بسمه تعالی

هوش مصنوعی

حل مسئله – ۴

نیمسال اوّل ۰۳-۱۴۰۲

دکتر مازیار پالهنگ

آزمایشگاه هوش مصنوعی

دانشکدهٔ مهندسی برق و کامپیوتر

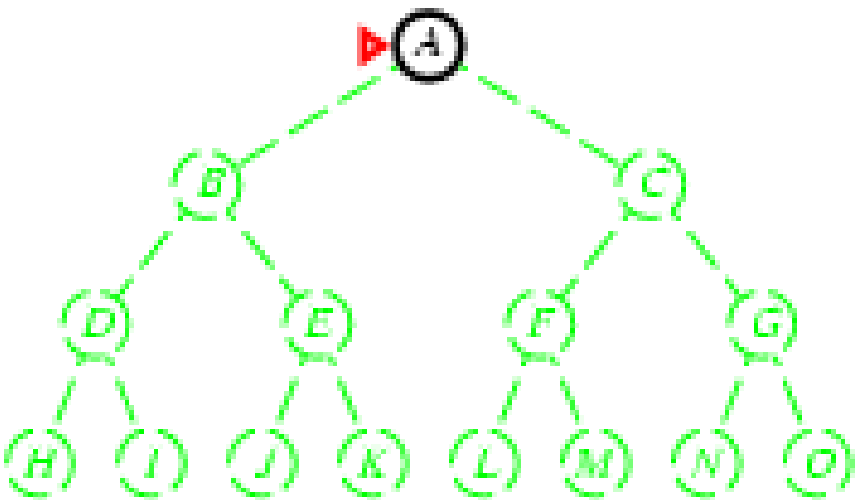
دانشگاه صنعتی اصفهان

یادآوری

- مثال جهانگرد
- تدوین هدف و مسئله
- شرایط محیط برای یک عامل مسئله حل کن:
  - ایستا، مشاهده پذیر، قطعی، گسسته
- تدوین مسئله
  - حالت اولیه، مجموعهٔ اعمال ممکن، مدل انتقال، هدف، هزینهٔ مسیر
- چند مثال:
  - دنیای جارو، جورچین ۸، مسیریابی، گردشگری، فروشندهٔ دوره گرد
- جستجو برای حل
  - ایجاد درخت، مجموعهٔ پیشگام، جستجوی درختی، جستجوی گرافی
- ساختمان داده برای جستجو
- معیارهای ارزیابی استراتژیهای جستجو
  - کامل بودن، بهینه بودن، پیچیدگی فضا، پیچیدگی زمان
- جستجوی عرض نخست
- جستجوی هزینه یکنواخت

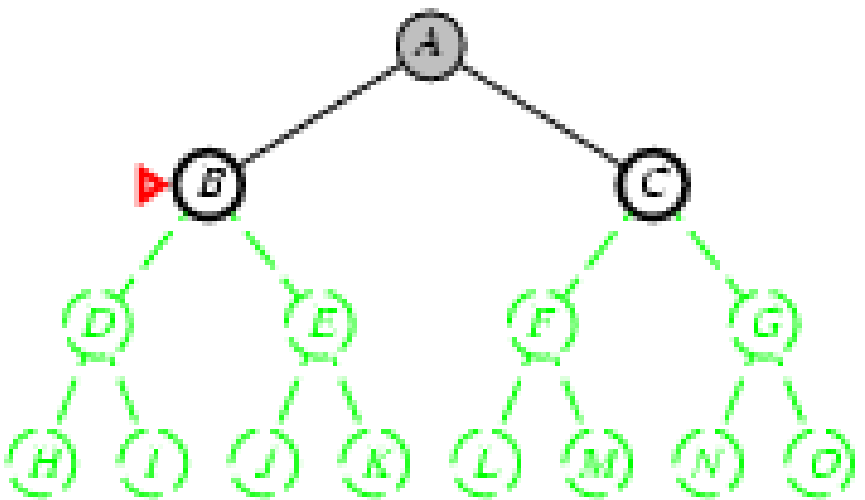
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



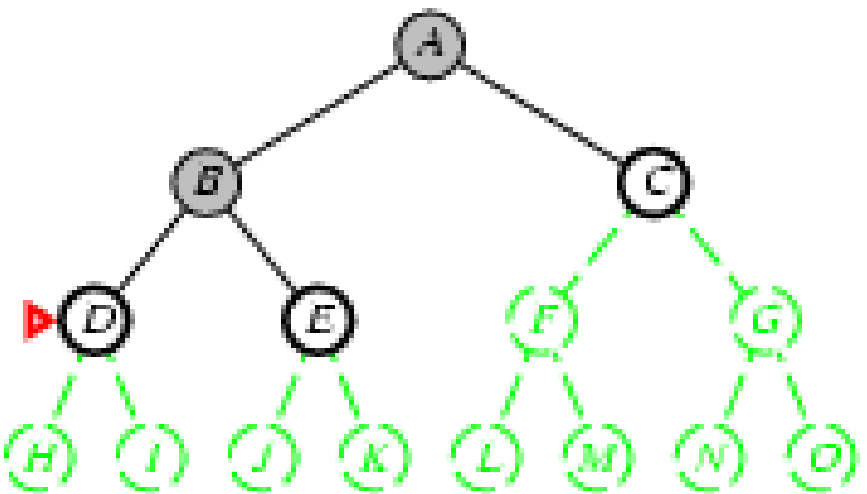
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



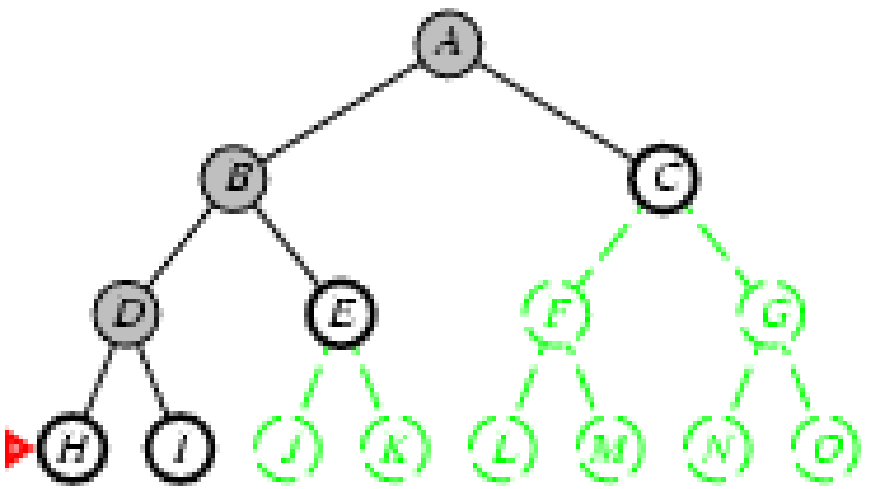
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



جستجوی عمق نخست

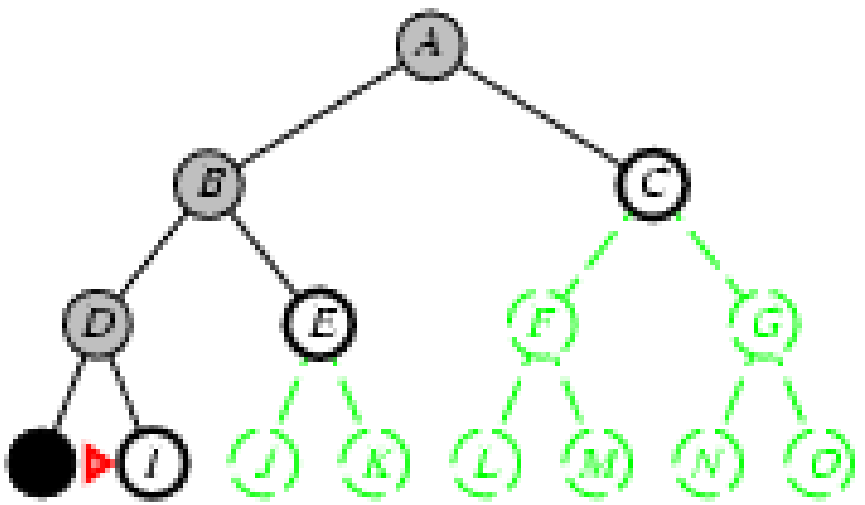
- بسط عمیق ترین گرهٔ رسیده شده



6

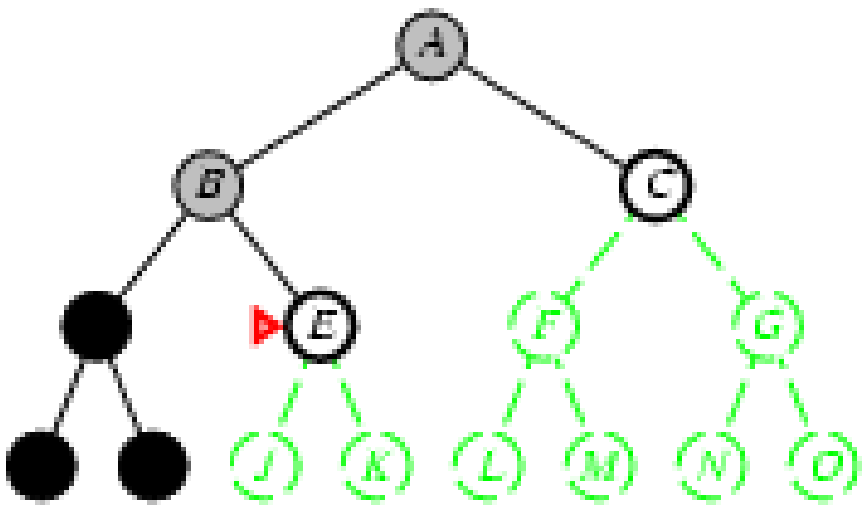
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



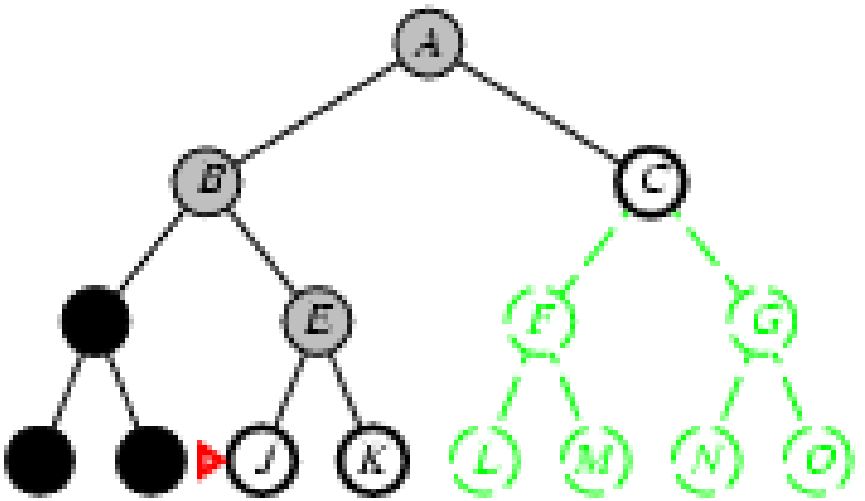
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



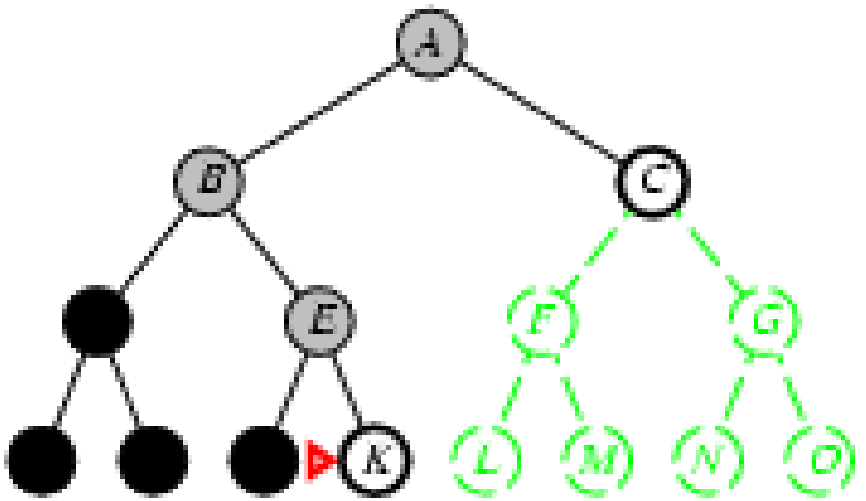
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



جستجوی عمق نخست

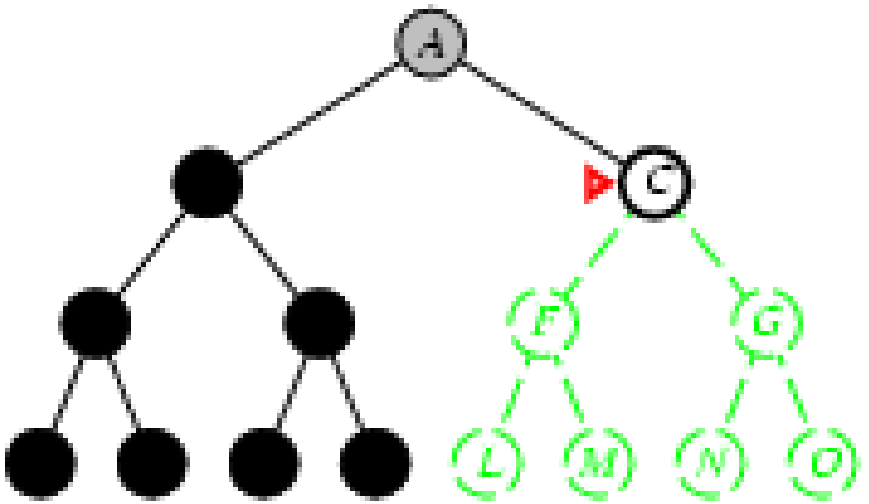
- بسط عمیق ترین گرهٔ رسیده شده



10

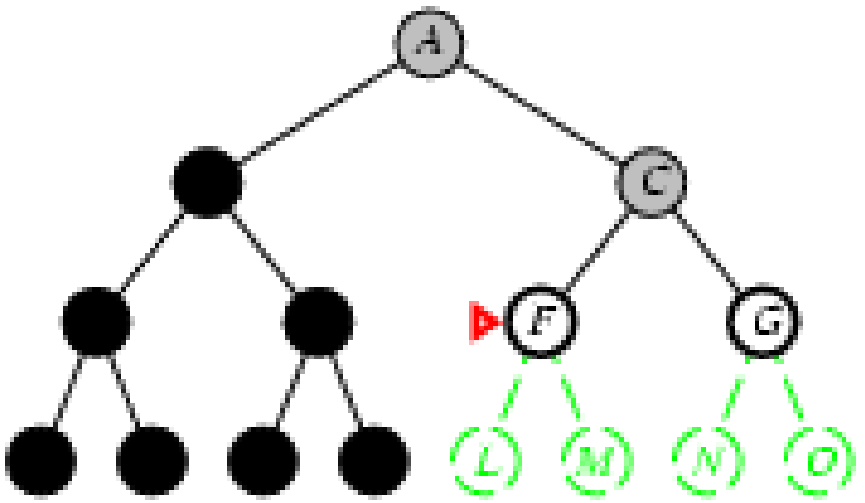
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



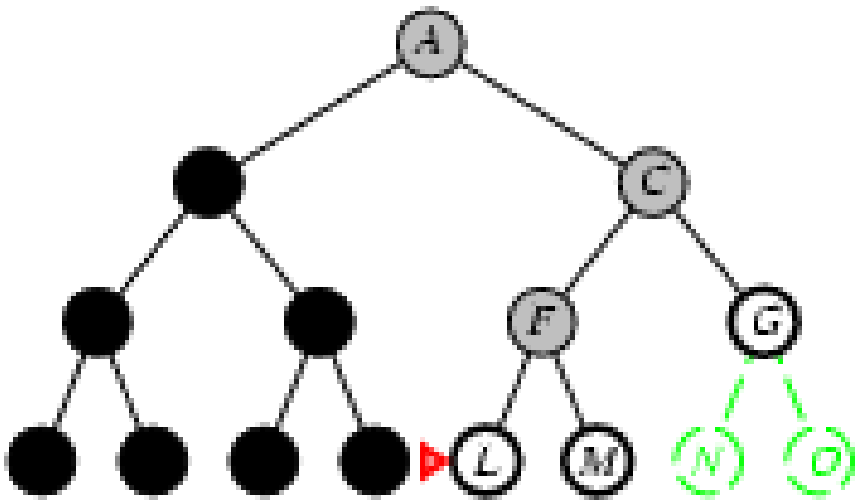
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



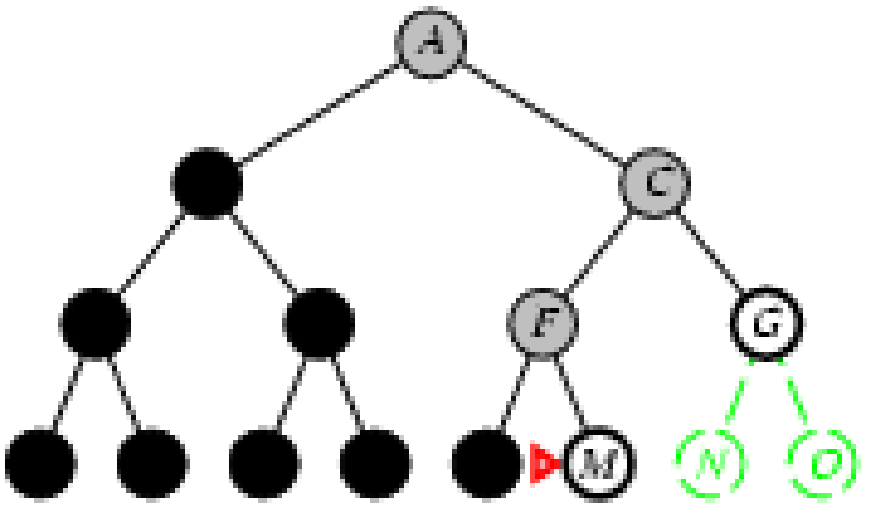
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



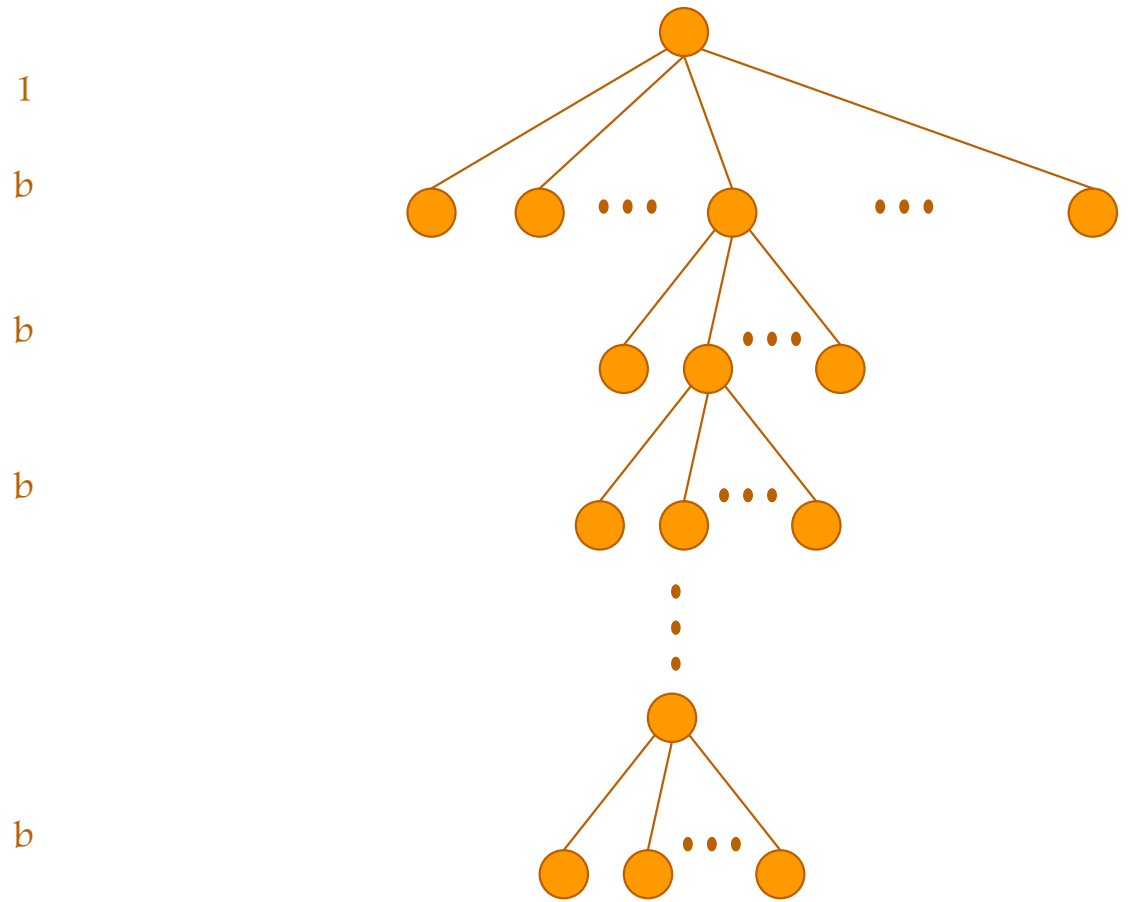
جستجوی عمق نخست

- بسط عمیق ترین گرهٔ رسیده شده



جستجوی عمق نخست

- می تواند از جستجوی بهترین نخست استفاده کرد با $f(n)$ برابر منفی عمق n
- frontier یک صف LIFO می تواند باشد.
- کامل:
  - نه ممکن است در حلقهٔ بی نهایت قرار گیرد
- بهینه:
  - نه


1
b
b
b
b

جستجوی عمق نخست

- زمان: $O(b^m)$
- فضا: $O(bm)$

جستجوی عقبگرد

- تنوعی از جستجوی عق نخست که در هر زمان یک تالی (بجای همهٔ تالیها) تولید می شوند.
- هر رأس جزئی بسط داده شده بخاطر دارد که بعداً چه تالی باید تولید شود.
- پیچیدگی حافظه $O(m)$
- اگر مشکل حافظه زیاد باشد می توان از حافظهٔ ایجاد شده برای یک حالت استفاده کرد و آنرا برای ایجاد حالت جدید تغییر داد.
- در این حالت باید راهی برای بازگشت به عقب از هر حالت وجود داشته باشد.

جستجوی عمق محدود شده

- جستجوی عمق نخست با حد عمق 1

```
function DEPTH-LIMITED-SEARCH(problem, ℓ) returns a node or failure or cutoff
  frontier ← a LIFO queue (stack) with NODE(problem.INITIAL) as an element
  result ← failure
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    if DEPTH(node) > ℓ then
      result ← cutoff
    else if not IS-CYCLE(node) do
      for each child in EXPAND(problem, node) do
        add child to frontier
  return result
```

به نظر می رسد علامت بزرگتر یا مساوی صحیح باشد

جستجوی عمق محدود شده

- اگر $l<d$ کامل نیست.
- اگر $l>d$ بهینه نیست.
- حد l را از دانش مسئله ممکن است بتوان بدست آورد.
- بطور مثال در مورد مثال جهانگرد می دانیم که حداکثر از چندشهر باید عبور کرد.

پیاده سازی بازگشتی در ویرایش سوم

```
function DEPTH-LIMITED-SEARCH(problem, limit) returns a solution, or failure/cutoff
  return RECURSIVE-DLS(MAKE-NODE(problem.INITIAL-STATE), problem, limit)

function RECURSIVE-DLS(node, problem, limit) returns a solution, or failure/cutoff
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  else if limit = 0 then return cutoff
  else
      cutoff_occurred? ← false
      for each action in problem.ACTIONS(node.STATE) do
          child ← CHILD-NODE(problem, node, action)
          result ← RECURSIVE-DLS(child, problem, limit − 1)
          if result = cutoff then cutoff_occurred? ← true
          else if result ≠ failure then return result
      if cutoff_occurred? then return cutoff else return failure
```

Figure 3.17 A recursive implementation of depth-limited tree search.

جستجوی عمیق ساز تکراری

Figure 3.12

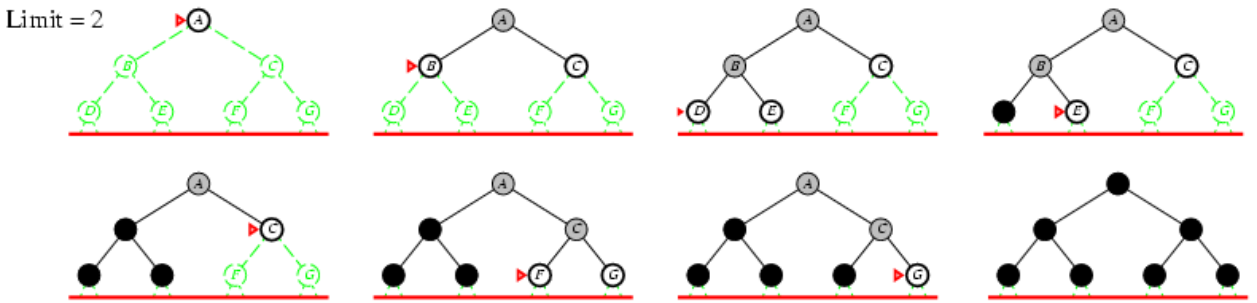
```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution node or failure
  for depth = 0 to ∞ do
    result ← DEPTH-LIMITED-SEARCH(problem, depth)
    if result ≠ cutoff then return result
```

جستجوی عمیق ساز تکراری

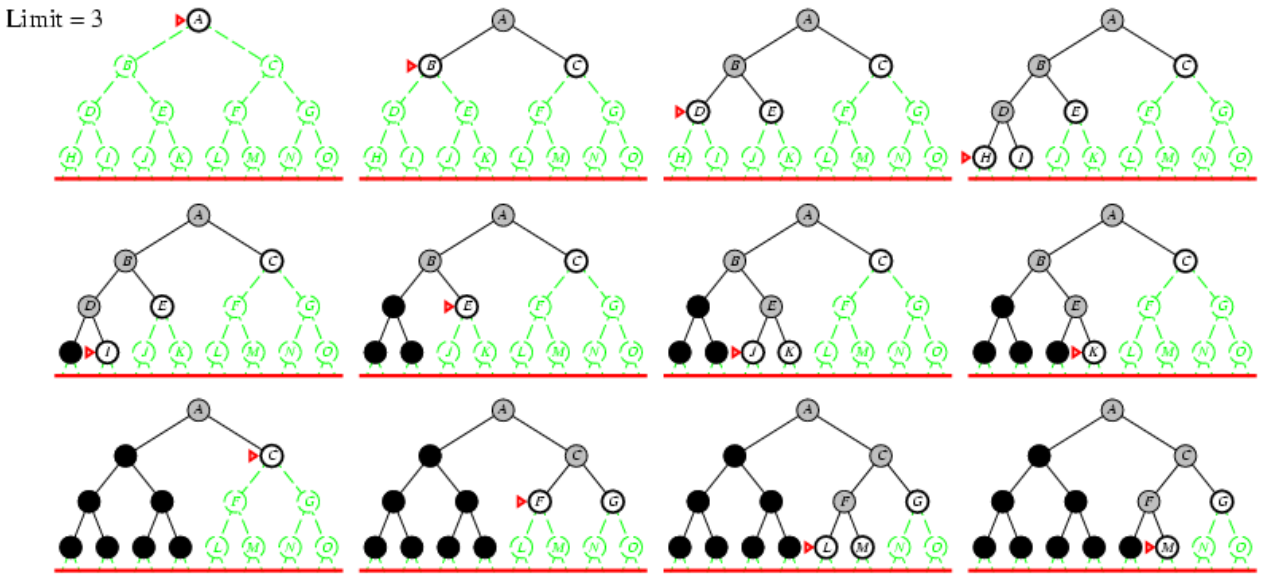
Limit = 0

جستجوی عمیق ساز تکراری

جستجوی عمیق ساز تکراری



جستجوی عمیق ساز تکراری



جستجوی عمیق ساز تکراری

- تعداد رئوس برای عمق محدود شده
- $N_{DLS} = b^0 + b^1 + b^2 + \ldots + b^{d-2} + b^{d-1} + b^d$
- برای عمیق ساز تکراری
- $N_{IDS} = (d+1)b^0 + d\,b^1 + (d-1)b^2 + \ldots + 3b^{d-2} + 2b^{d-1} + 1b^d$
- برای $b=10$ و $d=5$
  - $N_{DLS} = 1 + 10 + 100 + 1{,}000 + 10{,}000 + 100{,}000 = 111{,}111$
  - $N_{IDS} = 6 + 50 + 400 + 3{,}000 + 20{,}000 + 100{,}000 = 123{,}456$
  - $N_{BFS1} = 1 + 10 + 100 + 1{,}000 + 10{,}000 + 100{,}000 = 111{,}111$
  - $N_{BFS2} = 1 + 10 + 100 + 1{,}000 + 10{,}000 + 100{,}000 + 999{,}990 = 1{,}111{,}101$
- سربار $= (123{,}456 - 111{,}111)/111{,}111 = 11\%$

- کامل: بله
- زمان: $(d+1)b^0 + d\,b^1 + (d-1)b^2 + \ldots + b^d = O(b^d)$
- مکان: $O(bd)$
- بهینه: بله اگر هزینهٔ مسیر برابر باشد.
- یک راه دیگر آن است که تا جائی که حافظه اجازه می دهد از جستجوی عرض نخست استفاده کنیم، و
- پس از آن بر روی رئوسی که در پیشگام هستند جستجوی عمیق ساز تکراری استفاده کنیم.
- جستجوی ناآگاهانه مناسب وقتی که عمق پاسخ را نمی دانیم و فضای حالت بزرگ است.

جستجوی دو طرفه

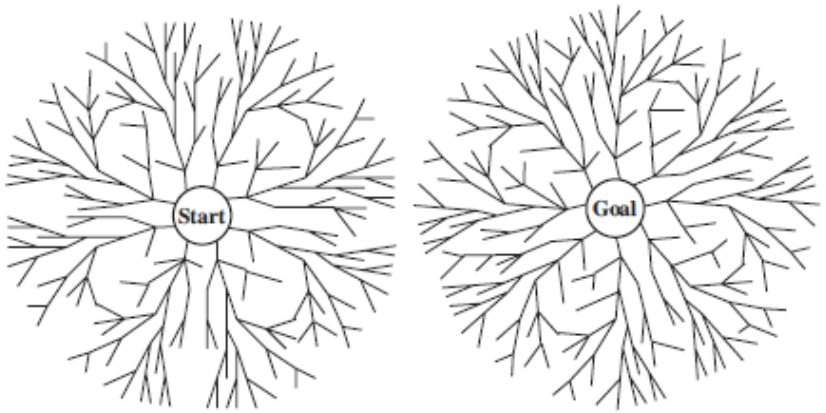


Figure 3.20 A schematic view of a bidirectional search that is about to succeed when a branch from the start node meets a branch from the goal node.

جستجوی دو طرفه

- انگیزه: $b^{d/2}+b^{d/2}$ بهتر از b^d است.
- پیشرفت در هر دو سو تا پیشگامان دو طرف اشتراک پیدا کنند.
- اگر برای مسئله ای $d=6$ و هر دو طرف جستجوی عرض نخست انجام دهد یک رأس در هر زمان
- در بدترین زمان هنگامی به هم می رسند که تمام رئوس عمق ۳ ایجاد شده است.
- ایجاد ۲/۲۲۰ رأس بجای ۱/۱۱۱/۱۱۰ رأس
- در صورتی که هر دو طرف از جستجوی عرض نخست استفاده کنند (و هزینهٔ مراحل برابر باشد) این روش بهینه و کامل است.

جستجوی دو طرفه

- در صورتی امکان پذیر است که عملگرها برگشت پذیر باشند.
- مثال جهانگرد اینگونه است.
- برای دنیای جارو که دو حالت هدف می تواند داشته باشد (هر دو خانه تمیز ولی ربات در راست یا چپ) ساخت یک حالت هدف ساختگی که حالتهای قبلی آن حالات هدف واقعی هستند.
- مثال شطرنج مشکل است (تعداد حالات هدف بسیار)

مقایسه

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[1]	Yes[1,2]	No	No	Yes[1]	Yes[1,4]
Optimal cost?	Yes[3]	Yes	No	No	Yes[3]	Yes[3,4]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$

Evaluation of search algorithms. b is the branching factor; m is the maximum depth of the search tree; d is the depth of the shallowest solution, or is m when there is no solution; ℓ is the depth limit. Superscript caveats are as follows: [1] complete if b is finite, and the state space either has a solution or is finite. [2] complete if all action costs are $\geq \epsilon > 0$; [3] cost-optimal if action costs are all identical; [4] if both directions are breadth-first or uniform-cost.

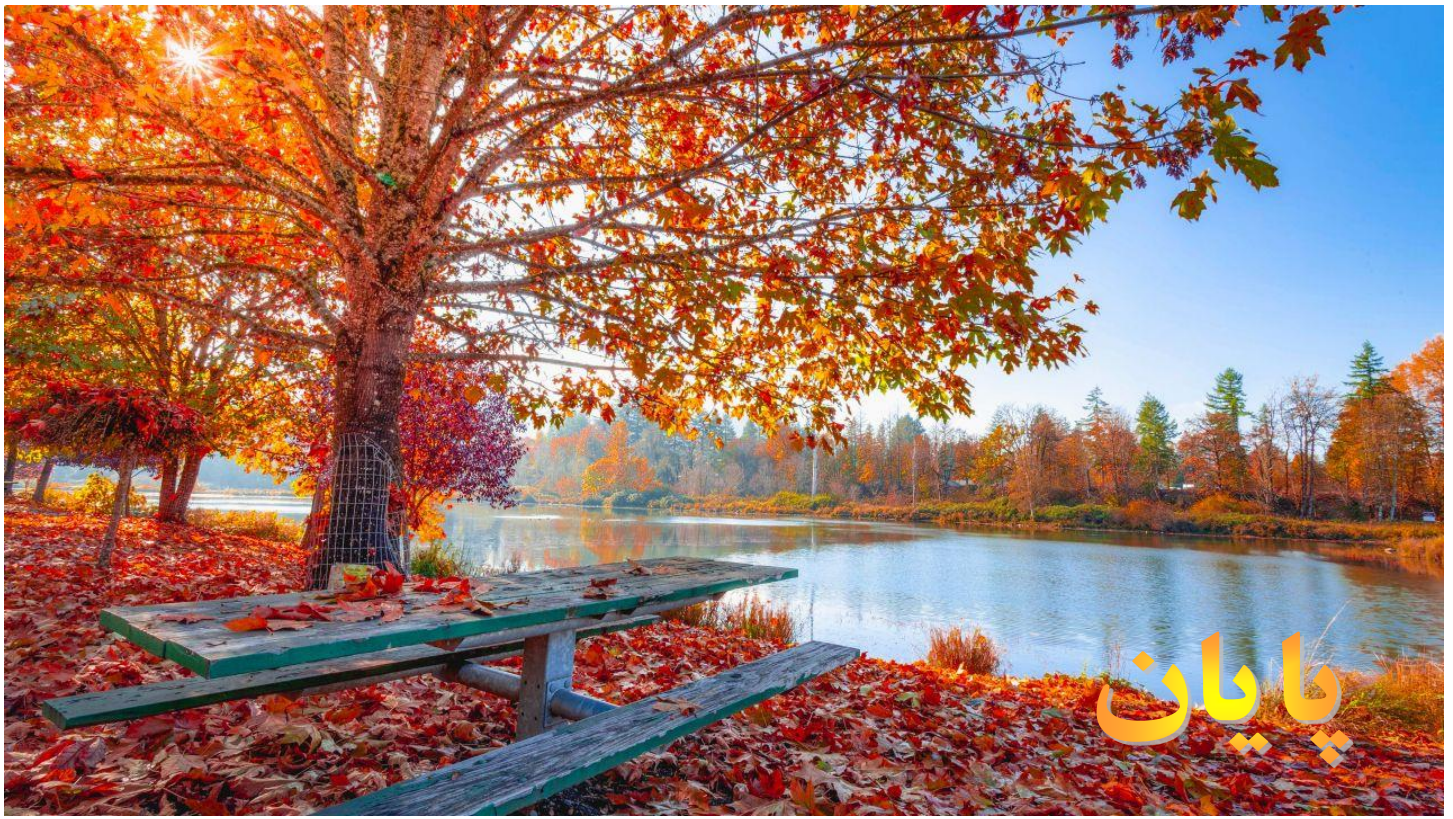
- مقایسه در حالت جستجوی درختی

جستجوی گرافی

- چون رئوس در لیست رسیده شده نگهداری می شوند، عمق نخست و عمیق ساز تکراری دیگر پیچیدگی حافظهٔ خطی ندارند.
- عمق نخست در حالت جستجوی گرافی برای فضای حالت محدود کامل است.
- برخی جستجوها به روش گرافی ممکن است امکان پذیر نباشند (بخاطر حافظه).

خلاصه

- جستجوی عقبگرد
- جستجوی عمق محدودشده
- جستجوی عمیق ساز تکراری
- جستجوی دوطرفه
- مقایسه جستجوهای ناآگاهانه


پایان