بسمه تعالی

هوش مصنوعی

حل مسئله – ۲

نیمسال اوّل ۱۴۰۳-۱۴۰۲

دکتر مازیار پالهنگ

آزمایشگاه هوش مصنوعی

دانشکدهٔ مهندسی برق و کامپیوتر

دانشگاه صنعتی اصفهان

یادآوری

- مثال جهانگرد
- تدوین هدف
- تدوین مسئله
- شرایط محیط برای یک عامل مسئله حل کن:
  - مشاهده پذیر، قطعی، شناخته شده
- تدوین مسئله
  - حالت اولیه، مجموعهٔ اعمال ممکن، مدل انتقال، هدف، هزینهٔ مسیر
- چند مثال:
  - دنیای جارو، جورچین

مسئلهٔ knuth

- با شروع از ۴ و با استفاده از دنباله ای از اعمال جذر، کف، و فاکتوریل می توان به هر عدد صحیح مثبتی رسید.

$$\left\lfloor \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{(4!)!}}}}} \right\rfloor = 5.$$

- حالات: اعداد حقیقی مثبت
- حالت اولیه: ۴
- اعمال: جذر، کف، فاکتوریل (فقط برای اعداد صحیح)
- مدل انتقال: طبق تعریف اعمال
- هدف: عدد صحیح مطلوب
- هزینه: هر عمل ۱

مسیریابی

- رفتن از شهری به شهر دیگر با خودرو
- یافتن مسیر در شبکه های کامپیوتری
- یافتن پروازهای مورد نظر برای سفر از یک شهر به شهر دیگر
- **حالات**: مکانها (فرودگاهها) و زمان فعلی
- **حالت اولیه**: بودن در فرودگاه مبدأ کاربر
- **اعمال**: رفتن از یک فرودگاه به فرودگاه دیگر
- **مدل انتقال**: پس از پرواز، فرودگاه مقصد فرودگاه فعلی و زمان رسیدن زمان فعلی می شود.
- **هدف**: فرودگاه مقصد مورد نظر کاربر
- **هزینه**: هزینهٔ پولی، مدت زمان انتظار، زمان پرواز

گردشگری

- همانند مسیریابی
- بازدید از تعدادی شهر حداقل یکبار
- حالت: در کدام شهر و چه شهرهائی بازدید شده
- هدف: در شهر مقصد و بازدید همهٔ شهرها

فروشندهٔ دوره گرد

- همان مسئلهٔ گردشگری فقط هر شهر فقط باید یکبار دیده شود و یافتن کوتاهترین مسیر، و تمام شهرهای یک نقشه مورد نظر
- حرکت یک دریل برای سوراخ کردن یک مدارچاپی

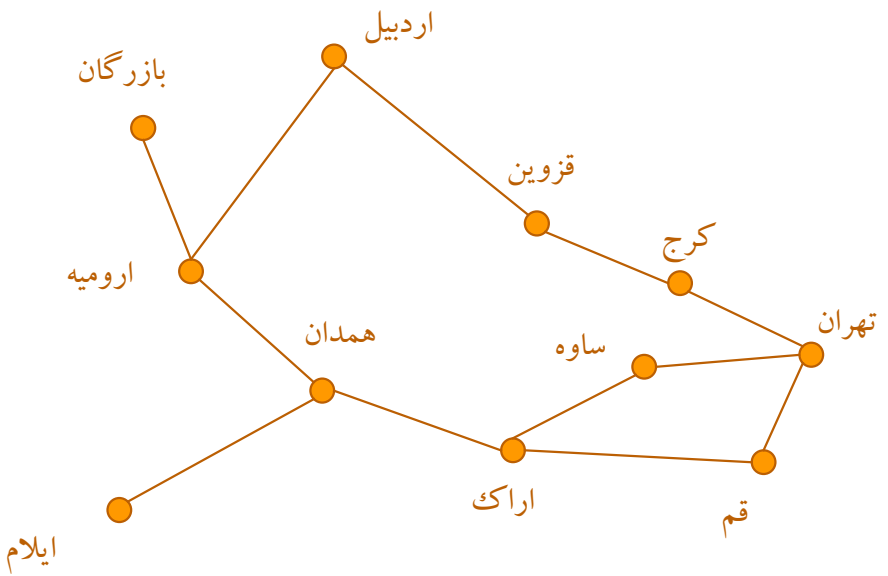
مسائل دیگر

- چینش مدارهای الکترونیک
- چینش مدارهای VLSI

جستجو برای حل

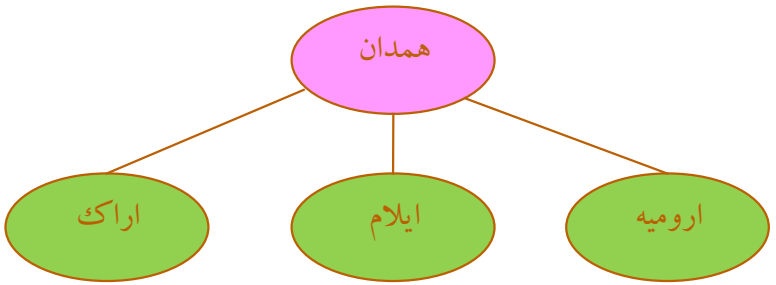
- پس از تدوین مسئله باید آن را حل نمود.
- یک حل دنباله ای است از اعمال
- الگوریتمهای جستجو، با در نظر گرفتن دنباله های اعمال متفاوت کار می کنند.
- دنباله های عمل ممکن با شروع از حالت اولیه یک درخت جستجو می سازند.
- حالت اولیه در ریشه
- شاخه ها متناظر با اعمال ممکن

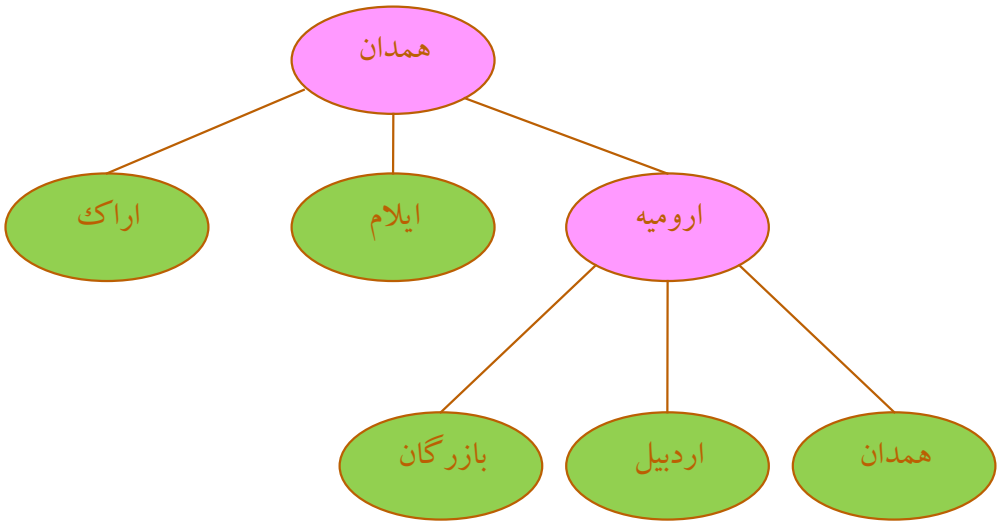
جستجو برای حل



- جستجو در فضای حالت


همدان

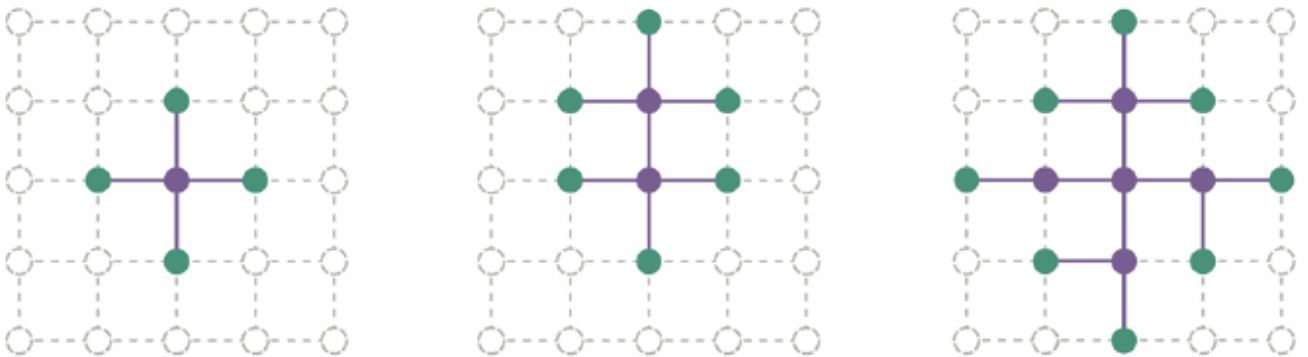

همدان
اراک
ایلام
ارومیه


همدان
اراک
ایلام
ارومیه
بازرگان
اردبیل
همدان

- مجموعهٔ همهٔ رئوس در دسترس برای بسط داده شدن در هر زمان مجموعهٔ پیشگام (frontier) نامیده می شود.
- گاهی به آن لیست باز (open list) نیز گفته می شود.
- حالتی که برای آن رأسی ایجاد شده، گفته می شود که به آن رسیده ایم (reached) (ممکن است هنوز بسط داده نشده باشد).

- مجموعهٔ پیشگام، فضای حالت را به دو ناحیه تقسیم می کند: داخلی و خارجی

Figure 3.6



The separation property of graph search, illustrated on a rectangular-grid problem. The frontier (green) separates the interior (lavender) from the exterior (faint dashed). The frontier is the set of nodes (and corresponding states) that have been reached but not yet expanded; the interior is the set of nodes (and corresponding states) that have been expanded; and the exterior is the set of states that have not been reached. In (a), just the root has been expanded. In (b), the top frontier node is expanded. In (c), the remaining successors of the root are expanded in clockwise order.

جستجوی بهترین نخست

- یک روش عمومی برای انتخاب رأسی که باید بسط داده شود، استفاده از جستجوی بهترین نخست است.
- انتخاب رأس n که کمترین مقدار یک تابع ارزیابی مثل $f(n)$ را داراست.

جستجوی بهترین نخست

Figure 3.7

```
function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node ← NODE(STATE=problem.INITIAL)
  frontier ← a priority queue ordered by f, with node as an element
  reached ← a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s] ← child
        add child to frontier
  return failure
```

```
function EXPAND(problem, node) yields nodes
  s ← node.STATE
  for each action in problem.ACTIONS(s) do
    s' ← problem.RESULT(s, action)
    cost ← node.PATH-COST + problem.ACTION-COST(s, action, s')
    yield NODE(STATE=s', PARENT=node, ACTION=action, PATH-COST=cost)
```

الگوریتمهای جستجوی درختی

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

جستجو برای حل

- اگر به درخت جستجوی مثال دقت شود، رأس همدان دوبار دیده می شود.
- به چنین رئوسی، رئوس تکراری گفته می شود.
- رئوس تکراری باعث ایجاد مسیر حلقوی می شوند.
- وجود چنین رئوسی باعث می شود که درخت بی نهایت بزرگ شود.
- ولی فضای حالت محدود است.
- مسیرهای حلقوی حالت خاص مسیرهای زائد هستند.
- مسیر زائد هنگامی وجود دارد که بیش از یک مسیر بین دو حالت وجود دارد.

جستجو برای حل

- گاهی می توان مسئله را به گونه ای تدوین کرد که دارای تکرار نباشد.
- بطور مثال در مسئلهٔ ۸ وزیر، اگر هر وزیر را بتوان در هر ستونی از صفحهٔ شطرنج گذاشت، در این حالت هر وضعیت قرار گیری n وزیر در صفحه دارای $n!$ مسیر مختلف خواهد بود.
- ولی اگر وزیر را فقط بتوان در چپترین ستون خالی قرار داد فقط یک مسیر وجود دارد.
- در برخی از مسائل که اعمال برگشت پذیر هستند، همانند مثال همدان، حالتهای تکراری اجتناب ناپذیر هستند.

- مثالی است که "**جامعه ای که تاریخ خود را فراموش کند، محکوم به تکرار آن است.**"
- برای اجتناب از مسیرهای زائد، لازم است مکانهائی که بوده ایم را به خاطر بسپاریم.
- استفاده از ساختمان داده ای به نام **مجموعهٔ اکتشاف شده** (explored set)
- یا لیست بسته
- الگوریتمی که از این مجموعه استفاده می کند جستجوی گرافی نامیده می شود.
- در الگوریتم بهترین نخست از ساختمان دادهٔ reached استفاده شده


پایان

- تذکر مهم:
- پاورپوینت وسیله ای برای کمک به تدریس و یک ارائهٔ شفاهی می باشد و به هیچ وجه یک جزوهٔ درسی نیست و
- لازم است حتماً مرجع درس مطالعه شود.