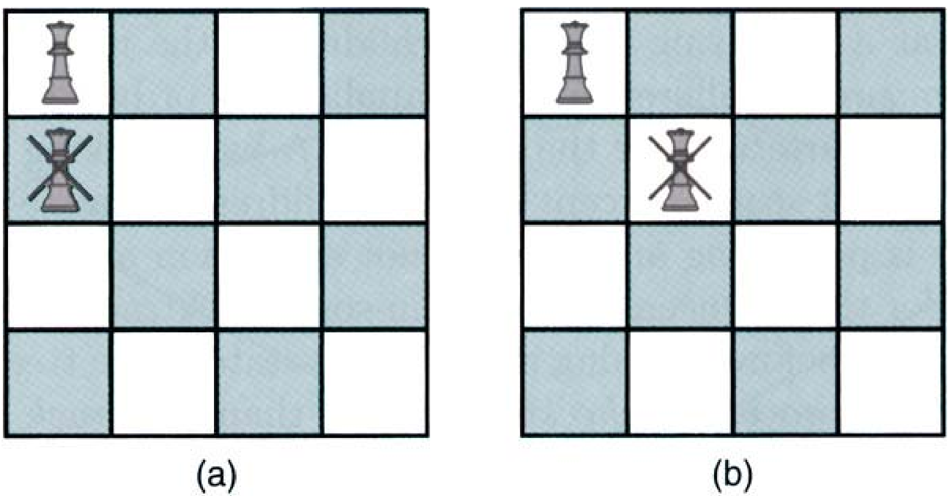
گره‌های غیرامیدبخش = Non-promising

مثال: مسئله‌ی n ـ وزیر

N-QUEENS PROBLEM

می‌خواهیم n وزیر را در یک صفحه‌ی شطرنج $n \times n$ قرار دهیم،
به طوری که هیچ دو وزیری یکدیگر را تهدید نکنند.

مطابق قوانین شطرنج: **هیچ دو وزیری نباید هم سطر، هم ستون یا هم‌قطر باشند.**



مثال: مسئله‌ی n ـ وزیر

N-Queens Problem

می‌خواهیم n وزیر را در یک صفحه‌ی شطرنج $n \times n$ قرار دهیم،
به طوری که هیچ دو وزیری یکدیگر را تهدید نکنند.

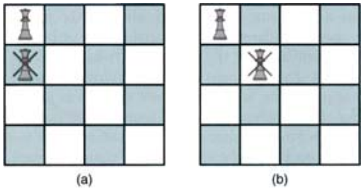
مطابق قوانین شطرنج: **هیچ دو وزیری نباید هم سطر، هم ستون یا هم‌قطر باشند.**

- هر وزیر باید در یک سطر مجزا باشد.
  - پس هر وزیر را به یک سطر نسبت می‌دهیم (شماره‌ی وزیر = شماره‌ی سطر).
- وزیر i و وزیر j نباید در یک ستون باشند:

$$col[i] \neq col[j]$$

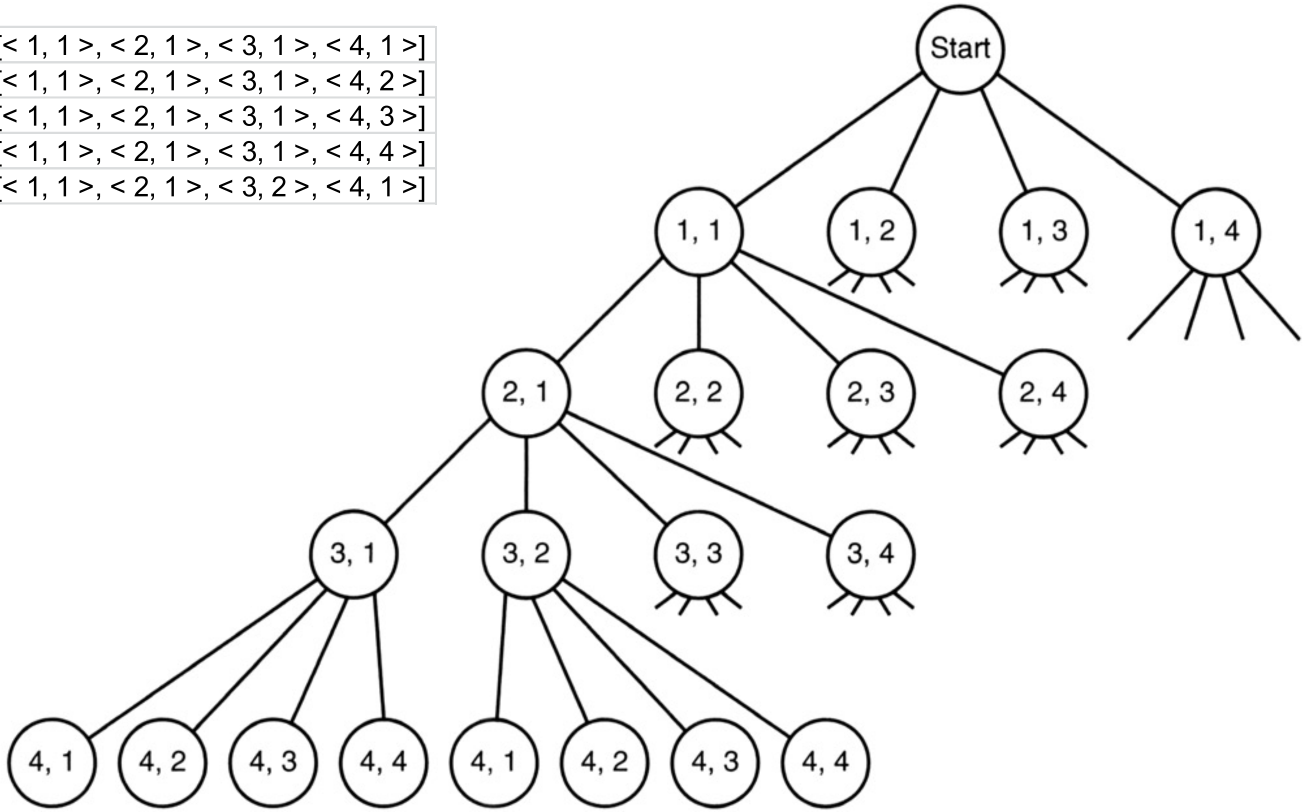
- وزیر i و وزیر j نباید در یک قطر باشند:

$$|col[i] - col[j]| \neq |i - j|$$

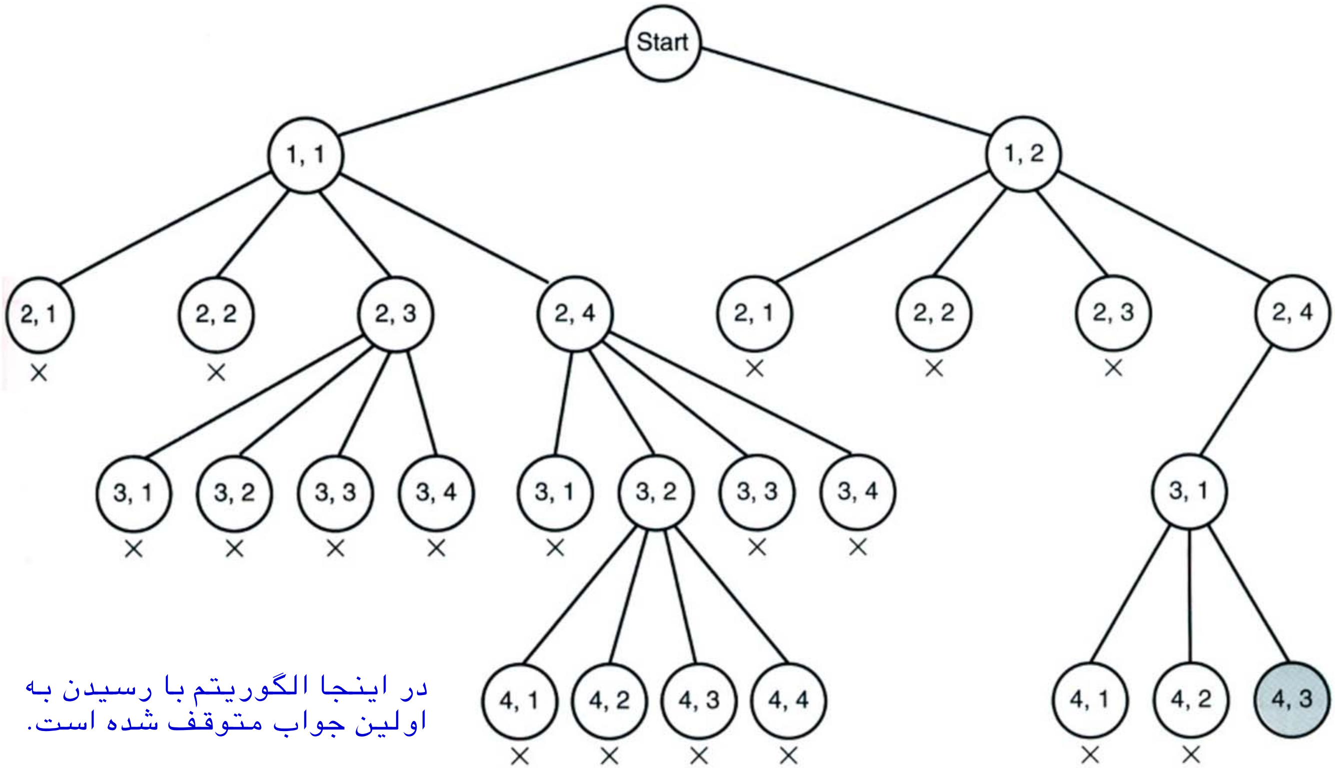


درخت جستجو: مسئله‌ی ۴-وزیر

[< 1, 1 >, < 2, 1 >, < 3, 1 >, < 4, 1 >]
[< 1, 1 >, < 2, 1 >, < 3, 1 >, < 4, 2 >]
[< 1, 1 >, < 2, 1 >, < 3, 1 >, < 4, 3 >]
[< 1, 1 >, < 2, 1 >, < 3, 1 >, < 4, 4 >]
[< 1, 1 >, < 2, 1 >, < 3, 2 >, < 4, 1 >]

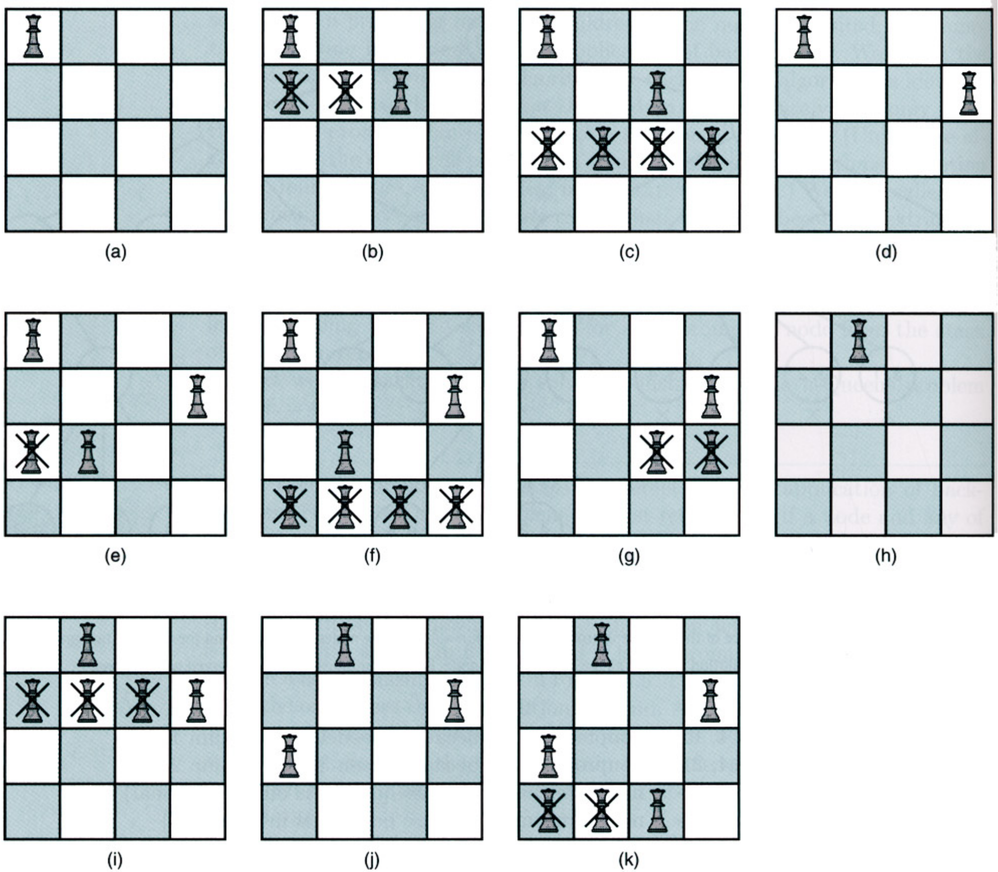


درخت جستجوی هرس شده: مسئله‌ی ۴ـ وزیر

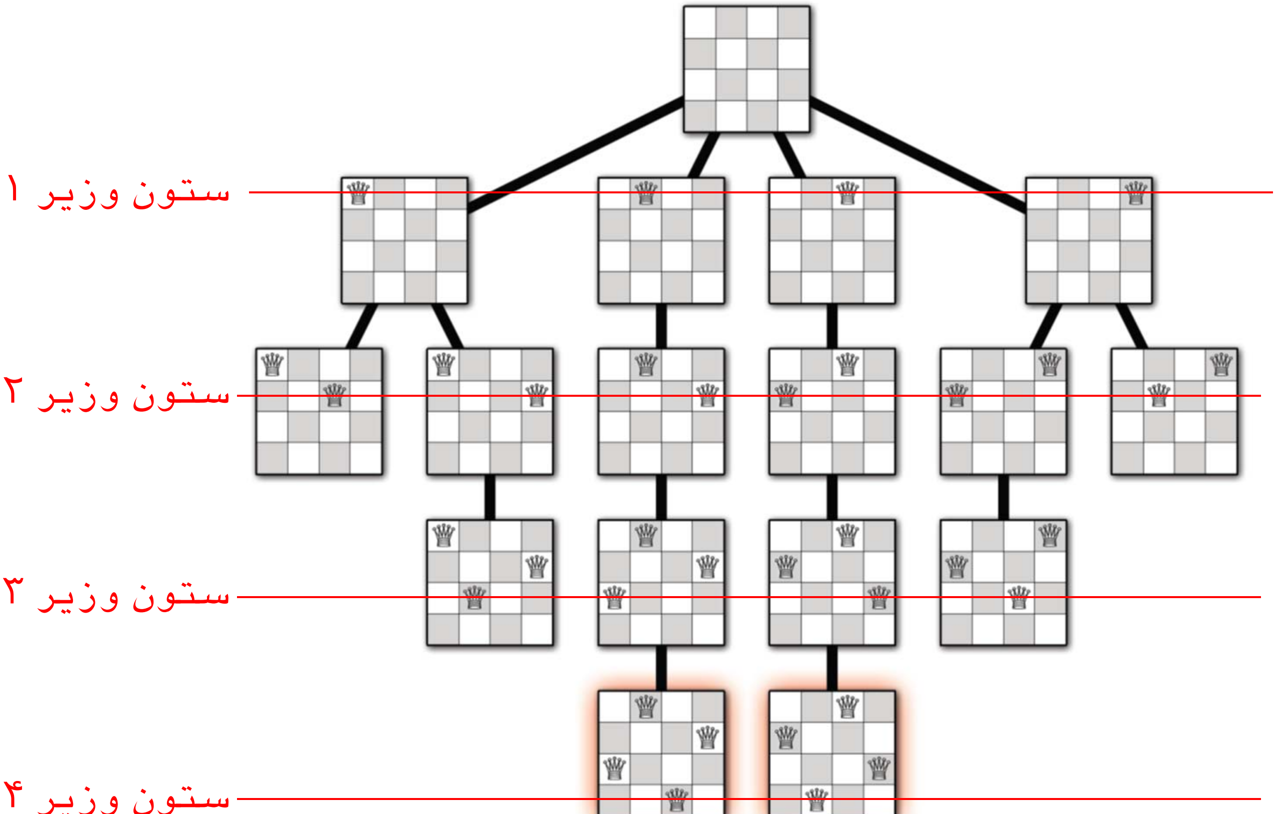


در اینجا الگوریتم با رسیدن به اولین جواب متوقف شده است.

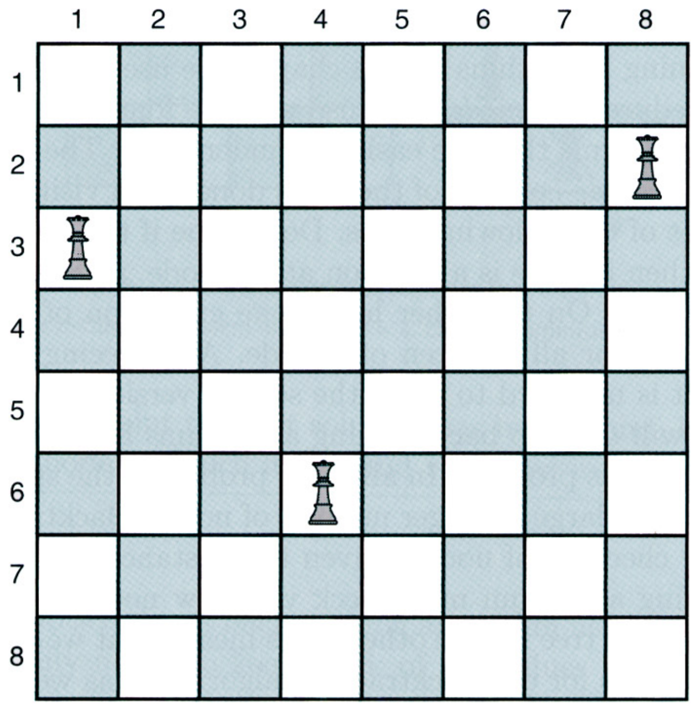
مراحل رسیدن به جواب در مسئله‌ی ۴ـ وزیر



درخت جستجوی هرس شده: مسئله‌ی ۴ـ وزیر



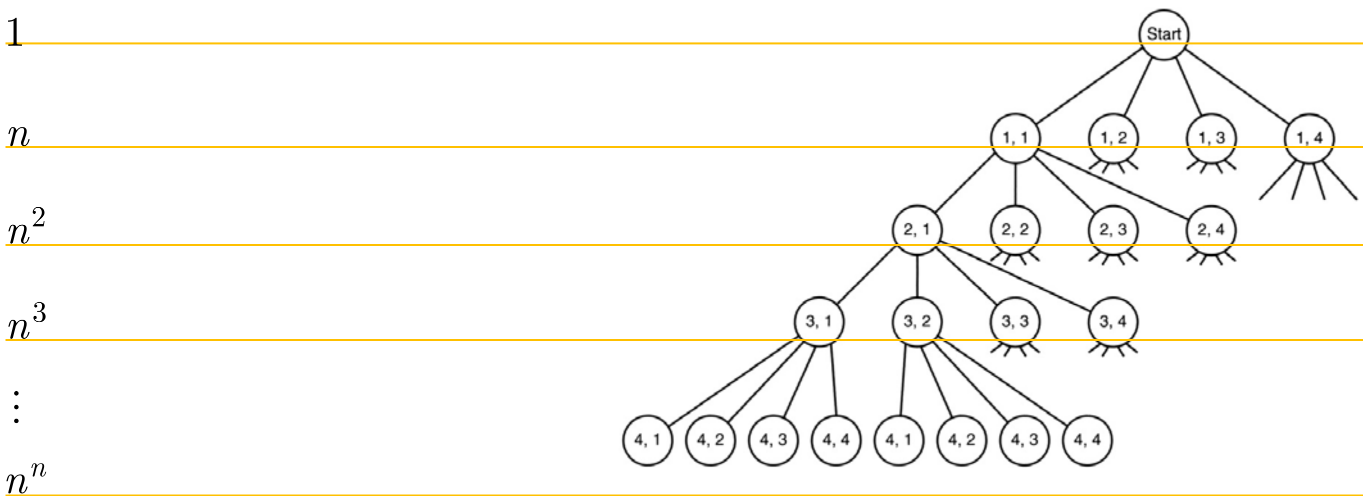
مسئله‌ی ۸-وزیر



مسئله‌ی n ـ وزیر: زمان اجرا

N-QUEENS PROBLEM

زمان اجرا برابر است با تعداد گره‌های بررسی شده در درخت جستجو



تعداد کل:

$$T(n) \leq 1 + n + n^2 + n^3 + \cdots + n^n \in O(n^{n+1})$$

مسئله‌ی n ـ وزیر: زمان اجرا

N-QUEENS PROBLEM

زمان اجرا برابر است با **تعداد گره‌های** بررسی شده در درخت جستجو

1

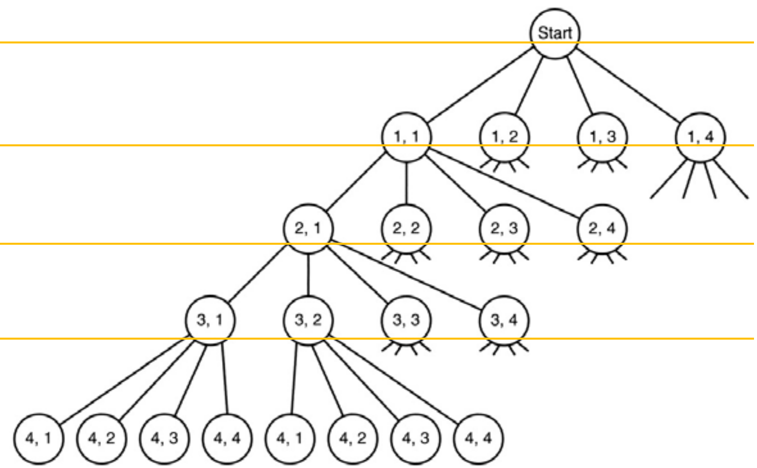
n

$n(n-1)$

$n(n-1)(n-2)$

$\vdots$

$n!$



تعداد کل (با فرض اینکه هیچ دو وزیری در یک ستون قرار ندارند):

$$T(n) \leq 1 + n + n(n-1) + n(n-1)(n-2) + \cdots + n! \in O(n!)$$

مسئله‌ی حاصل‌جمع زیرمجموعه‌ها

SUBSET SUM PROBLEM

مجموعه‌ی S با n عضو از اعداد صحیح و عدد صحیح W داده شده است.
می‌خواهیم همه‌ی زیرمجموعه‌های S را بیابیم (A) که حاصل‌جمع عناصر آنها W است.

$$A \subseteq S$$
$$\sum_{a \in A} a = W$$

مثال:

$$W = 6$$
$$S = \{w_1, w_2, w_3\} \quad , w_1 = 2, w_2 = 4, w_3 = 5$$
$$n = 3$$

$$\boxed{A = \{w_1, w_2\}}$$

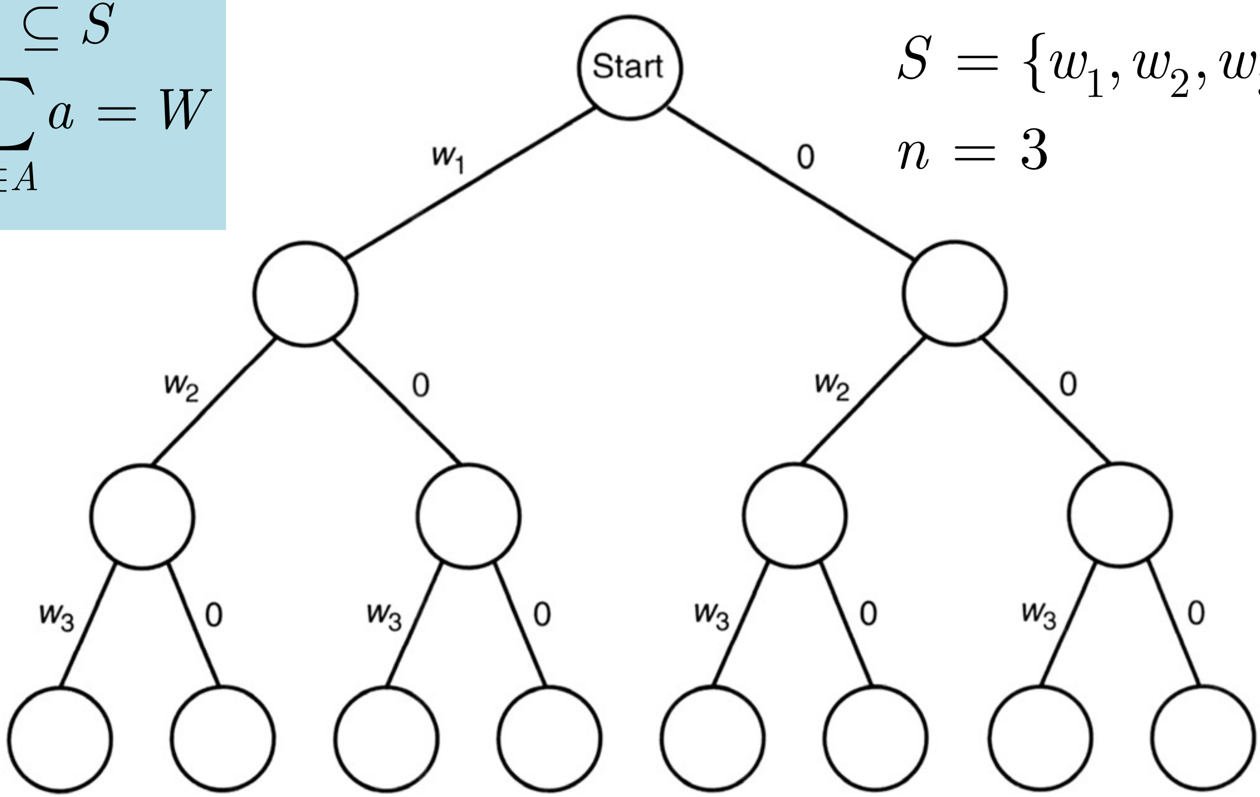
مسئله‌ی حاصل‌جمع زیرمجموعه‌ها: مثال

$$A \subseteq S$$

$$\sum_{a \in A} a = W$$

$$S = \{w_1, w_2, w_3\}$$

$$n = 3$$



مسئله‌ی حاصل‌جمع زیرمجموعه‌ها: مثال

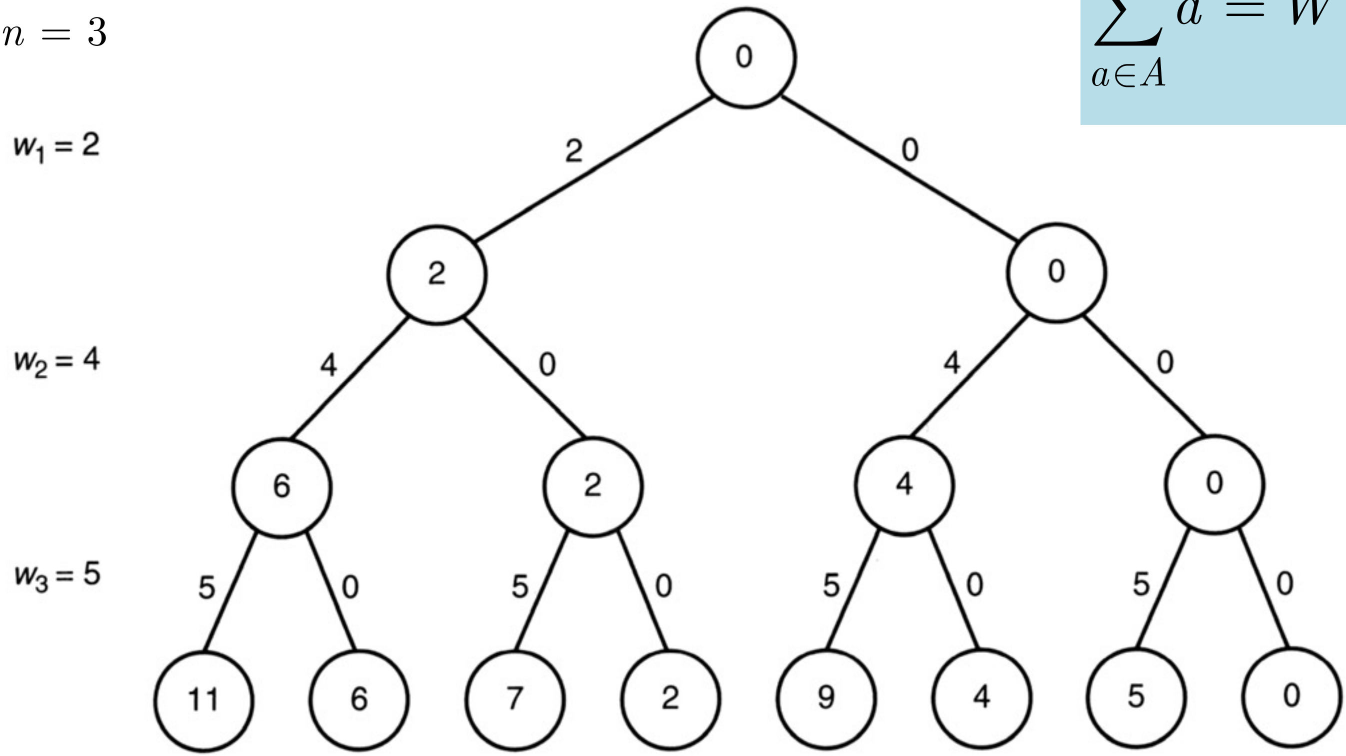
$W = 6$

$S = \{w_1, w_2, w_3\} \quad , w_1 = 2, w_2 = 4, w_3 = 5$

$n = 3$

$$A \subseteq S$$

$$\sum_{a \in A} a = W$$



$A = \{w_1, w_2\}$

مسئله‌ی حاصل‌جمع زیرمجموعه‌ها: مثال

$A \subseteq S$

$$\sum_{a \in A} a = W$$

$W = 13$

$S = \{w_1, w_2, w_3, w_4\}$

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

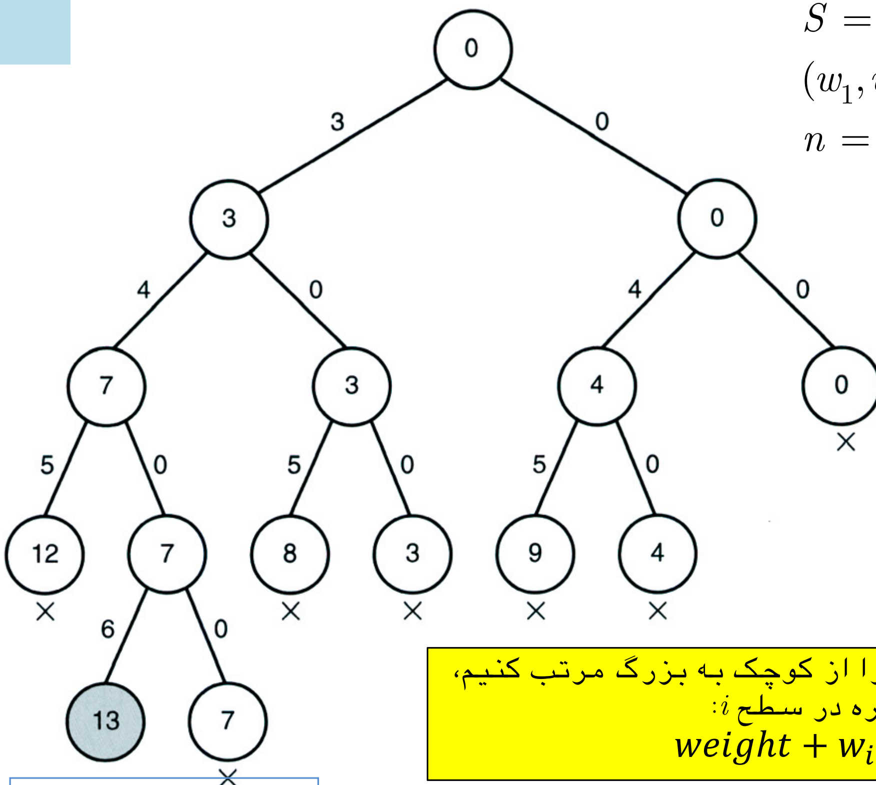
$n = 4$

$w_1 = 3$

$w_2 = 4$

$w_3 = 5$

$w_4 = 6$



$A = \{w_1, w_2, w_4\}$

اگر قبل از جستجو وزن‌ها را از کوچک به بزرگ مرتب کنیم،
شرط امیدبخش بودن یک گره در سطح i:

$$weight + w_{i+1} \leq W$$

مسئله‌ی حاصل‌جمع زیرمجموعه‌ها: زمان اجرا

زمان اجرا برابر است با تعداد گره‌های بررسی شده در درخت جستجو

1

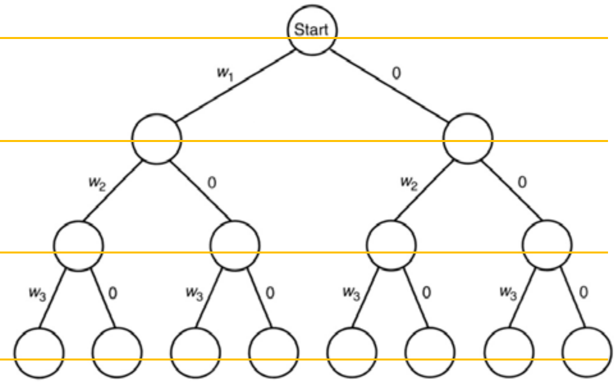
2

2^2

2^3

$\vdots$

2^n



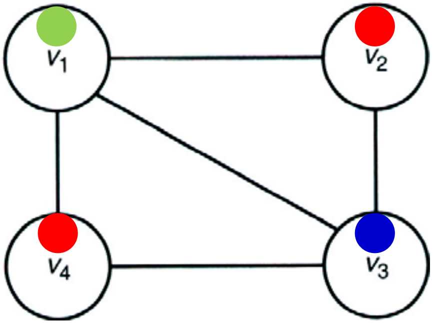
تعداد کل:
$$T(n) \leq 1 + 2 + 2^2 + 2^3 + \cdots + 2^n \in O(2^{n+1})$$

مسئله‌ی رنگ‌آمیزی گراف

GRAPH M-COLORING

یک گراف با n رأس داریم.
می‌خواهیم آن را با حداکثر m رنگ، رنگ‌آمیزی کنیم؛
به طوری که هیچ دو رأس مجاوری هم‌رنگ نباشند.

Vertex	*Color*
v_1	color 1
v_2	color 2
v_3	color 3
v_4	color 2

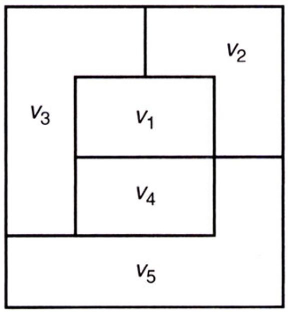


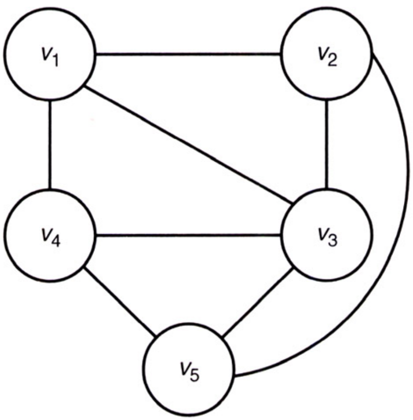
هدف یافتن همه‌ی رنگ‌آمیزی‌های ممکن است.

رنگ‌آمیزی نقشه با رنگ‌آمیزی گراف مسطح متناظر با آن

یکی از کاربردهای رنگ‌آمیزی گراف:

رنگ‌آمیزی یک نقشه به طوری که نواحی مجاور، همرنگ نباشند.

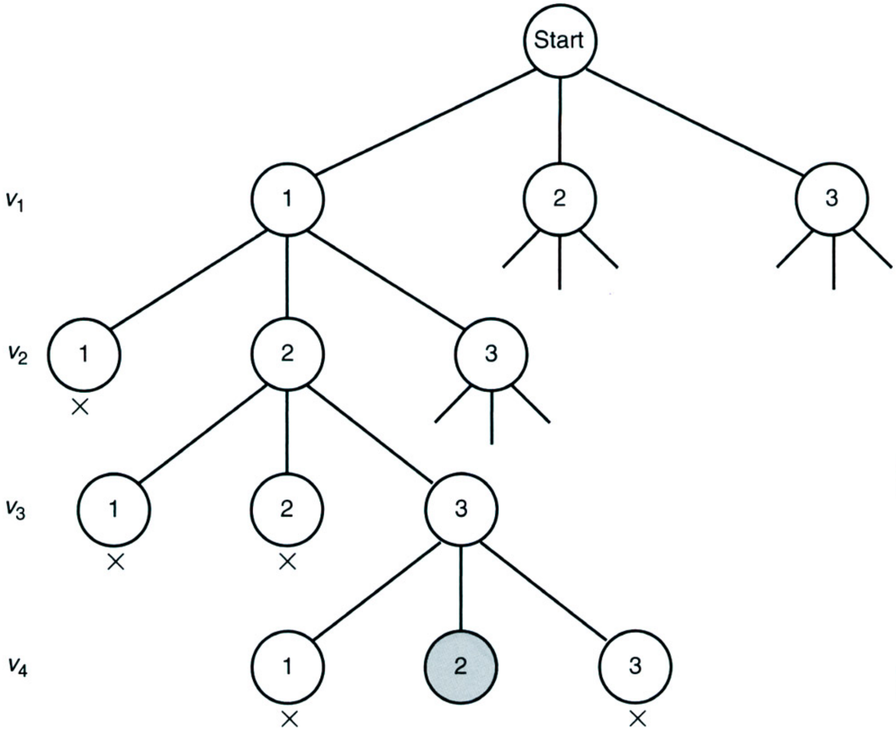




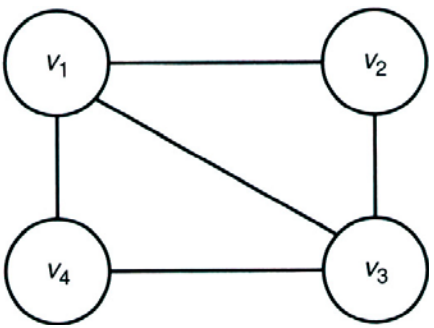
مسئله‌ی رنگ‌آمیزی گراف

مثال

در هر سطح، یکی از رئوس را رنگ‌آمیزی می‌کنیم:



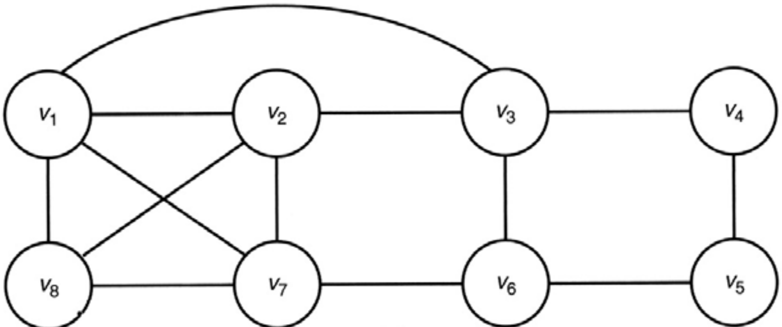
$$\boxed{m = 3}$$



مسئله‌ی رنگ‌آمیزی گراف

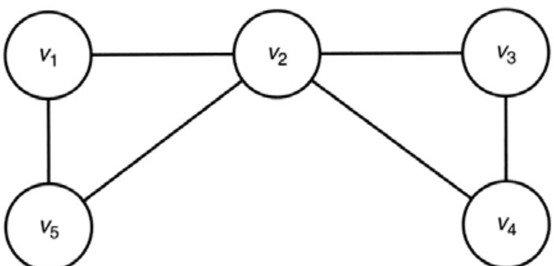
مثال

گراف زیر را با ۴ رنگ، رنگ‌آمیزی کنید:



(a)

گراف زیر را با ۳ رنگ، رنگ‌آمیزی کنید:



(b)

مسئله‌ی رنگ‌آمیزی گراف: زمان اجرا

زمان اجرا برابر است با تعداد گره‌های بررسی شده در درخت جستجو

1

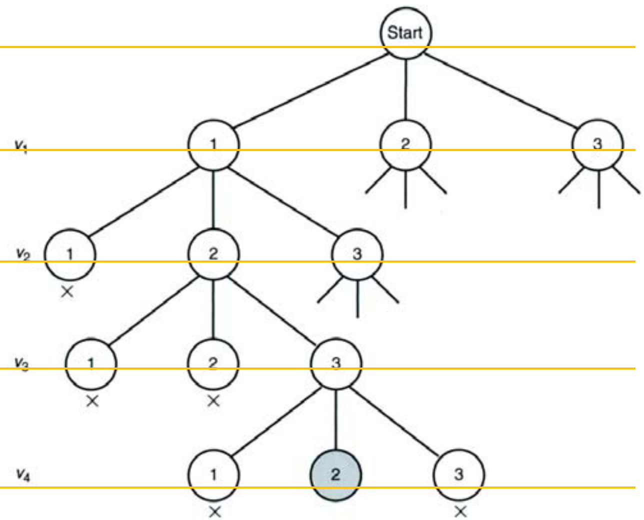
m

m^2

m^3

$\vdots$

m^n



تعداد کل: $T(n,m) \leq 1 + m + m^2 + m^3 + \cdots + m^n \in O(m^{n+1})$

مسئله‌ی کوله‌پشتی 0/1

0/1- KNAPSACK

قطعه	وزن	ارزش
i	w_i	p_i
1	2	40$
2	5	30$
3	10	50$
4	5	10$

n قطعه با وزن‌ها و ارزش‌های مختلف داریم.
یک کوله‌پشتی با ظرفیت W موجود است.
می‌خواهیم قطعه‌هایی را انتخاب کنیم که
- کوله‌پشتی تا حد امکان پر شود، و
- مجموع ارزش کل ظرف ماکزیمم شود.

راه حل با استفاده از تغییریافته‌ی الگوریتم عقب‌گرد:
یک متغیر مانند best بهترین مقدار تابع هزینه تا آن گره را در خود نگه می‌دارد.

مسئله‌ی کوله‌پشتی 0/1

روش عقب‌گرد:
قطعات را به ترتیب نزولی نسبت ارزش به وزن در هر مرحله در نظر می‌گیریم.
در هر سطح درخت، یک قطعه را بررسی می‌کنیم.
به روش حریصانه، قطعات را برمی‌داریم:

- ارزش آن را به $profit$ می‌افزاییم (منفعت کل)
- وزن آن را به $totalweight$ می‌افزاییم (وزن کل)

تا به قطعه‌ای برسیم که اگر آن را برداریم، $totalweight > W$ شود.

$$weight = \sum_{j=1}^{i} w_j \qquad profit = \sum_{j=1}^{i} p_j$$

اگر گره در سطح i ام باشد و
گره‌ای که باعث تجاوز حاصلجمع وزن‌ها از W می‌شود در سطح k باشد،

$$total\,weight = weight + \sum_{j=i+1}^{k-1} w_j$$

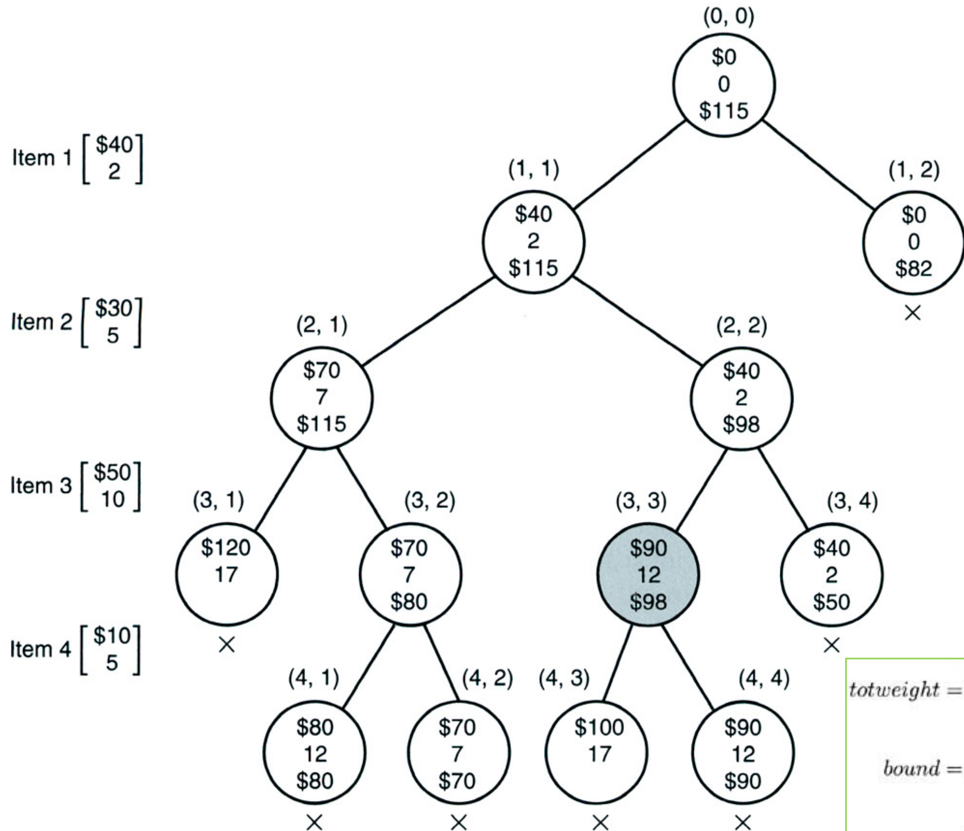
کران منفعت = bound

$$bound = profit + \sum_{j=i+1}^{k-1} p_j + (W - total\,weight) \times \frac{p_k}{w_k}$$

بهره‌ی حاصل از k-1 قطعه‌ی نخست + ظرفیت موجود برای قطعه‌ی kام × ارزش واحد وزن برای قطعه‌ی kام

اگر در یک گره مقدار $bound$ از بیشترین منفعت تا آن گره کمتر بود، آن گره امیدبخش نیست.

مسئله‌ی کوله‌پشتی 0/1



قطعه	وزن	ارزش	نسبت
i	w_i	p_i	p_i / w_i
1	2	40$	20$
2	5	30$	6$
3	10	50$	5$
4	5	10$	2$

$$W = 16, \quad n = 4$$

منفعت کل

وزن کل

کران منفعت

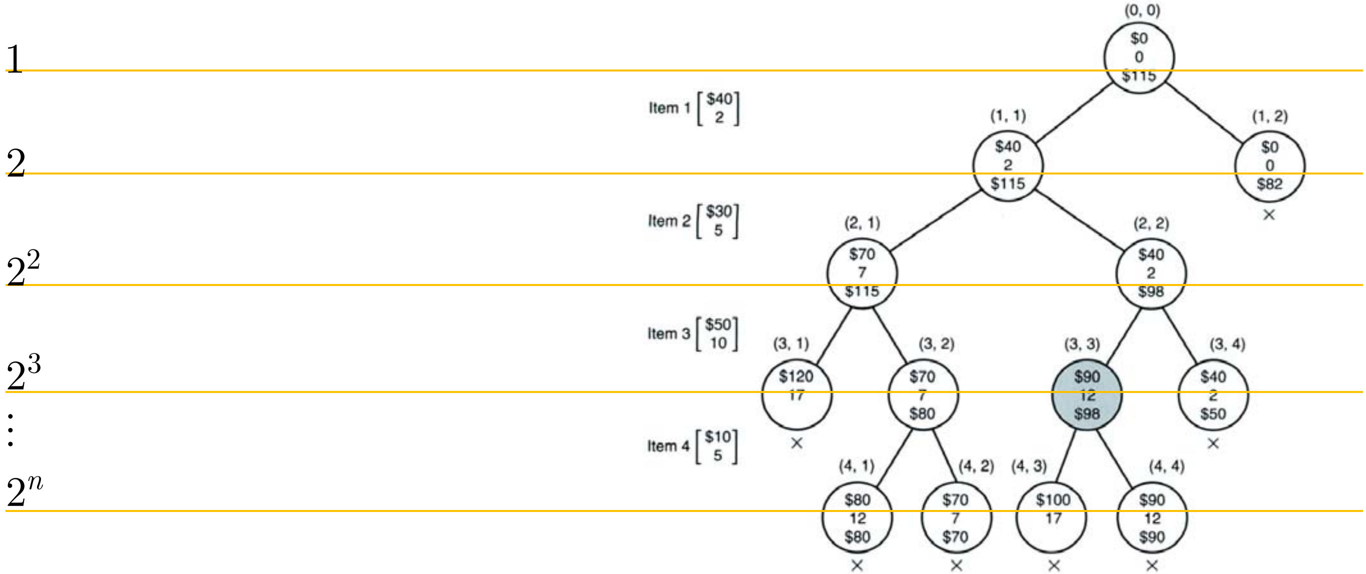
$$totweight = weight + \sum_{j=1+1}^{3-1} w_j = 2 + 5 = 7$$

$$bound = profit + \sum_{j=1+1}^{3-1} p_j + (W - totweight) \times \frac{p_3}{w_3}$$

$$= \$40 + \$30 + (16 - 7) \times \frac{\$50}{10} = \$115.$$

مسئله‌ی کوله‌پشتی 0/1: زمان اجرا

زمان اجرا برابر است با تعداد گره‌های بررسی شده در درخت جستجو



تعداد کل:
$$T(n) \leq 1 + 2 + 2^2 + 2^3 + \cdots + 2^n \in O(2^{n+1})$$