به نام پروردگار دانایی

هوش مصنوعی

درس چهارم: استراتژی‌های جستجو (کلاسیک)

سید کاوه احمدی

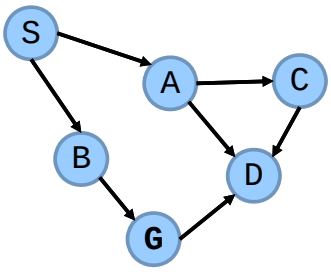
مرور

- مفهوم فضای حالت
- مدل‌سازی گراف فضای حالت
  - فرموله‌سازی مسئله
- فضای حالت یک مدل ریاضی از بیان مسئله
- هدف پیدا کردن روش‌های عمومی برای حل مسائل

درخت جستجو در جستجوهای کلاسیک

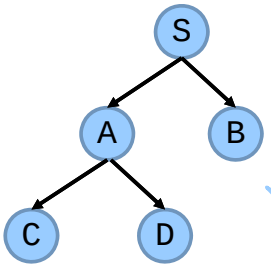
- استراتژی‌های جستجو برای انجام عمل جستجو نیاز به تولید درخت جستجو دارند. در حالی که صرفا نیاز به توصیف گراف فضای حالت داشتیم.
  - استراتژی‌های جستجو مجبورند از وضعیت اولیه شروع کرده و فضای جستجوی پیمایش شده (جستجو شده) از مسئله را تولید (ایجاد) کنند.
  - شکل تولیدی به شکل درخت خواهد بود (درخت جستجو)

درخت جستجو



گراف فضای حالت یا
یک توصیف از آن

برای جستجو باید درخت جستجو را ایجاد (تولید) کنیم



از حالت آغازین شروع می‌کنیم

گره S را **بسط** می‌دهیم. گره‌های A و B **تولید** می‌شوند.

اکنون کدام گره باید **بسط** پیدا کند؟ A یا B؟
فرض می‌کنیم A.

- گره‌های C و D **تولید** می‌شوند.
- اکنون کدام گره باید بسط پیدا کند؟ B، C یا D؟
- **استراتژی جستجو این مورد را مشخص می‌کند.**

درخت جستجو

- توسعه دادن درخت (Expansion)
  - در درخت جستجو، گره‌های برگ را می‌توان بسط داد. بسط هر گره منجر به تولید گره‌های جدید می‌شود (Generation).
  - بسط یک گره منجر به تولید گره‌های مابعد آن می‌شود (در درخت گره‌های فرزند یا Child Nodes).
  - گره‌های مابعد با انجام اعمال مشخص شده و بر اساس مدل انتقال قابل دسترس هستند.
    - در درخت به وسیله یک لبه مرزی قابل اتصال به گره پدر یا Parent Node باشد.

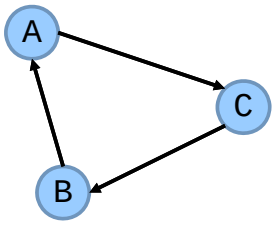
درخت جستجو

- در حین توسعه درخت جستجو با ۳ نوع گره مواجه خواهیم بود.
  - گره‌های بسط داده شده
  - گره‌های بسط داده نشده (تولید نشده و تولید شده‌ای که بسط داده نشده‌اند)
  - گره‌هایی که در نوبت بعدی می‌توانند بسط داده شوند (گره‌های برگ)
    - به این گره‌ها گره‌های حاشیه‌ای (Fringe) یا مرزی (Frontier) یا Open List گفته می‌شود.
    - این گره‌ها باید در یک ساختار داده نگهداری شوند.
    - تفاوت استراتژی‌های جستجو در ساختار داده‌ای مورد استفاده و گره انتخابی از لیست گره‌های در نوبت بسط است.

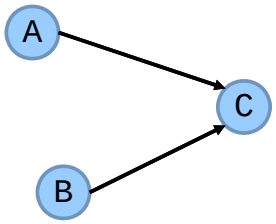
وضعیت‌های تکراری (Repeated States)

- به دلایل زیر در پروسه جستجو ممکن است چندین بار به یک وضعیت مشخص برسیم.

1. تکرار مسیر در حلقه (Loopy Paths):



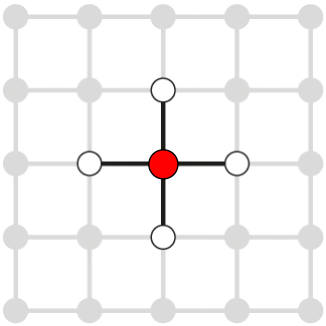
2. مسیرهای افزونه (Redundant Paths): بیش از یک مسیر برای رسیدن به یک وضعیت مشخص داریم.



وضعیت‌های تکراری (Repeated States)

- برای اینکه بتوانیم از جستجوی گره‌هایی که قبلا جستجو شده‌اند (بسط پیدا کرده‌اند) جلوگیری کنیم، وضعیت‌های بسط داده شده را در یک مجموعه (Explored Set یا Close List) ذخیره می‌کنیم.
- گره‌هایی که داخل Close List قرار گرفته‌اند را مجددا درون مجموعه نقاط حاشیه‌ای قرار نمی‌دهیم تا مجددا پیمایش نشوند.

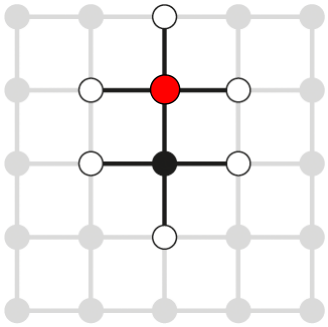
وضعیت‌های تکراری (Repeated States)



- نقاط خاکستری: گراف فضای حالت
- نقاط سیاه: وضعیت‌های پیمایش شده (بسط داده شده)
- نقاط سفید: وضعیت‌های مرزی.

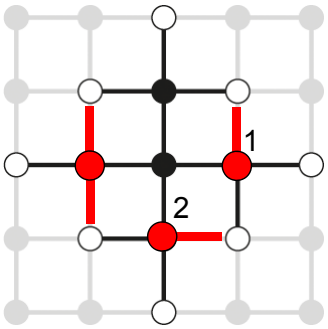
- حالت اولیه (گره قرمز) بسط داده شده و مابعدهای آن (گره‌های سفید) تولید شده‌اند اما هنوز بسط داده نشده‌اند (این گره‌ها گره مرزی هستند).

وضعیت‌های تکراری (Repeated States)



- یکی از گره‌های مرزی (گره قرمز) بسط پیدا کرده است. گره‌های مرزی جدید به لیست گره‌های مرزی اضافه می‌شود (گره‌های سفید).
- عملا درخت جستجو در حال ایجاد است.

وضعیت‌های تکراری (Repeated States)



- بسط گره‌های مرزی ادامه پیدا کرده (گره‌های قرمز). برخی نقاط مرزی که در لیست نقاط مرزی قرار دارند باز تولید نمی‌شوند (یال‌های قرمز در درخت جستجو نمی‌آیند).
  - برای این امر نیاز به ذخیره‌ی گره‌های بسط داده شده خواهیم داشت (Close List).
- **سوال:** بین گره‌های ۱ و ۲ مشخص شده کدام زودتر بسط داده شده‌اند؟

جستجوی درخت در برابر جستجوی گراف

- گراف فضای حالت یک مسئله ممکن است ماهیتا یک درخت باشد
  - مسئله ۸ وزیر (با فرموله سازی افزایشی)
  - در این حالت نیازی به ذخیره Close List وجود ندارد.
- در سایر موارد گراف فضای حالت مسائل واقعا یک گراف است.
  - مسئله معمای ۸
  - در این حالت باید برای اجتناب از حالات تکراری از Close List کمک بگیریم.

جستجوی درخت در برابر جستجوی گراف

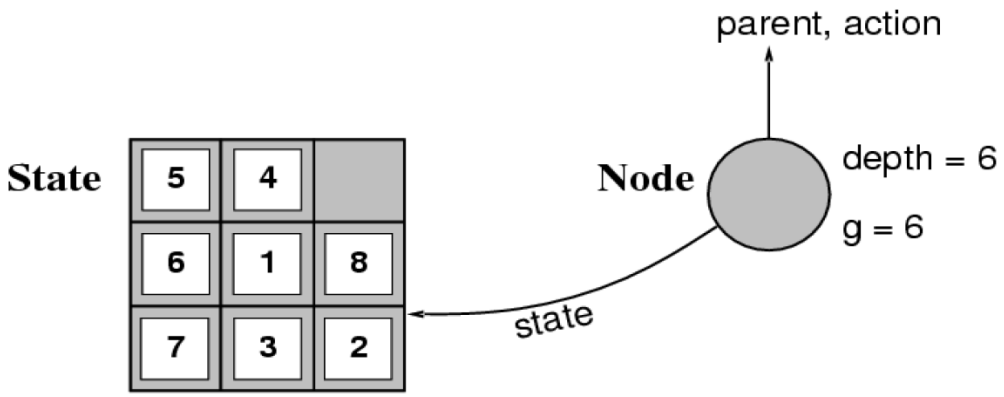
function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

پیاده‌سازی: حالت در برابر گره

- یک حالت (state): یک بازنمایی از یک ترکیب فیزیکی بیرونی
- یک گره (node): یک ساختار داده‌ای که جزئی از یک درخت جستجو را تشکیل دهد:
  - در پیاده‌سازی‌ها، گره‌ها در ساختار داده‌ای مربوط به Open List و Close List ذخیره می‌شوند، اما ساختار درختی مربوطه قابل پیمایش است.



معیارهای ارزیابی استراتژی‌های مختلف (اندازه‌گیری کارایی استراتژی)

- هر استراتژی جستجو از جنبه‌های زیر مورد ارزیابی قرار می‌گیرد.
  - کامل بودن (Completeness): آیا جواب را پیدا می‌کند اگر یکی وجود داشته باشد؟ چه گراف فضای حالت، متناهی حالت یا غیر متناهی حالت داشته باشد.
  - بهینگی (Optimality): آیا کم هزینه‌ترین راه‌حل را پیدا می‌کند؟ اگر چند هدف داریم بهترین را برمی‌گزیند؟
  - پیچیدگی زمانی (Time Complexity): چقدر طول می‌کشد راه حل را پیدا کند؟ چند گره تولید یا گسترش داده می‌شود.
  - پیچیدگی فضای مصرفی (Space Complexity): برای جستجو چقدر حافظه نیاز دارد؟ حداکثر تعداد گره‌هایی که باید در حافظه نگهداری شود چندتا است؟

معیارهای ارزیابی استراتژی‌های مختلف (اندازه‌گیری کارایی استراتژی)

- پیچیدگی زمان و فضای مصرفی بر مبنای موارد زیر اندازه‌گیری می‌شود:
  - b فاکتور انشعاب (Branching Factor): میانگین (یا حداکثر) مابعدهایی که یک گره در مسئله می‌تواند داشته باشد (درجه خروجی گره‌های گراف فضای حالت).
  - d عمق پاسخ (Depth of Solution): کم عمق‌ترین گره هدف
  - m عمق حداکثر گراف (Maximum Depth of Graph): طول بلندترین مسیر در گراف فضای حالت
    - در مسائل مورد بررسی، d باید متناهی باشد اما m می‌تواند متناهی یا نامتناهی باشد.
  - C^* هزینه پاسخ بهینه.

معیارهای ارزیابی استراتژی‌های مختلف (اندازه‌گیری کارایی استراتژی)

- استراتژی قابل قبول (Admissible)
  - به استراتژی‌ای که کامل و بهینه باشد گفته می‌شود.

مثال: رومانی

- برای تعطیلات به شهر آراد در رومانی آمده بودیم.
- فردا بلیطی از مقصد بخارست داریم.
  - هدف رفتن از آراد به بخارست
  - راه‌های مختلفی برای رفتن از آراد به بخارست وجود دارد.

مثال: رومانی

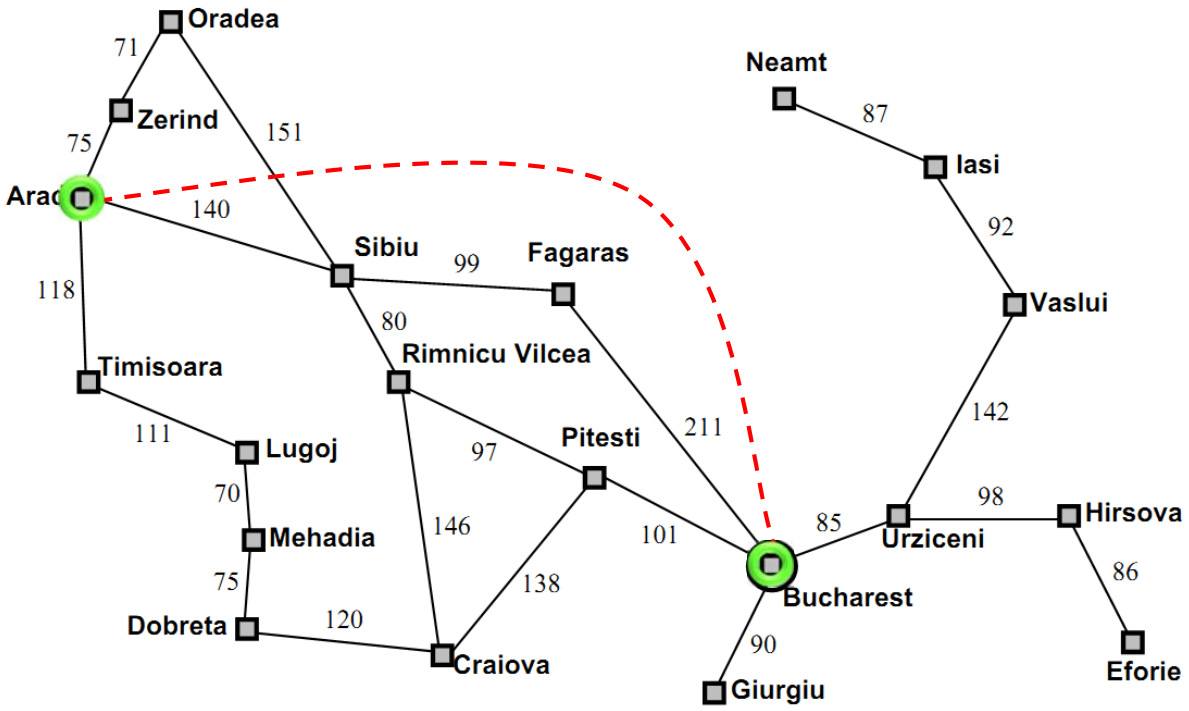
- صورت مسئله: رفتن از آراد به بخارست
- فرموله کردن هدف: رسیدن به بخارست
- فرموله کردن مسئله:
  - وضعیت‌ها (states): شهرها
  - عمل‌ها (actions): رانندگی بین شهرها
- جستجو: دنباله‌ای از شهرها (مثلا Arad, Sibiu, Fagaras, Bucharest)
  - این جستجو بر اساس کم‌هزینه‌ترین مسیر

مثال: رومانی

- حالت اولیه:
  - `in(Arad)`
- عمل‌ها
  - `in(Arad) {go(Sibui), go(Timisoara), go(Zerind)}`
- مدل انتقال: حالت بعد از انجام عمل چه می‌شود؟
  - `Result(in(Arad), go(Zerind) = Zerind`
- ویرایش دوم کتاب:
  - تابع مابعد (`successor function`)

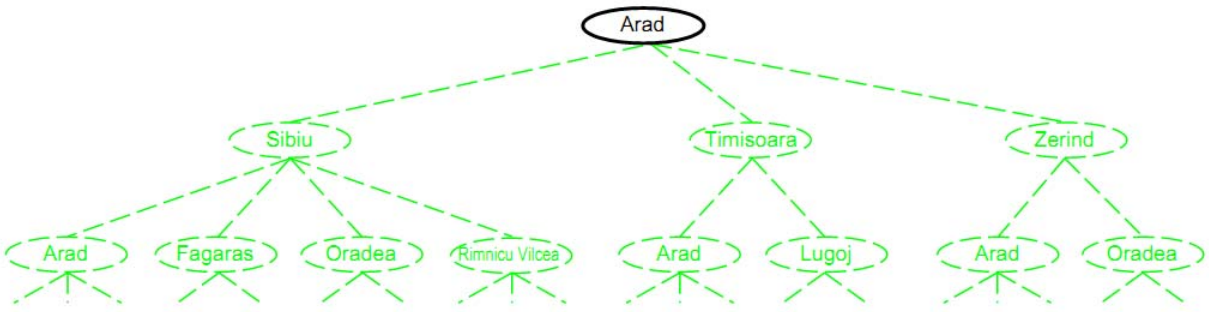
- `Succesor-func[In(Arad)] = {<Go(Sibiu), In(Sibiu)>, <Go(Timiso), In(Timiso)>, <Go(Zerind), In(Zerind)>}`

مثال: رومانی – گراف فضای حالت



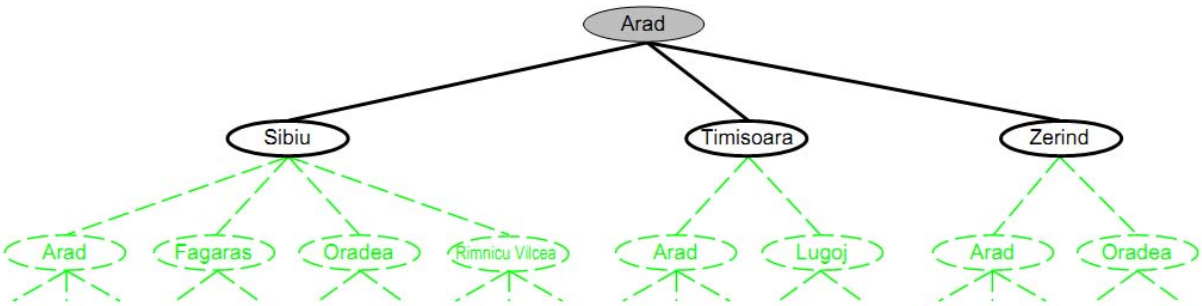
درخت جستجو برای مسئله رومانی

- شروع جستجو از حالت اولیه
  - آیا هدف است؟



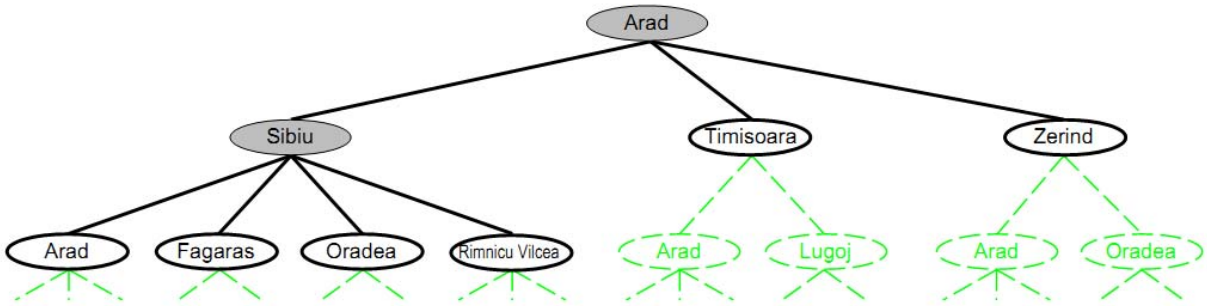
درخت جستجو برای مسئله رومانی

- توسعه حالت جاری (مدل انتقال (تابع مابعد) به حالت جاری)
  - انتخاب یکی از حالت‌های بسط داده شده (استراتژی جستجو مشخص می‌کند کدامیک)
  - بقیه حالت‌ها برای بعدا نگه داشته خواهند شد (گره‌های حاشیه‌ای).
  - توسعه تا زمانی که یا پاسخی یافت شود و یا حالتی برای توسعه باقی نماند ادامه می‌یابد.



درخت جستجو برای مسئله رومانی

- فرض کنید Sibiu انتخاب می‌شود
  - آیا هدف است؟ خیر. توسعه درخت ادامه پیدا می‌کند.
- انتخاب شهرهای تازه توسعه داده شده یا بازگشت به عقب و انتخاب دو شهر باقی‌مانده (همه آنها گره‌های مرزی)؟ (استراتژی جستجو مشخص می‌کند کدام!)



استراتژی جستجو

- استراتژی جستجو، حالتی که برای توسعه انتخاب می‌شود را مشخص می‌کند.
  - بین فضای حالت و درخت جستجو تفاوت است
    - یک ارتباط بین دو شهر وجود دارد اما بینهایت مسیر

انواع استراتژی‌های جستجو با تولید درخت جستجو

- ناآگاهانه (Blind/Uninformed Search): به جز تعریف مسأله هیچگونه اطلاعاتی در مورد مسأله داده نمی‌شود.
  - این الگوریتم‌ها فقط می‌توانند مابعدهایی را تولید و هدف را از غیر هدف تشخیص دهند.
- آگاهانه (Heuristic/Informed Search): دارای ایده‌هایی درمورد اینکه کجا به دنبال هدف بگردیم.
  - راهبردهایی که تشخیص می‌دهد یک حالت غیر هدف نسبت به گره غیر هدف دیگر، امید بخش‌تر است، جستجوی آگاهانه، جستجوی اکتشافی نامیده می شود. بنابراین نواحی امیدبخش (Inspiring Regions) زودتر بررسی می‌شوند.

مهندسی آی تی ۸۹

- کدام عبارت در مورد یک روش جستجوی مناسب غلط است؟

1. پیچیدگی مکانی جستجو باید قابل قبول باشد
2. ضریب انشعاب جستجو باید یک باشد
3. پیچیدگی زمانی جستجوی باید قابل قبول باشد
4. جستجو باید کامل و بهینه باشد

مهندسی آی تی ۸۹

- گزینه ۲ صحیح است.