مبانی و کاربردهای هوش مصنوعی

Artificial Intelligence: Basics & Applications

فصل سوم:
حل مسئله با جستجو
(جستجو ناآگاهانه)

مدرس:
دکتر حسام عمران‌پور

حل تمرین‌ها:
محسن امیری | آرش حاجیان‌نژاد

لینک‌ها:
کلاس کوئرا | کانال تلگرام

عامل حل مسئله (عامل جستجوگر)

- این عامل، یک عامل هدفگراست که سعی دارد با استفاده از عملیات جستجو، دنباله‌ای از فعالیت‌ها برای رسیدن به وضعیت هدف پیدا کند.

- در ادامه‌ی این فصل، فرض می‌کنیم محیط مسئله کاملا مشاهده‌پذیر، قطعی، ایستا و گسسته است. همچنین فرض می‌کنیم که عامل جستجوگر، وضعیت اولیه محیط را می‌داند.

فرموله‌سازی مسئله

حالت اولیه: وضعیت اولیه محیطی که عامل در آن فعالیت خواهد کرد.

فعالیت‌ها: کارهایی که عامل می‌تواند در هر لحظه و با توجه به شرایط فعلی محیط انجام دهد.

تابع جانشین(Successor Function): تابعی که در هر وضعیت دلخواه، وضعیت محیط عامل پس از انجام یک فعالیت را مشخص می‌کند.

جانشین: به هر وضعیتی که از وضعیت فعلی محیط و با انجام فقط یک فعالیت بتوان به آن رسید، جانشین وضعیت فعلی می‌گویند.

تابع آزمون هدف(Goal Test): این تابع با گرفتن یک وضعیت محیط به عنوان ورودی، مشخص می‌کند این وضعیت، وضعیت هدف است یا خیر.

فرموله‌سازی مسئله

فضای حالت(State Space):

به مجموعه تمامی حالت‌های محیط که با انجام فعالیت‌های ممکن، می‌توان از حالت اولیه به آن‌ها رسید، فضای حالت مسئله می‌گویند.

گراف فضای حالت: گرافی است که در آن، هر وضعیت محیط به عنوان یک راس در نظر گرفته شده و بین هر حالت و جانشین‌های آن، یک یال رسم شده است.

حل مسئله: برای حل مسئله، لازم است در گراف فضای حالت، مسیری بین حالت اولیه و وضعیت مقصد پیدا کنیم.

فرموله‌سازی مسئله

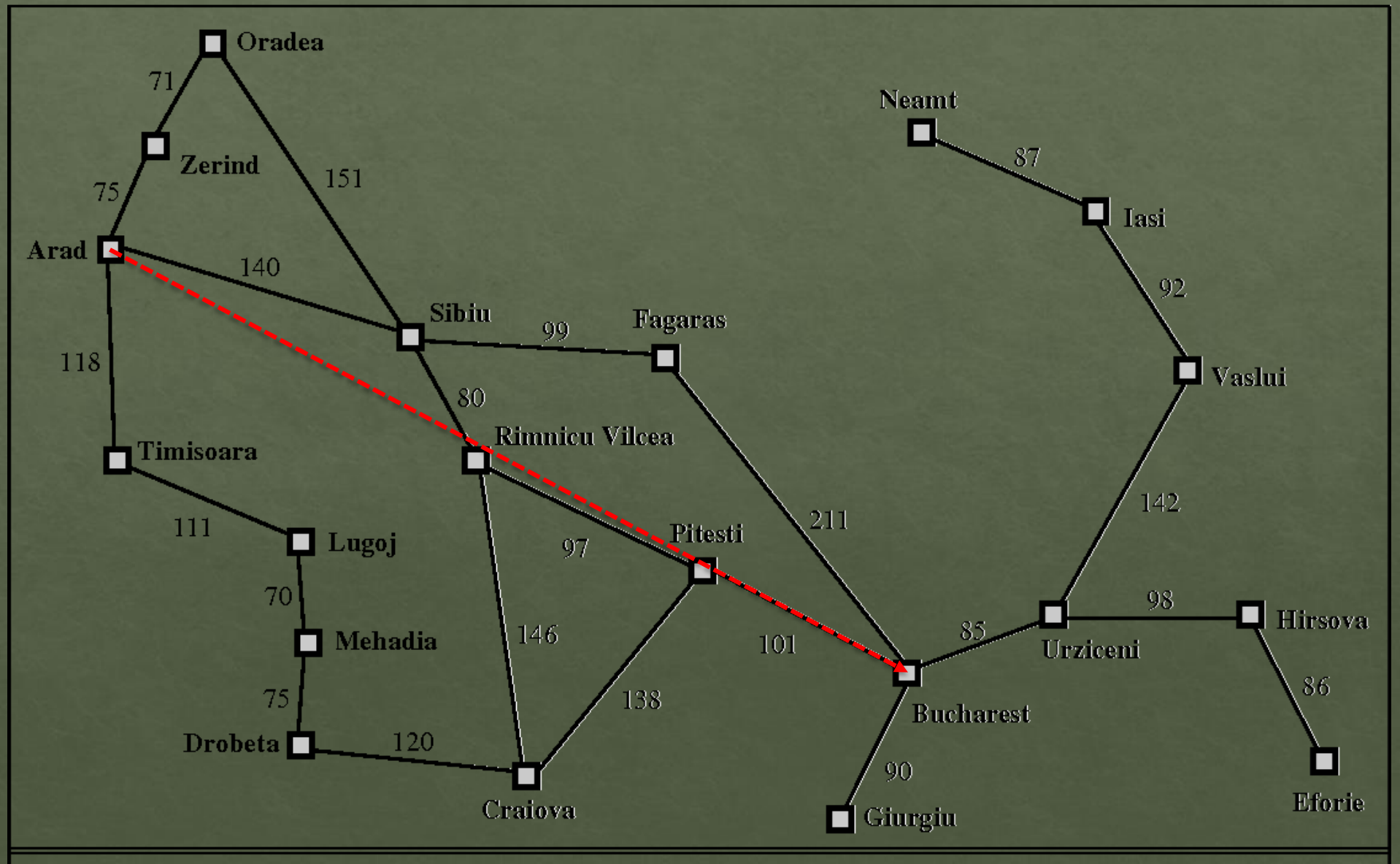
تابع هزینه مسیر (Path-cost Function): این تابع مسیر بین دو وضعیت محیط را به عنوان ورودی گرفته و هزینه آن مسیر را برمیگرداند.

هزینه مرحله (Step Cost): هر مسیر در گراف فضای حالت با دنباله‌ای از عملیات متناظر است که با انجام آن‌ها از وضعیتی به وضعیت دیگر می‌رویم. در نتیجه، هزینه هر مسیر با مجموع هزینه‌های هر یک از عملیات موجود در آن مسیر برابر است. هزینه هر یک از عمل‌های عامل را هزینه آن مرحله نامیده می‌شود.

- هزینه رفتن از وضعیت x به وضعیت y با گذر از مسیر a را به صورت $c(x,a,y)$ نمایش می‌دهند.
- هزینه رفتن از حالت اولیه به وضعیت n را به صورت $g(n)$ نمایش می‌دهند.

راه حل بهینه: مسیری از حالت اولیه به وضعیت هدف که کمترین هزینه ممکن را داشته باشد.

مثال: فرموله‌سازی مسئله سفر در رومانی (Arad به Bucharest)



مثال: فرموله‌سازی مسئله سفر در رومانی (Arad به Bucharest)

- **حالت اولیه:** شخص (عامل) در شهر Arad قرار دارد.
- **حالت مقصد:** عامل قصد سفر به شهر Bucharest را دارد.
- **تابع جانشین:** این تابع مشخص می‌کند از هر شهر به چه شهرهایی می‌توان رفت. مثال:

successor(in_Arad) = {Goto(Zerind), in_Zerind},
{Goto(Sibiu), in_Sibiu},
{Goto(Timisora), in_Timisoara}

- **تابع آزمون هدف:** این تابع، مقصد (بخارست) بودن شهر ورودی را مشخص می‌کند.

goal_test(in_Fagaras) = False

- **تابع هزینه مسیر:** خروجی این تابع، مجموع هزینه‌های یک مسیر را برمی‌گرداند.

cost(Timisoara, <Lugoj, Mehadia, Drobeta>, Craiova) = 111 + 70 + 75 + 120 = 376

مثال: فرموله‌سازی مسئله پازل ۸

حالت اولیه:

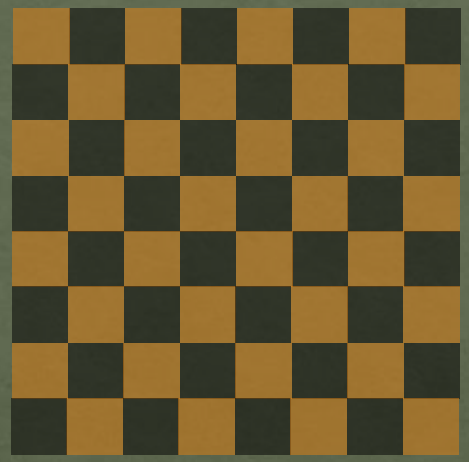
حالت مقصد:

- می‌خواهیم با جابجایی اعداد، از حالت اولیه به وضعیت مقصد برسیم.
- فعالیت‌های ممکن، به جابجا کردن مکان مربع خالی و مربع‌های آن می‌باشد.
- در این مسئله، فضای حالت مجموعه‌ی تمامی حالت‌هایی است که از حالت اولیه می‌توان به آن‌ها رسید. برای نمایش حالت در این مسئله، می‌توان از یک آرایه به طول ۹ استفاده کرد:

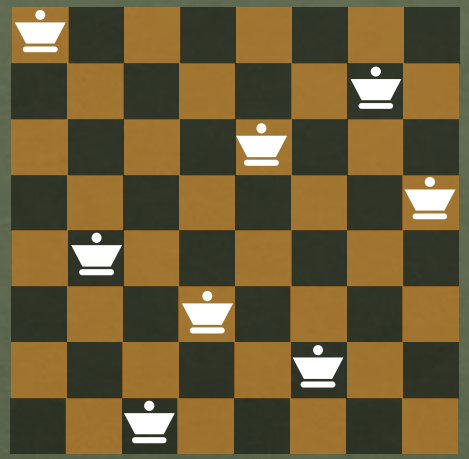
Goal State = <0, 1, 2, 3, 4, 5, 6, 7, 8>

مثال: فرموله‌سازی مسئله ۸ وزیر

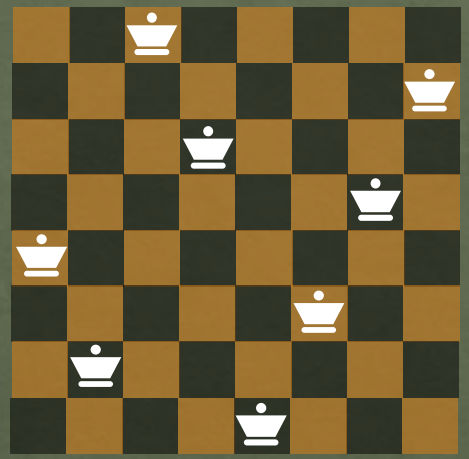
حالت اولیه:



حالت مقصد:



،



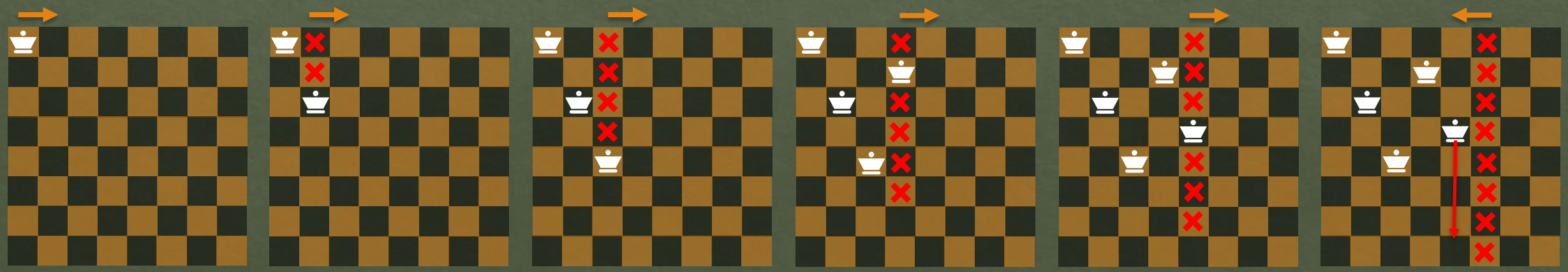
و ...

- در این مسئله می‌خواهیم ۸ مهره وزیر را به گونه‌ای بر روی یک صفحه ۸ در ۸ (صفحه شطرنج) بچینیم که هیچ یک از آن‌ها دیگری را تهدید نکند.
- در این مسئله هدف پیدا کردن یک حالت مقصد است و به مسیر رسیدن به هدف اهمیتی نمی‌دهیم. بنابراین، هزینه مسیر در این مسئله بی‌معنی بوده و صفر در نظر گرفته می‌شود.

مثال: فرموله‌سازی مسئله ۸ وزیر

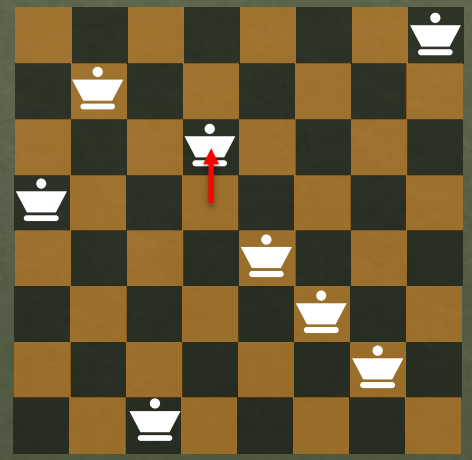
- به طور کلی، برای حل این مسئله می‌توان به ۲ روش عمل کرد:
  ۱. روش افزایشی
  ۲. روش چیدمان کامل

- **روش افزایشی:** در این روش، از ستون سمت چپ شروع کرده و در هر مرحله، یک وزیر را در یکی از خانه‌های آن ستون قرار می‌دهیم و این کار را به نوبت برای ستون‌های بعدی نیز انجام می‌دهیم. هرگاه به وضعیتی رسیدیم که در ستون جدید هیچ مکانی برای مهره وزیر وجود نداشته باشد، عقبگرد کرده و وزیر ستون قبل (یا در صورت نیاز قبل‌تر) را جابجا می‌کنیم.

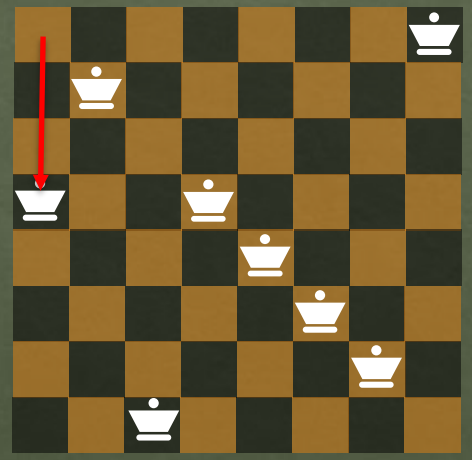


مثال: فرموله‌سازی مسئله ۸ وزیر

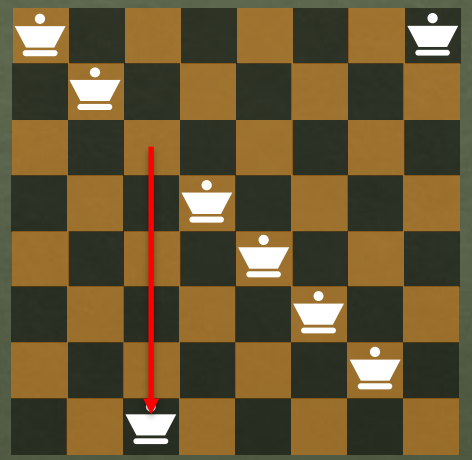
- **روش چیدمان کامل:** در این روش، در هر یک از ستون‌های صفحه، یک وزیر به دلخواه قرار می‌دهیم. سپس با جابجا کردن وزیرها در ستون خودشان، سعی می‌کنیم به وضعیت مقصد نزدیک شویم. این کار را ادامه می‌دهیم تا به وضعیت مقصد برسیم.



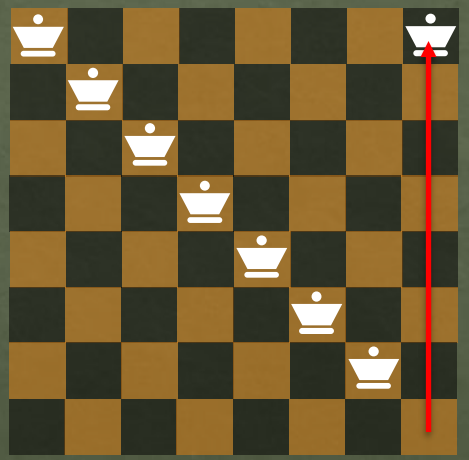
تهدیدها = ۶



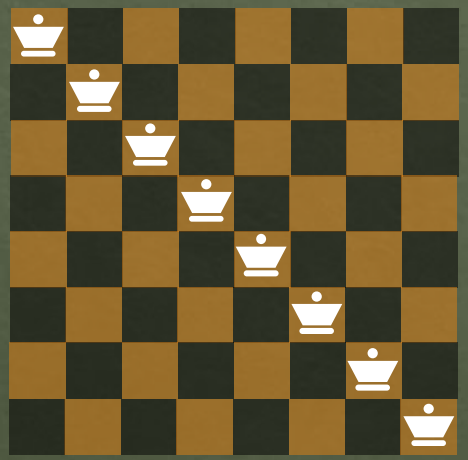
تهدیدها = ؟



تهدیدها = ۱۶



تهدیدها = ۲۲



تهدیدها = ۲۸

حل مسئله به کمک جستجو

درخت جستجو (Search Tree): نمایش درختی فضای حالت است.

- در این درخت، هر شاخه (یا یال) نشان‌دهنده‌ی یک فعالیت است. گره‌ها و برگ‌های این درخت نیز حالت‌های محیط را نشان می‌دهند.

- در الگوریتم‌های جستجو، در هر مرحله، یکی از برگ‌های درخت بسط داده شده و زیرشاخه‌های آن به درخت اضافه می‌شوند. همچنین اگر گره بسط داده شده، گره هدف باشد، جستجو پایان یافته و گره هدف به عنوان خروجی مشخص می‌شود.

- ترتیب انتخاب گره‌های بسط داده شده توسط راهبرد (یا استراتژی) جستجو مشخص می‌شود. برخی از این راهبردها در ادامه‌ی این فصل بررسی خواهند شد.

معیار سنجش الگوریتم‌های جستجو

- **کامل بودن:** الگوریتمی کامل است که در صورت وجود راه حل، آن را حتما پیدا کند.
- **بهینه بودن:** الگوریتمی بهینه است که همیشه راه حل بهینه را پیدا کند.
- **پیچیدگی زمانی:** مرتبه زمانی اجرای الگوریتم جستجو است. از آن‌جایی که برای جستجو از درخت جستجو استفاده می‌کنیم، پیچیدگی زمانی را معادل تعداد گره‌های تولید شده فرض می‌کینم.
- **پیچیدگی فضائی:** حداکثر حافظه‌ی لازم برای اجرای الگوریتم است که در درخت جستجو، آن را معادل حداکثر تعداد گره‌های ذخیره شده در حافظه می‌دانیم.

معیار سنجش الگوریتم‌های جستجو

- برای بیان پیچیدگی زمانی و فضائی از پارامترهای زیر کمک می‌گیریم:

❖ ماکزیمم فاکتور انشعاب (Branching Factor) که با **b** نمایش می‌دهیم.

❖ کمترین عمقی که در آن یک گره هدف وجود داشته باشد که آن را با **d** نمایش می‌دهیم.

❖ حداکثر عمق درخت جستجو که آن را با **m** نشان می‌دهیم.

راهبردهای جستجو

- به طور کلی ۲ راهبرد برای جستجو وجود دارد:
- **جستجوی ناآگاهانه** (کورکورانه) (Uninformed Search – Blind Search): الگوریتم هیچ اطلاعاتی مضاف بر صورت مسئله (فرموله‌سازی مسئله) در اختیار ندارد.
- **جستجوی آگاهانه** (اکتشافی) (Informed Search – Heuristic Search): الگوریتم جستجو علاوه بر صورت مسئله به اطلاعات کمک‌کننده‌ای دسترسی دارد. این اطلاعات به طور کلی میزان امیدبخشی یک گره را به الگوریتم ارائه می‌دهد.
- در ادامه این فصل به بررسی الگوریتم‌های جستجو ناآگاهانه می‌پردازیم.

جستجو سطحی (عرضی) (Breadth First Search)

- این الگوریتم گره‌های درخت را به صورت سطح به سطح، از بالا به پایین، بسط می‌دهد. به عبارت دیگر این الگوریتم از بین گره‌های حاشیه (برگ‌های قسمتی از درخت جستجو که تا الان بسط داده شده)، هربار کم‌عمق‌ترین گره را بررسی کرده و بسط می‌دهد.

- برای این الگوریتم طبق قرارداد داریم:
  ۱. در صورتی که، در یک مرحله از جستجو، کم‌عمق‌ترین گره یکتا نباشد، چپ‌ترین گره بسط داده می‌شود.
  ۲. فرزندان هر گره، در صورت امکان، از چپ به راست به ترتیب حروف الفبا یا به صورت صعودی قرار می‌گیرند.

ارزیابی الگوریتم BFS

- **کامل بودن:** کامل است. BFS درخت جستوجو را سطح به سطح از بالا به پایین پیمایش می‌کند، گره جواب در هر سطحی باشد، الگوریتم بالاخره به آن می‌رسد. (مگر زمانی که b بی‌نهایت باشد، یعنی هرگره بتواند بی‌نهایت فرزند داشته باشد. چون در سطحی که بی‌نهایت فرزند دارد گیر می‌کند و هیچ‌گاه به سطح‌های دیگر نمی‌رسد.)

- **بهینه بودن:** اگر هزینه تمام عملیات یکسان باشد جواب بهینه را پیدا می‌کند. (جستوجو سطحی همواره کم‌عمق‌ترین جواب را پیدا می‌کند، لذا وقتی هزینه تمام عملیات یکسان باشد، کم‌عمق‌ترین جواب همان کم‌هزینه‌ترین جواب خواهد بود.)

- **پیچیدگی زمانی:** $O(b^{d+1})$ **پیچیدگی فضائی:** $O(b^{d+1})$

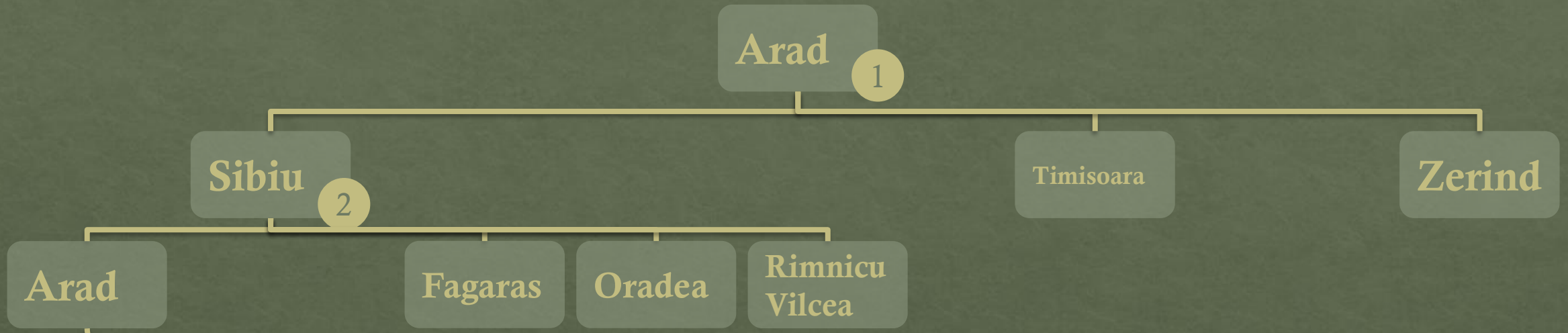
- پیچیدگی‌های این الگوریتم از مرتبه نمایی هستند که خوب نیست. مشکل اصلی، پیچیدگی فضایی این الگوریتم است، زیرا بعد از گذشت زمان اندکی از شروع جستجو، حافظه مورد نیاز این الگوریتم از حافظه موجود بیشتر می‌شود.

مثال: الگوریتم جستجو سطحی در مسئله سفر در رومانی

Arad

مثال: الگوریتم جستجو سطحی در مسئله سفر در رومانی

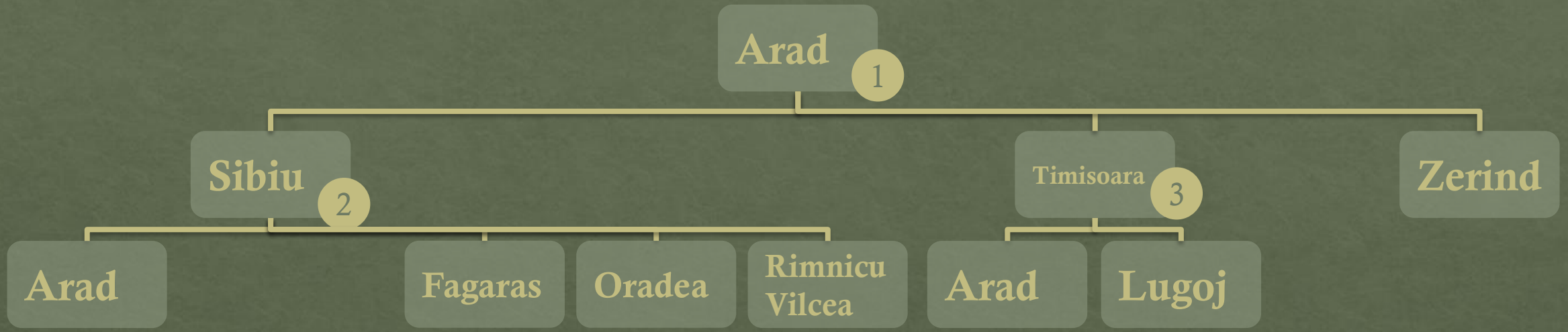
مثال: الگوریتم جستجو سطحی در مسئله سفر در رومانی



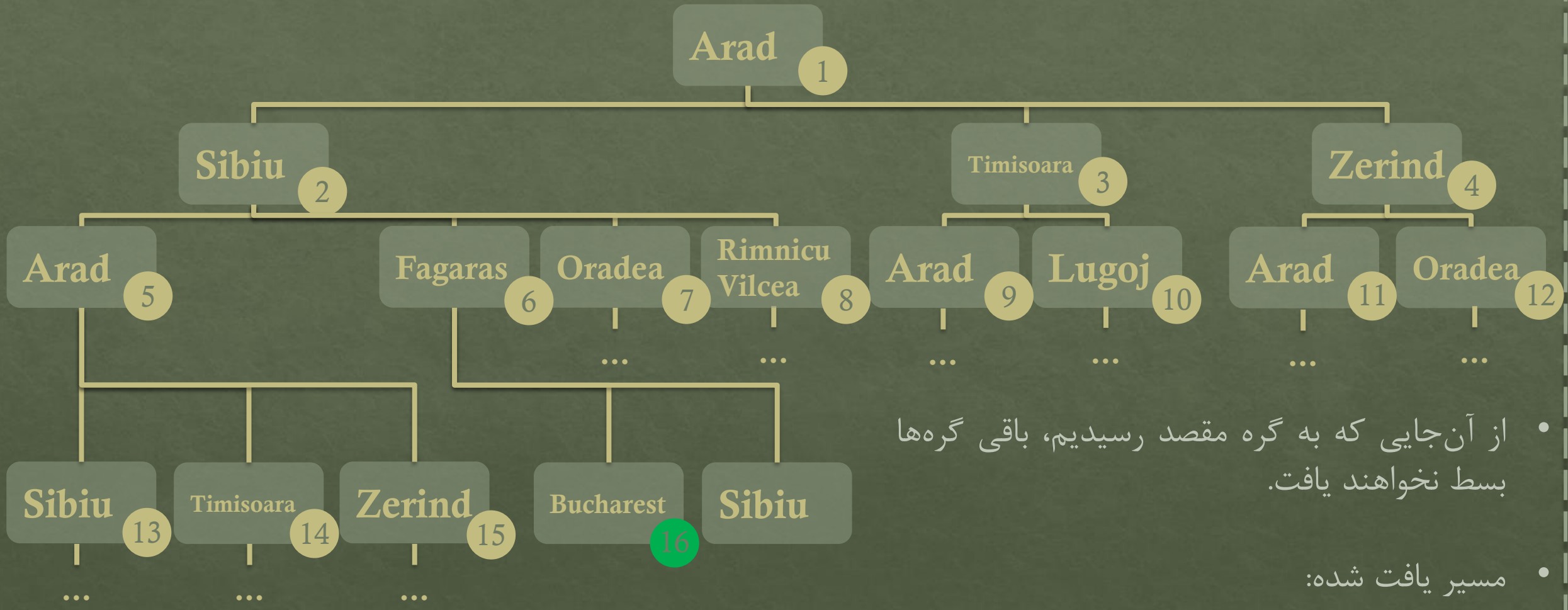
- در الگوریتم جستجوی سطحی، تکراری بودن گره‌ها کنترل نمی‌شود.

- به طور کلی در الگوریتم‌های جستجوی درختی، تکراری بودن گره‌ها کنترل نمی‌شود.

مثال: الگوریتم جستجو سطحی در مسئله سفر در رومانی



مثال: الگوریتم جستجو سطحی در مسئله سفر در رومانی



- از آن‌جایی که به گره مقصد رسیدیم، باقی گره‌ها بسط نخواهند یافت.
- مسیر یافت شده:

Arad → Sibiu → Fagaras → Bucharest

جستجو عمقی (Depth First Search)

- این الگوریتم در هر مرحله از جستجو، از بین گره‌های برگ قسمت بسط‌داده شده‌ی درخت، همواره عمیق‌ترین گره را بسط می‌دهد.

ارزیابی الگوریتم DFS:

- **کامل بودن:** کامل نیست زیرا ممکن است در یک حلقه بی‌نهایت از گره‌های تکراری گیر کند. (معایب)
- **بهینه بودن:** در صورت پیدا کردن راه حل، ممکن است راه حل همچنان بهینه نباشد و راه حلی در عمق کمتری وجود داشته باشد.

- **پیچیدگی زمانی:** $O(b^m)$ **پیچیدگی فضائی:** $O(bm)$ (مزایا)

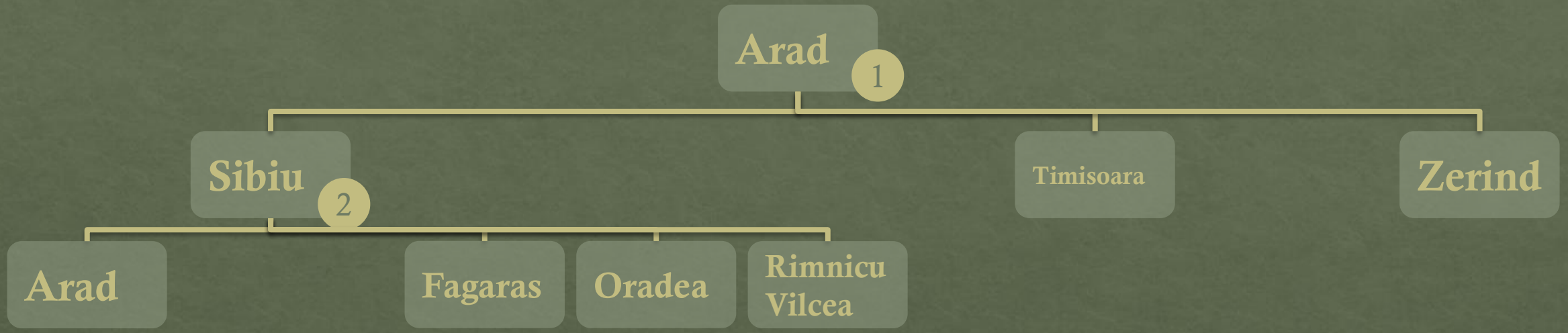
- ویژگی مثبت این الگوریتم پیچیدگی فضائی خطی آن است.

مثال: الگوریتم جستجو عمقی در مسئله سفر در رومانی

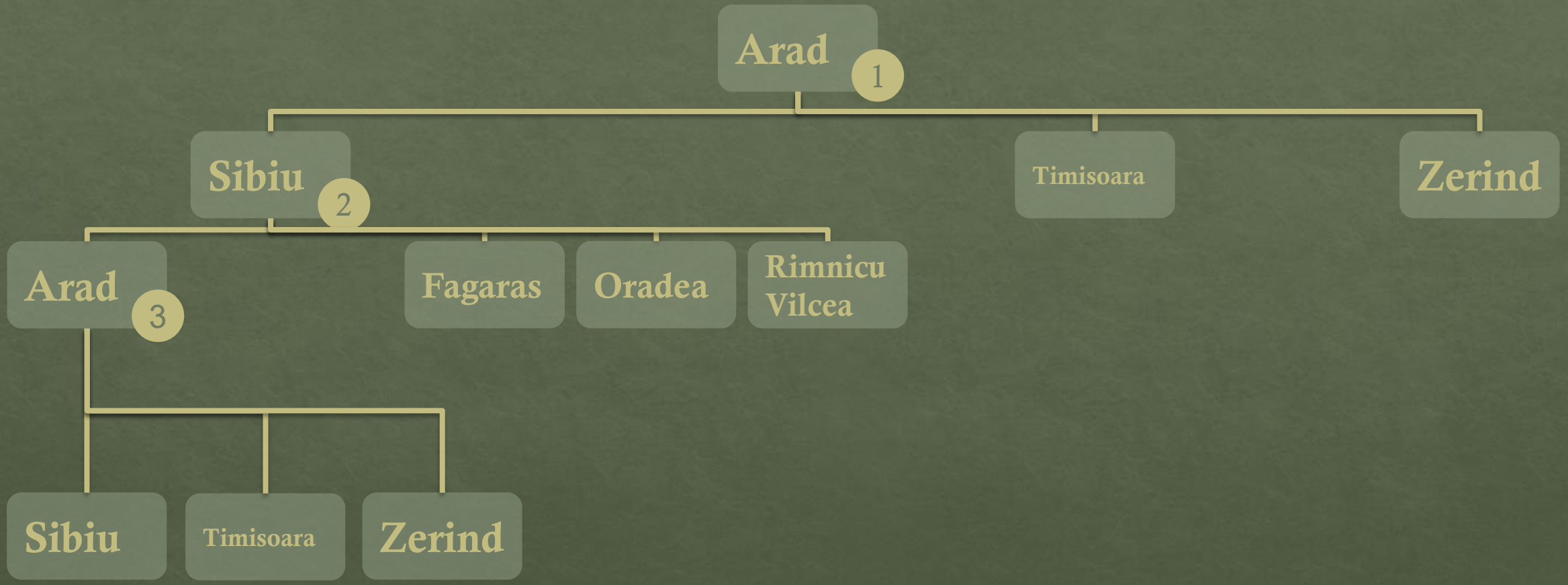
Arad

مثال: الگوریتم جستجو عمقی در مسئله سفر در رومانی

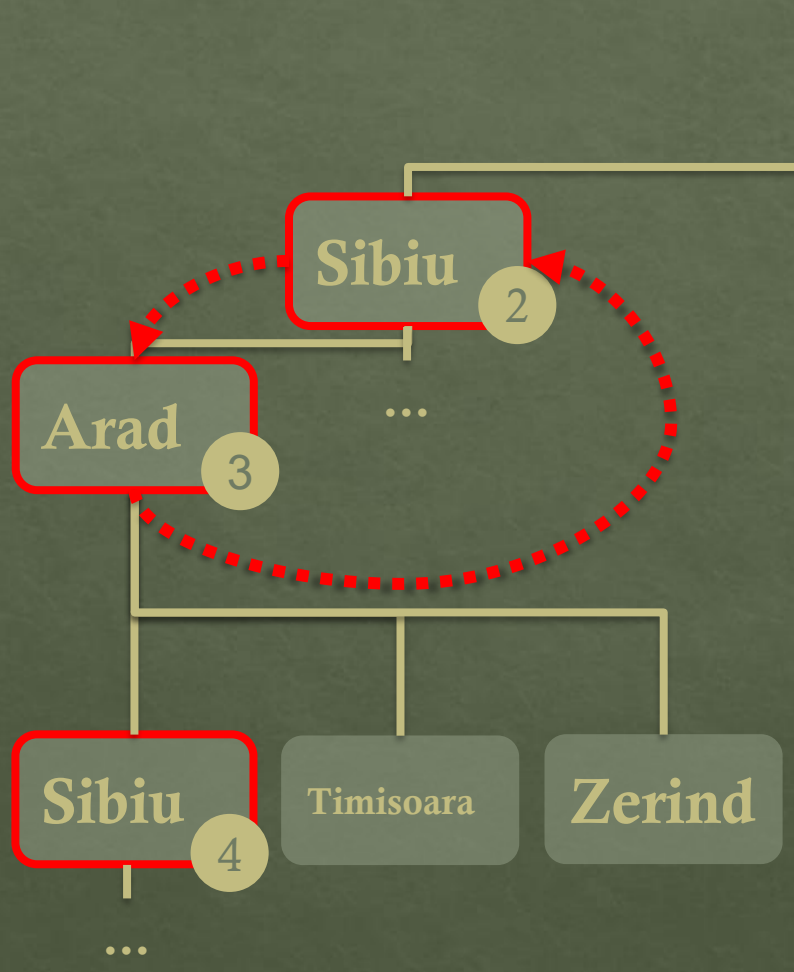
مثال: الگوریتم جستجو عمقی در مسئله سفر در رومانی



مثال: الگوریتم جستجو عمقی در مسئله سفر در رومانی



مثال: الگوریتم جستجو عمقی در مسئله سفر در رومانی



- از آن‌جایی که تکراری بودن گره‌ها کنترل نمی‌شود، در حلقه‌ی بی‌نهایتی بین Arad و Sibiu گیر می‌کنیم.

جستجو هزینه یکنواخت (Uniform Cost Search)

- در این الگوریتم، به هر گره در درخت جستجو یک هزینه نسبت داده می‌شود. هزینه هر گره، مجموع هزینه‌های عملیات انجام شده از ریشه درخت تا آن گره ($g(n)$) فرض می‌شود. سپس در هر مرحله، گرهی که کمترین هزینه را دارد بسط داده می‌شود.

- برای نگهداری گره‌های حاشیه در الگوریتم هزینه یکنواخت، از صف اولویت (گره باهزینه کمتر، اولویت بیشتری دارد و در صف، جلوتر قرار می‌گیرد) استفاده می‌شود.

- اگر هزینه تمام عملیات یکسان باشد، الگوریتم هزینه یکنواخت به الگوریتم جستجوی سطحی تبدیل می‌شود.

ارزیابی الگوریتم UCS

- **کامل بودن:** اگر هزینه هر عمل یک عدد مثبت باشد کامل است. (یعنی هزینه گره پدر همیشه از گره فرزند کمتر باشد.) اما اگر هزینه برخی از عملیات، صفر یا منفی باشد، کامل نیست.

- **بهینه بودن:** اگر هزینه هر عمل مثبت باشد، این الگوریتم بهینه است.

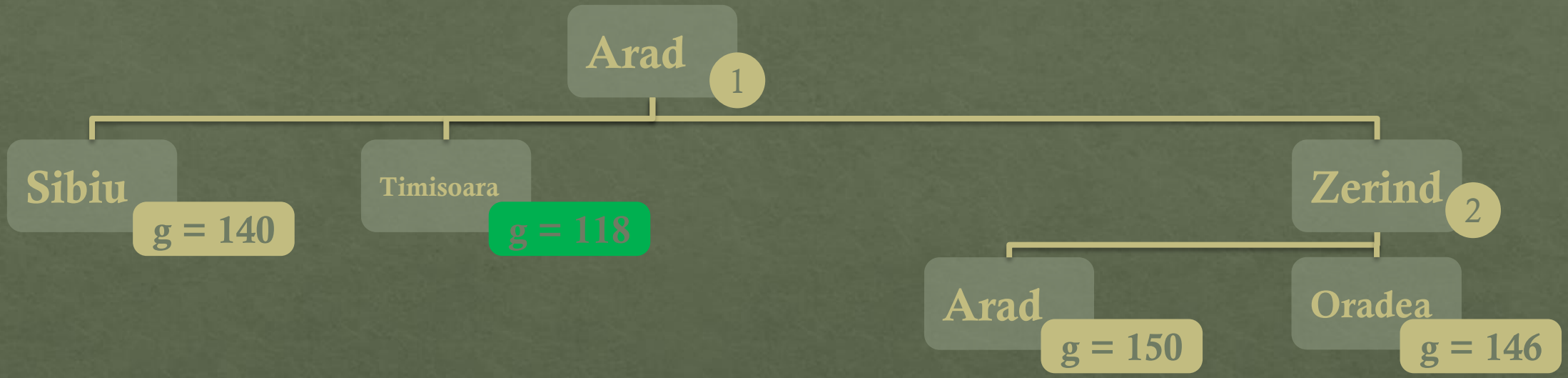
- **پیچیدگی زمانی:** $O(b^{1+(\frac{C}{\varepsilon})})$ **پیچیدگی فضائی:** $O(b^{1+(\frac{C}{\varepsilon})})$

- در مقادیر بالا، C هزینه کم‌هزینه‌ترین راه حل و ε حداقل هزینه‌ی هر عملیات می‌باشد.

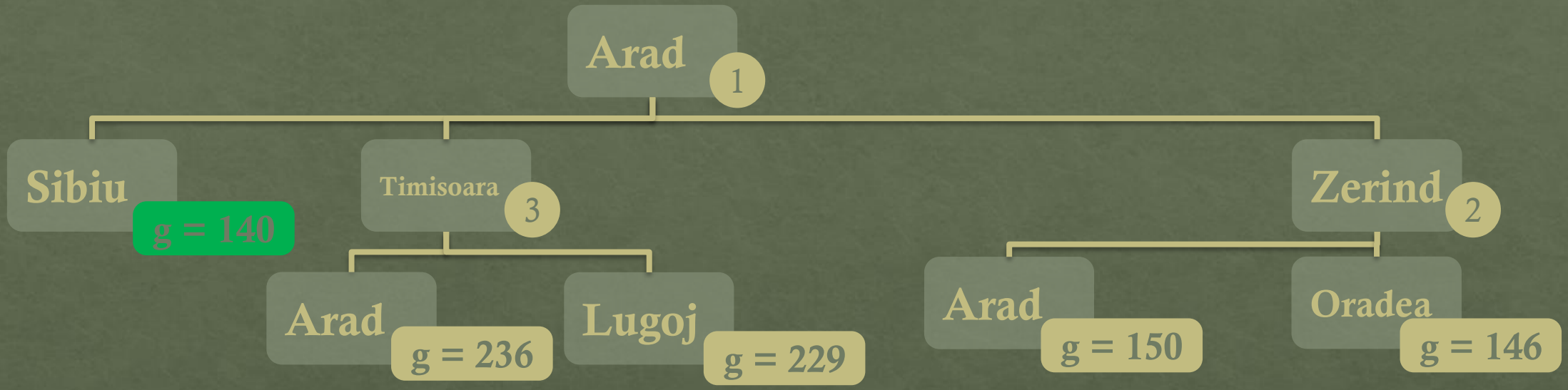
مثال: الگوریتم جستجو هزینه یکنواخت در مسئله سفر در رومانی

مثال: الگوریتم جستجو هزینه یکنواخت در مسئله سفر در رومانی

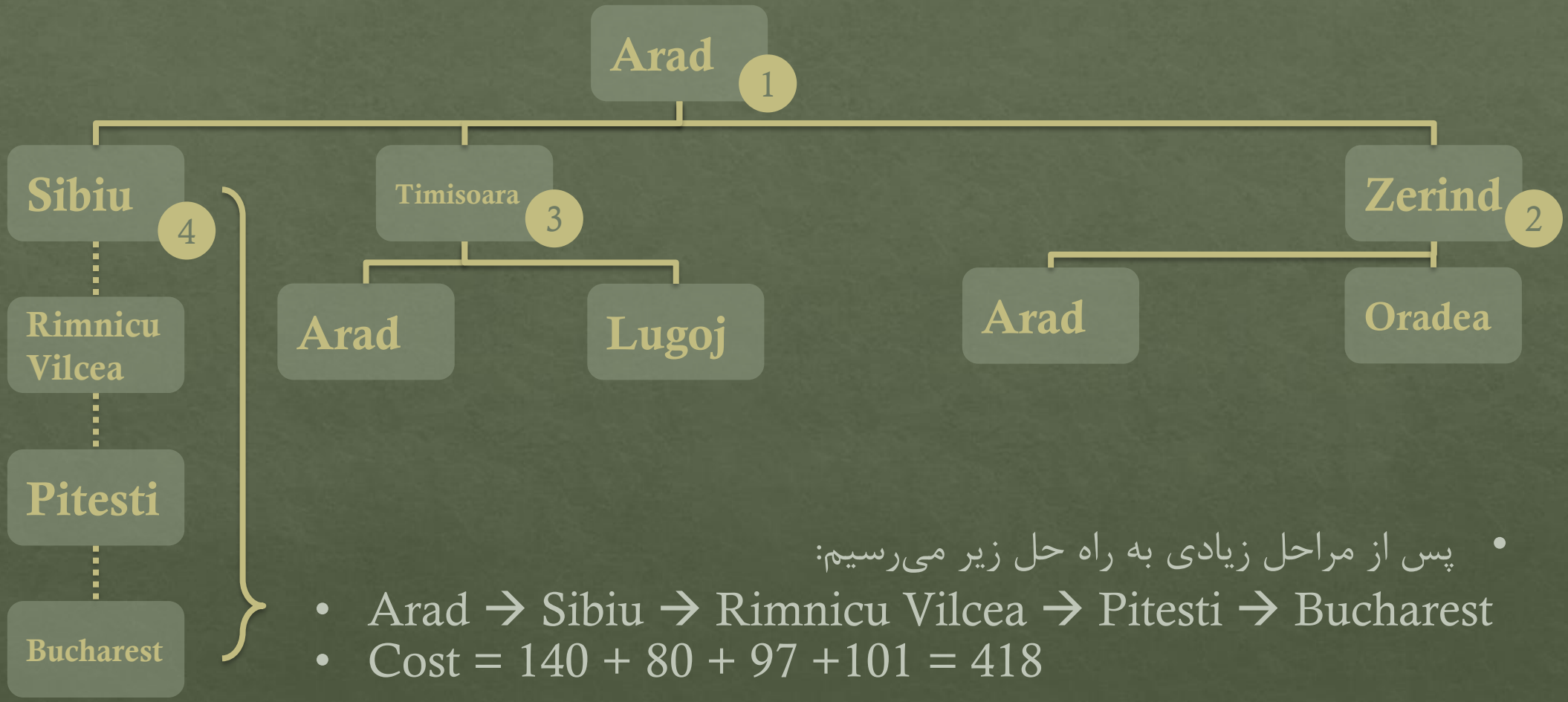
مثال: الگوریتم جستجو هزینه یکنواخت در مسئله سفر در رومانی



مثال: الگوریتم جستجو هزینه یکنواخت در مسئله سفر در رومانی



مثال: الگوریتم جستجو هزینه یکنواخت در مسئله سفر در رومانی



- پس از مراحل زیادی به راه حل زیر می‌رسیم:
- Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest
- Cost = 140 + 80 + 97 +101 = 418

جستجو عمق محدود (Depth Limited Search)

- این الگوریتم از منطق جستجوی عمقی استفاده می‌کند. اما در این الگوریتم برای جلوگیری از گیر افتادن در حلقه بی‌نهایت، مقدار حداکثر عمق جستجو (L) مشخص می‌شود. این الگوریتم پس از رسیدن به عمق L، دیگر به جستجوی عمق‌های بیشتر ادامه نمی‌دهد.

ارزیابی الگوریتم DLS:

- **کامل بودن:** کامل نیست زیرا در صورتی که هدف در عمقی بزرگتر از حداکثر عمق باشد (d>L)، به راه حل نخواهیم رسید.
- **بهینه بودن:** به دلایل مشابه به الگوریتم جستجوی عمقی بهینه نیست.
- **پیچیدگی زمانی:** $O(b^L)$ **پیچیدگی فضائی:** $O(bL)$

جستجو عمق تکراری (Iterative Depth Search)

- این الگوریتم از منطق جستجوی عمق محدود استفاده می‌کند. اما در این الگوریتم برای این‌که همیشه به جواب برسیم، مقدار L به تدریج از صفر تا بی‌نهایت افزایش پیدا می‌کند. بدین معنا که اگر تا عمق L=x به هدف نرسیدیم، مقدار L به x+1 تغییر کرده و دوباره جستجو از اول شروع می‌شود.

- در الگوریتم IDS، به ازای هر L، درخت جستجو دوباره از ابتدا ساخته شده و جستجو می‌شود.

- تفاوت این الگوریتم با الگوریتم جستجوی سطحی این است که در جستجوی سطحی، گره‌های در عمق d+1 نیز تولید می‌شوند، اما در این الگوریتم این‌طور نیست.

ارزیابی الگوریتم IDS

- **کامل بودن:** اگر $b=\infty$ نباشد، این الگوریتم کامل است.
- **بهینه بودن:** بهینه است. (از آن‌جایی که کم‌عمق‌ترین هدف را پیدا می‌کند، در صورتی که هزینه همه عملیات یکسان نباشد، بهینه نیست.)
- **پیچیدگی زمانی:** $O(b^d)$
- **پیچیدگی فضائی:** $O(bd)$

جستجو دوطرفه (Two-way Search)

- در این روش ۲ جستوجو به صورت هم‌زمان انجام می‌شوند. یکی از وضعیت اولیه به سمت هدف به طور رو به جلو (Forward) جستجو می‌کند و دیگری از وضعیت هدف به سمت وضعیت اولیه و رو به عقب (Backward) حرکت می‌کند. هرگاه این ۲ جستوجو یک گره مشترک تولید کنند مسیری از وضعیت اولیه به وضعیت هدف پیدا شده است.

ارزیابی روش Two-way:

- **کامل بودن:** به الگوریتم‌های استفاده شده بستگی دارد.
- **بهینه بودن:** اگر هر دو جستجو سطحی باشند و هزینه همه عملیات یکسان باشد، بهینه است.
- **پیچیدگی زمانی:** $O(b^{d/2})$
- **پیچیدگی فضائی:** $O(b^{d/2})$

مقایسه الگوریتم‌ها

	کامل بودن	بهینگی	پیچیدگی زمانی	پیچیدگی مکانی
اول سطح	بله (به شرط محدود بودن b)	بله (به شرط یکسانی هزینه همه گام‌ها)	$O(b^{d+1})$	$O(b^{d+1})$
هزینه یکنواخت	بله (به شرط محدود بودن b و هزینه هر گام $\geq \varepsilon$)	بله	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$
اول عمق	خیر	خیر	$O(b^m)$	$O(bm)$
عمق محدود	خیر	خیر	$O(b^l)$	$O(bl)$
عمیق شونده تکراری	بله (به شرط محدود بودن b)	بله (به شرط یکسانی هزینه همه گام‌ها)	$O(b^d)$	$O(bd)$
دو طرفه	بله (به شرط محدود بودن b و استفاده از جستجوی اول سطح در هر دو جهت)	بله (به شرط یکسانی هزینه همه گام‌ها و استفاده از جستجوی اول سطح در هر دو جهت)	$O(b^{d/2})$	$O(b^{d/2})$

اجتناب از حالت‌های تکراری

- در الگوریتم‌های جستوجو درختی یک وضعیت ممکن است چند بار در درخت بررسی شود. این بررسی‌های تکراری باعث افزایش زمان جستوجو و در برخی موارد باعث نرسیدن به جواب می‌شوند.

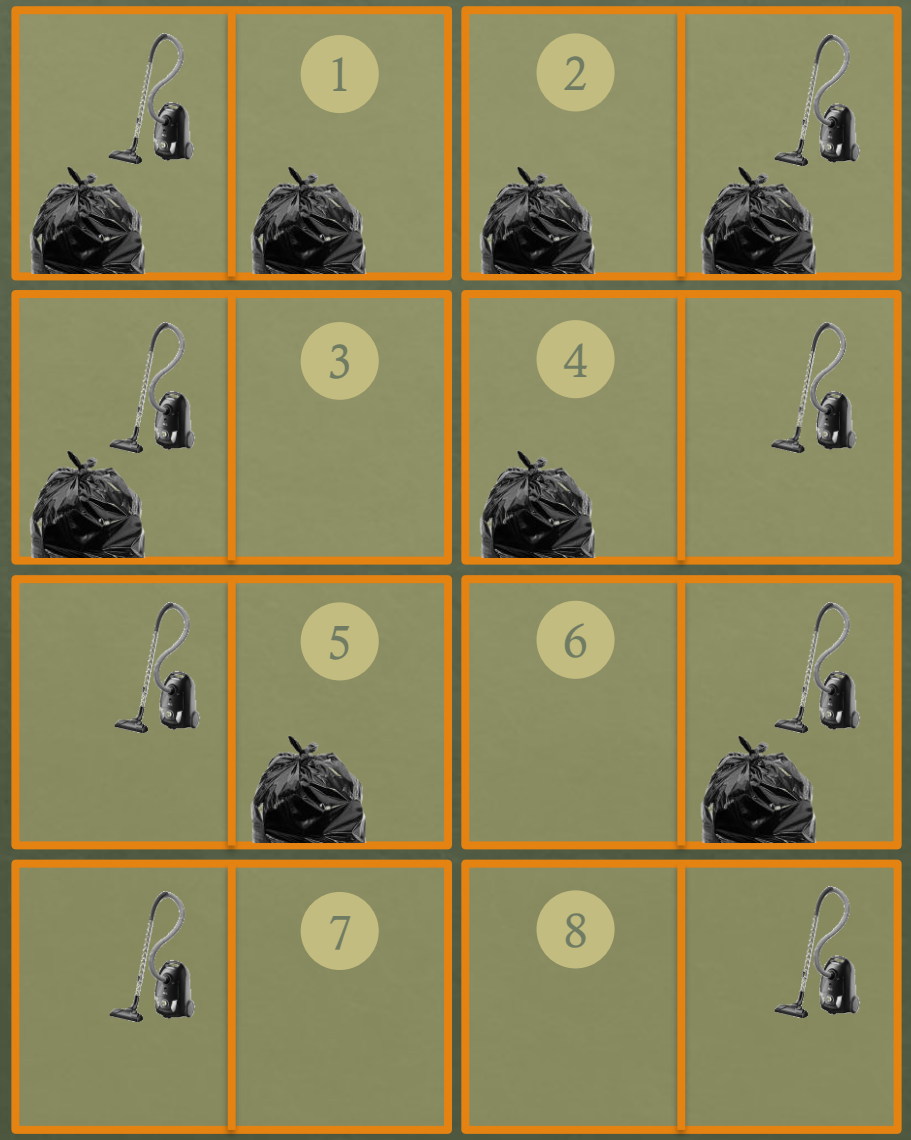
- اگر فضای حالت به صورت یک درخت جهت دار باشد (یعنی عملیات معکوس پذیر نباشند.)، هیچ‌گاه در درخت جستوجو وضعیت تکراری به وجود نمی‌آید. همچنین برای این مسائل درخت جستوجو متناهی خواهد بود.

جستجو با اطلاعات ناقص

- زمانی که محیط کاملا مشاهده‌پذیر نیست، عامل به جای یک حالت با مجموعه‌ای از حالت‌ها سر و کار دارد. بـه چنین مجموعه‌ای از حاالات، حالت باور گفته می‌شود.

- در محیط کاملا مشاهده‌پذیر حالت باور، تک عضوی می‌باشد. برای جستجو در محیط مشاهده‌پذیر جزئی، حالت باور هدف، حالتی است که همه اعضای آن حالت هدف باشند.

مثال: جستجو با اطلاعات ناقص در دنیای جاروبرقی بدون حسگر



- عامل جاروبرقی اثرات تمامی اقداماتش را می‌داند امل هیچ‌گونه حسگری برای بررسی محیط ندارد.

- حالت آغازین عضوی از مجموعه حالت‌های روبرو است. اقدام راست موجب قرار گرفتن عامل در یکی از حالتهای ۲، ۴، ۶ یا ۸ شده و رشته اقدام [راست، مکش] همواره به یکی از حالات ۴ یا ۸ منتهی می‌شود. در نهایت رشته اقدام [راست، مکش]، [چپ، مکش] تضمین می‌کند که حالت هدف ۷ مستقل از حالت شروع به دست بیاید.

مثال: مجموعه حالات باور در دنیای جاروبرقی بدون حسگر