$A^\star$-algorithm, Python

CS156

Chris Pollett

Sep 8, 2014

Outline

- More on $A^\star$-search
- Quiz
- Python

Introduction

- Last Wednesday, we were talking about uninformed and informed search strategies.
- Depth-first search, Breadth-first search, and uniform cost search were examples of uninformed strategies because they do not make use of knowledge about the goal state in performing the search.
- We then talked about iterative deepening depth first search as a way to combine the memory efficiency of DFS with the completeness and optimality of BFS.
- Finally, we looked two informed search strategies Greedy Best First Search and the $A^\star$-algorithm.
- Let's begin today by looking at the pseudo-code for the $A^\star$-algorithm.

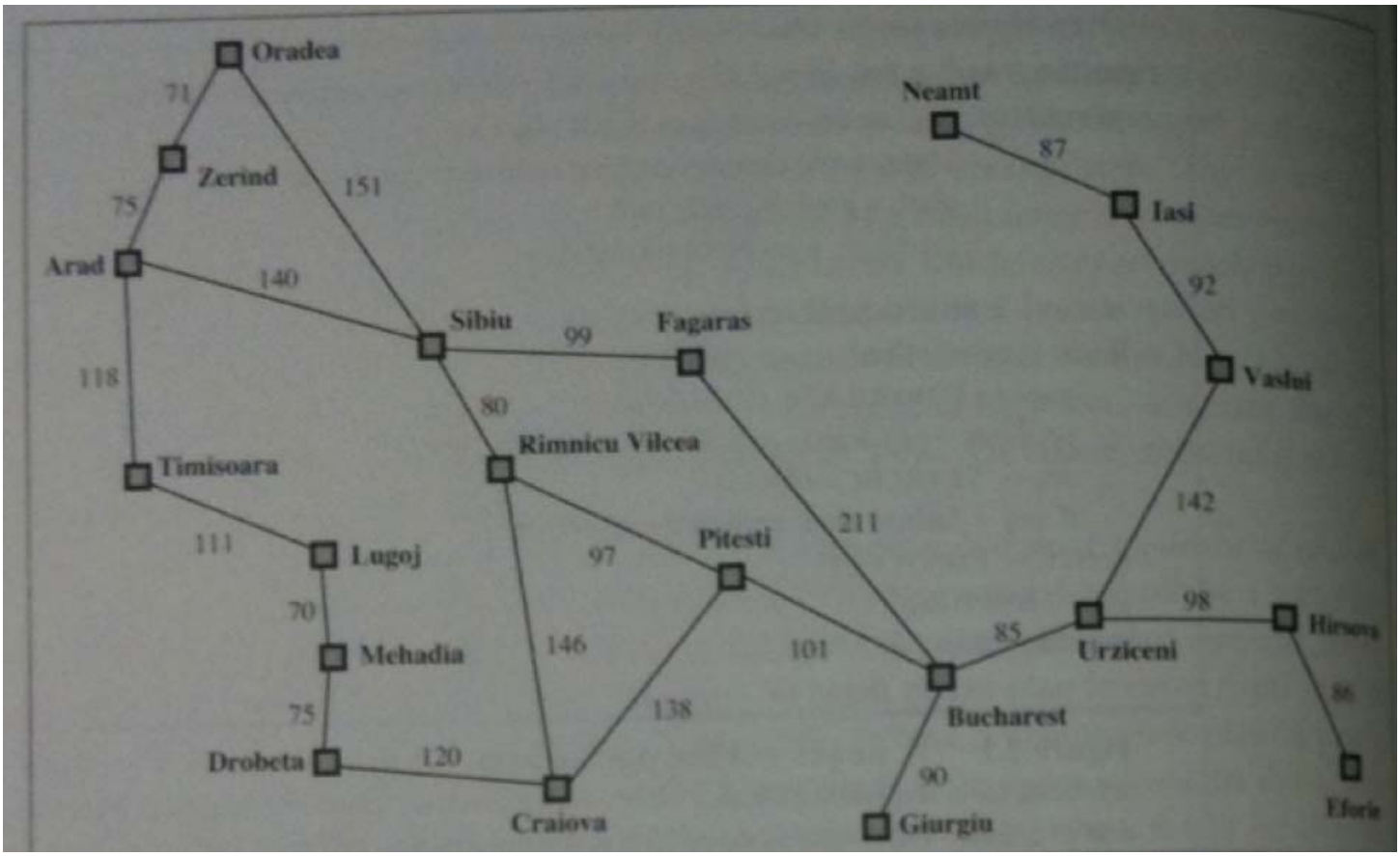
$A^\star$-algorithm

- Recall in informed search we have a fitness function f which says how good a current state is.
- For the Greedy algorithm, $f = h$, where h is a heuristic function estimating the cost to a solution; for $A^\star$, $f = c + h$, where h is as in the Greedy Algorithm and c was the cost to get to the current state from the initial state.
- The pseudo-code for the $A^\star$-algorithm looks like what you would do for uniform cost search:

```
function A-STAR-SEARCH(problem) returns a solution, or failure
    node := a node with STATE = problem.INITIAL-STATE, PATH-COST = 0,
        H-COST = heuristics cost to solution
    frontier := a priority queue ordered by f=c+h, with node as
        the only element
    explored := an empty set
    loop do
        if EMPTY?(frontier) then return failure
        node := POP(frontier) /* lowest f=c+h-value in frontier
            since priority-queue */
        if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
        add node.STATE to explored
        for each action in problem.ACTIONS(node.STATE) do
            child := CHILD-NODE(problem, node, action)
            if child.STATE is not in explored or frontier then
                frontier := INSERT(child, frontier)
            else if child.STATE is in frontier with higher f value
                replace that frontier node with child
```

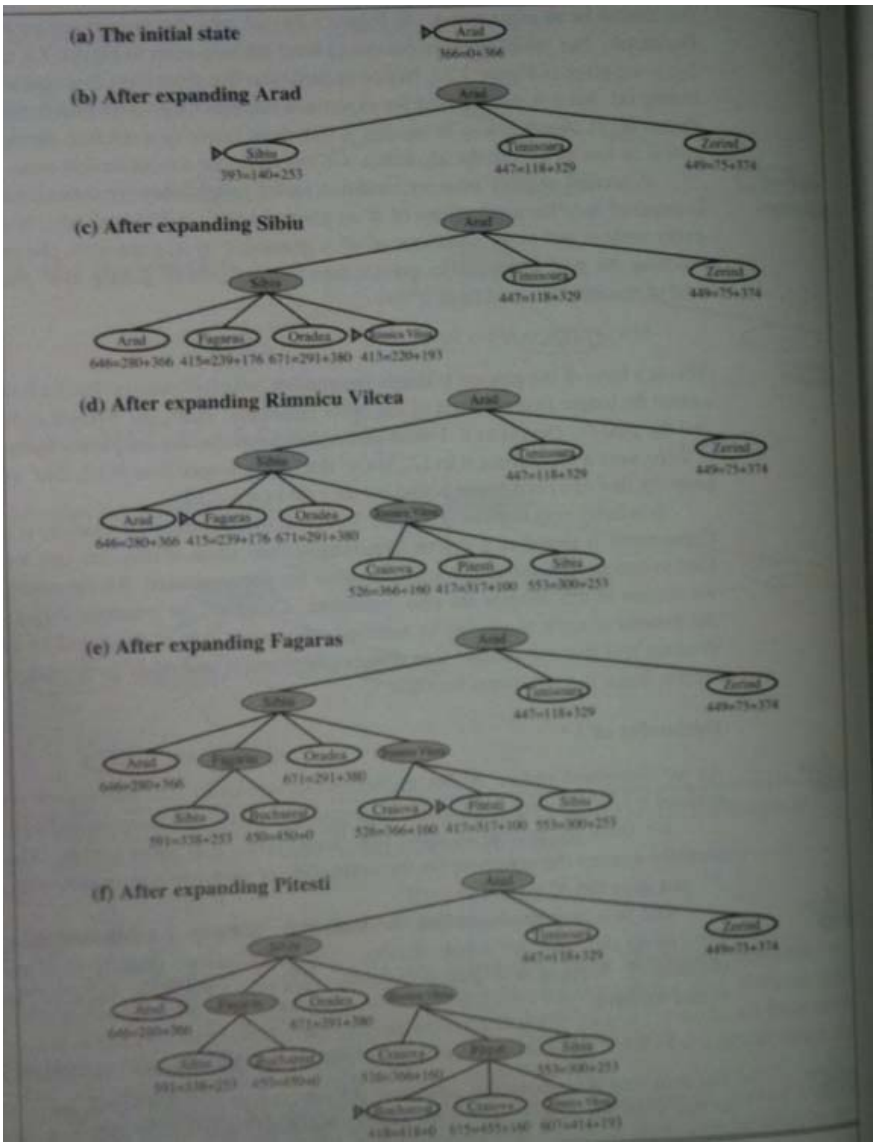
$A^\star$-Example Map

We will use the following road map of Romania to illustrate the $A^\star$-algorithm:



$A^\star$-Example Map

- The trees below illustrate a search for Bucharest using $A^\star$ starting at Arad.
- The darkened nodes are the explored ones, the node with the triangle next to it is the node under consideration, all other nodes are on the frontier.



Quiz

Which of the following is true?

1. A rational agent always has complete information about its environment.
2. Iterative Deepening Depth First search is typically more memory efficient than Breadth-First search.
3. Problem solving agents always use factored and not atomic representations of their environment.

Python

- For this semester, we are going to mainly code our AI projects in Python.
- Python is a programming language created by Guido van Rossum whose origins trace to the late 1980s.
- When looking for an language to code AI projects there are several things to look for:
  1. *Good string and list processing facilities (minimize awkward additional user coding)*
  2. *Scripted/interpreted coding available for faster testing and development*
  3. *Higher-order function support. i.e., Can easily make functions which take other functions as arguments.*
  4. *Syntax is not too weird compared to what everybody else uses.*
  5. *Good set of built-in libraries.*
  6. *Wide-range of free libraries/projects can build off.*
  7. *People outside of AI use it, so other people can appreciate code.*
- A case can be made that Python satisfies each of these requirements.
- The first three points are probably why LISP was originally popular as an AI language.
- Unfortunately, (4) and to a lesser degree (7) are weaknesses of LISP/SCHEME/Racket.
- According to TIOBE Language Popularity Index , Python is the third most popular scripting language after PHP and Basic.
- If one looks at Ohloh, there are actually more open-source projects listed there for Python than PHP.

Getting Started

- Python can be obtained from: http://python.org/download/.
- Python 2.7.x is available by default on Mac's running OSX Mavericks, so we'll stick to that rather than Python 3 (for what we do it won't make much difference).
- Some differences between Version 2 and 3 are unicode support in strings, exception chaining, annotations, etc -- none of which we'll really make use this semester.

Running Python

- Assuming you have set up your path environment, you can launch python in interactive mode by just typing:

  ```
  python
  ```

  at the cmmand prompt. You should see something like:

  ```
  Python 2.7.1 (r271:86832, Jun 16 2011, 16:59:05)
  [GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2335.15.00)] on darwin
  Type "help", "copyright", "credits" or "license" for more information.
  >>>
  ```

- Typing quit() on a line or hitting CTRL-D returns you to the command prompt.
- As an example, we could type:

  ```
  >>>print "hello world"
  ```

  which would print hello world to the terminal.

Python as Calculator; Programs in Files

- You can use the command prompt as a quick calculator of python expressions. For example,

  ```
  >>>2 / 3 + 7.9
  ```

  outputs 7.9; whereas, 2.0/3 + 7.9 outputs 8.566666666666666.
- You can refer to the last result by using an underscore similar to Perl. So after the last expression _ + 1 would give 9.566666666666666.
- Python file are put into files with the .py extension. For example, we might have the file: hello.py with the following in it:

  ```
  #!/usr/bin/env python
  # The above would mean don't have to type python when run under *nix
  # Comments begin with #
  print "Hello World"
  ```

  If we didn't have the first line or on Windows we would need to type "python hello.py" to run the program.

Variables and Arithmetic Expressions

- Variables are bound to values by assignments. We don't have to declare variables before using them.
- So we can do things like:

```
x = 7
x += 1 #note there are no ++ and -- operators
x *= 3
y = 5
z = x - y
print z
```

  which would print 19
- Notice we don't have to terminate statements with ';' although we can. If we want several statements of the same line we use ';'
- Python supports a formatted string operator %:

```
print "%3d %0.2f" % (10, .9799)
```

  prints 10 with a leading space followed by 0.98
- There is also a format() function which outputs a string. So format(0.979, "0.2f") outputs the string 0.98 .

Conditionals

- Python supports if elif else conditionals with the syntax:

```
a = 5
if a > 4:
    print "a is bigger than 4"

b = 99
if  b < 50:
    print "b is too little"
else:
    print "b is big enough"

if  a > 4 and  b < 50:
    print "1"
elif not a == 6:
    print "2"
else:
    print "3"
```

- The bodies of these statements are denoted using indentation.
- To create an empty clause, you can use the pass statement.
- Python has boolean connectives and, or, and not.
- Python uses the Boolean literal True and False.
- Python does not have switch/case

File I/O

- One could read in a file using a program like:

```
f = open("hello.py")
line = f.readline()
while line:
   print line, #comma omits print's newline char
   #print (line, end='') #in python 3
   line = f.readline()
f.close()
```

- open() returns a file object
- We can invoke methods of this object just as in Java (in this case, readline())
- We could have also written:

```
for line in open("hello.py"):
    print line,
```

  to achieve the same affect
- To write a file we can do things like:

```
f = open("somefile.txt", "w")
print >>f, "%02f" %0.7999
f.write("hello")
f.close()
```

- To get input from the terminal or write to the terminal we can use the sys.stdout and sys.stdin objects:

```
import sys
sys.stdout.write("Type something");
name = sys.stdin.readline()
```

Strings

- String literals can be enclosed in either single, double, or triple quotes:

```
a = "Hello"
b = 'Good "bye"'
c = """ Triple quotes one can
go over multiple lines
"""
d = ''' this one
works as well
'''
```

- It is relatively easy to extract characters out of a string:

```
a = "Hello World"
b = a[4] # b is 'o'
c = a[:5] # c is 'Hello'
d = a[6:] # d is 'World'
e = a[3:8] # e is "lo Wo"
```

- Strings are concatenated using + .
- A string can be converted to a integer using int() and a float using float(). For example, int("79")
- The command str and repr can be used to convert other types into strings. For example, str(3.4).