University of South Australia

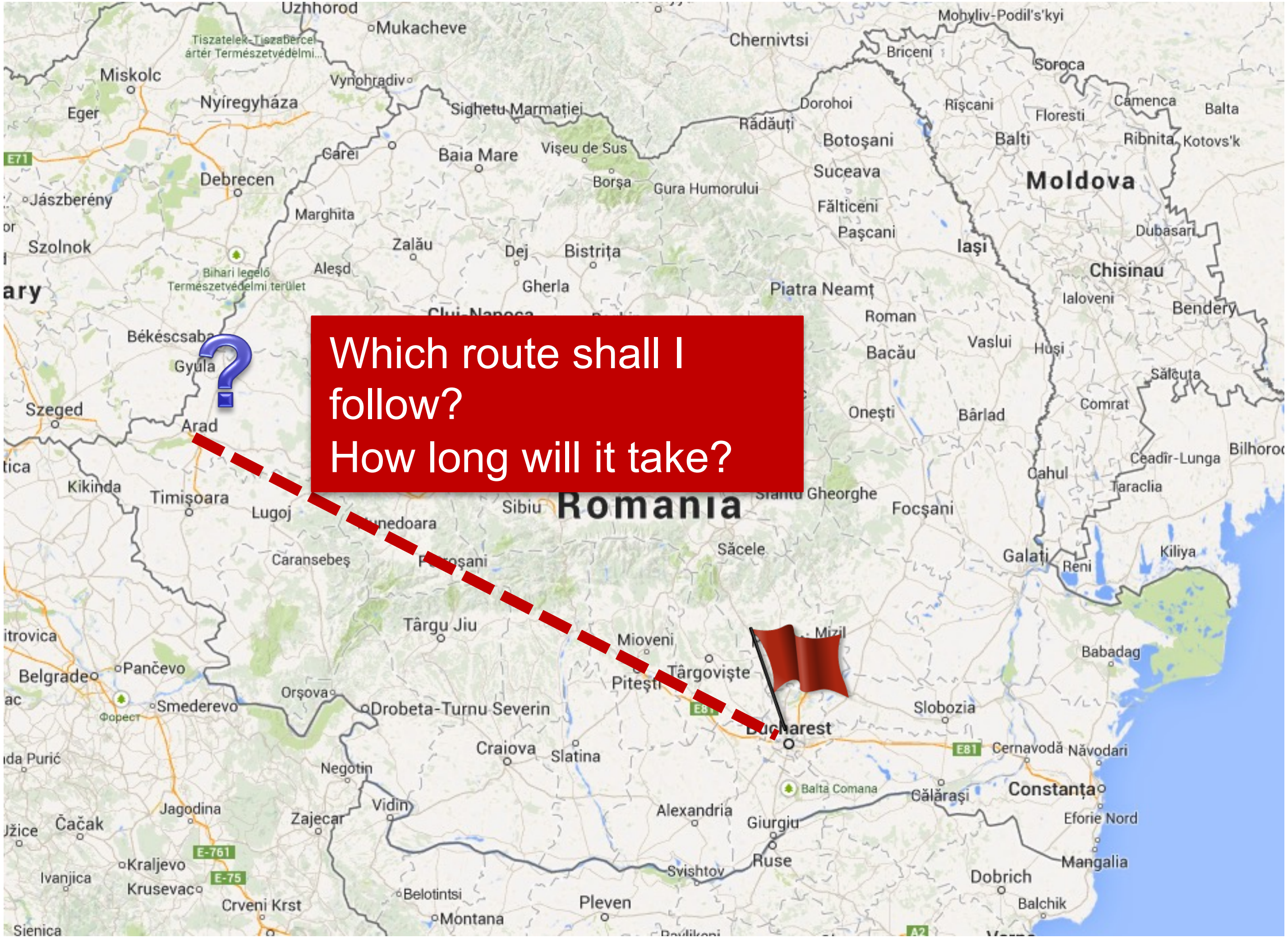
COMP 2019

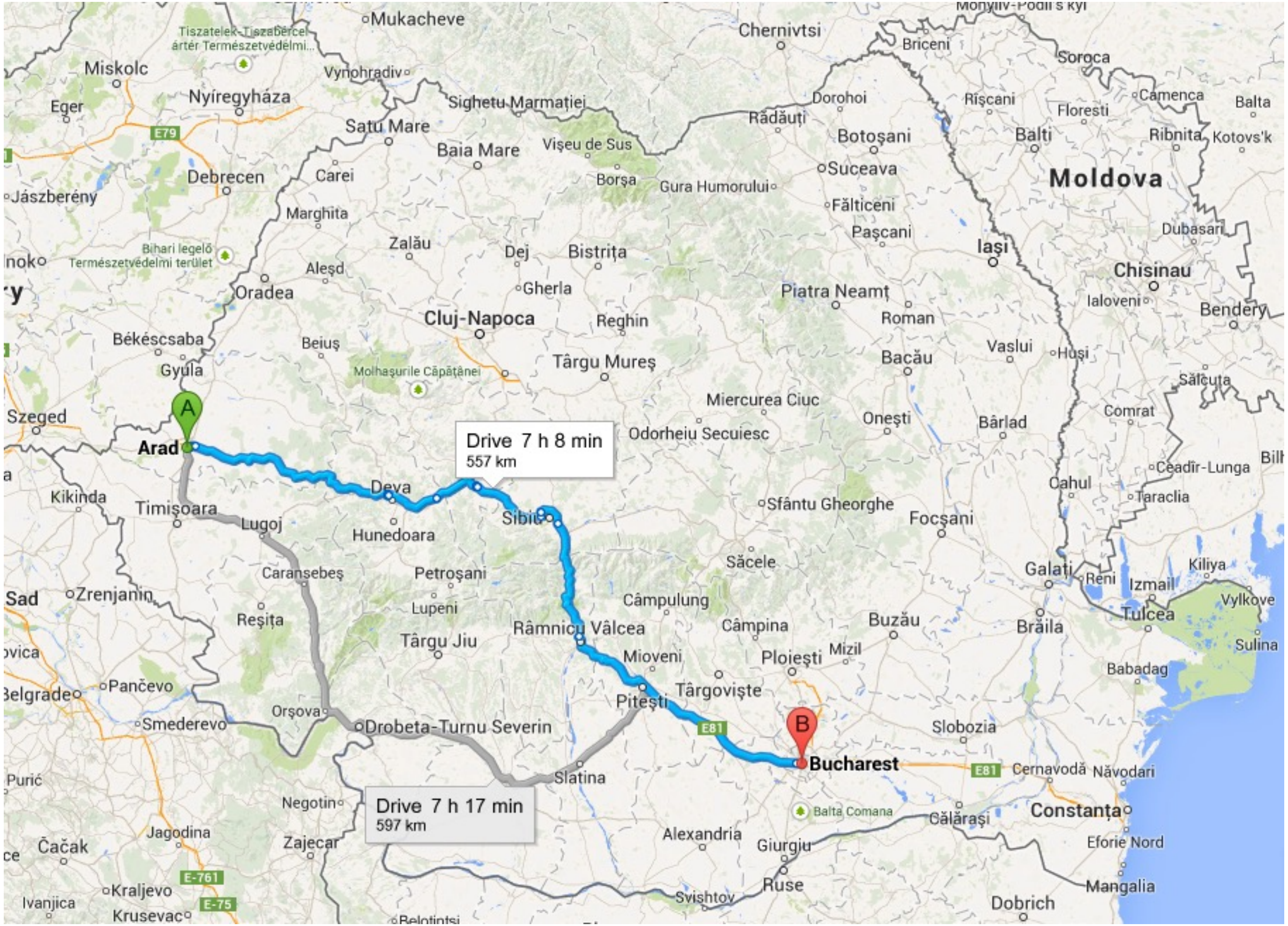
Week 2

Search-based Problem Solving

Learning Objectives

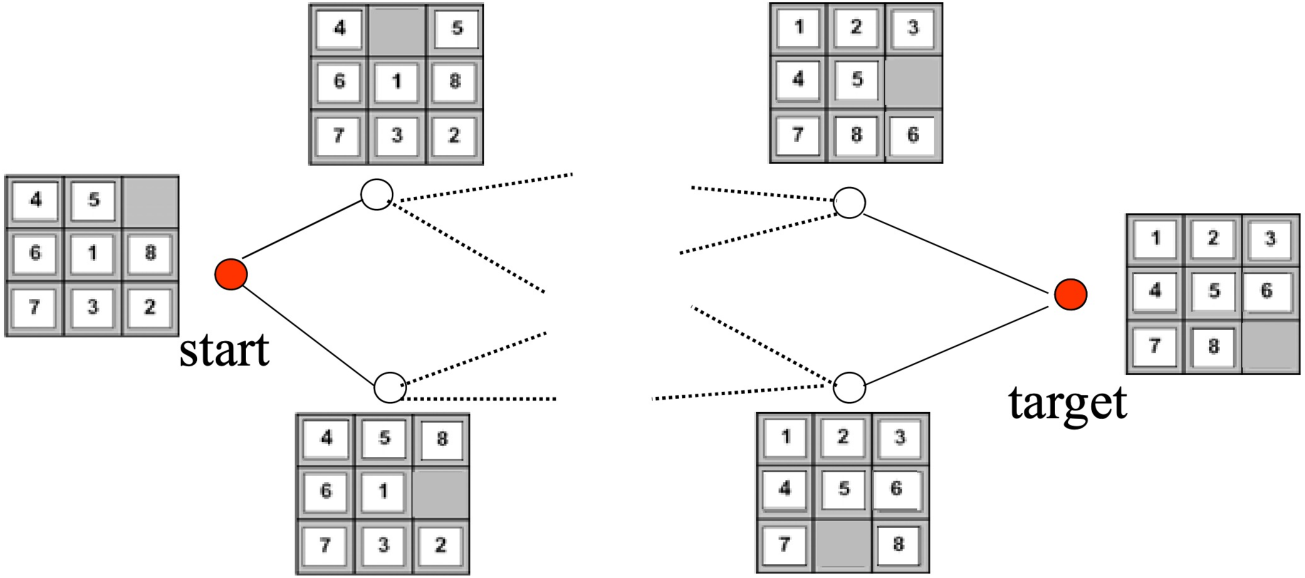
- Explain algorithms for solving problems by searching (CO1)


Which route shall I follow?
How long will it take?
Romania
Moldova
Arad
Bucharest
Uzhhorod
Mukacheve
Chernivtsi
Mohyliv-Podil's'kyi
Miskolc
Nyíregyháza
Eger
Debrecen
Jászberény
Szolnok
Békéscsaba
Gyula
Szeged
Timişoara
Lugoj
Kikinda
Caransebeş
Sibiu
Hunedoara
Petroşani
Târgu Jiu
Mioveni
Piteşti
Târgovişte
Mizil
Drobeta-Turnu Severin
Orşova
Craiova
Slatina
Alexandria
Giurgiu
Ruse
Svishtov
Pleven
Vidin
Negotin
Zajecar
Belotintsi
Montana
Belgrade
Pančevo
Smederevo
Jagodina
Čačak
Kraljevo
Krusevac
Crveni Krst
Ivanjica
Sienica
Vynohradiv
Sighetu Marmaţiei
Vişeu de Sus
Borşa
Baia Mare
Carei
Marghita
Zalău
Aleşd
Dej
Bistriţa
Gherla
Gura Humorului
Rădăuţi
Dorohoi
Botoşani
Suceava
Fălticeni
Paşcani
Iaşi
Piatra Neamţ
Roman
Bacău
Vaslui
Huşi
Oneşti
Bârlad
Focşani
Săcele
Galaţi
Reni
Slobozia
Cernavodă
Năvodari
Constanţa
Eforie Nord
Mangalia
Călăraşi
Dobrich
Balchik
Balta Comana
Babadag
Kiliya
Cahul
Taraclia
Ceadîr-Lunga
Comrat
Sălcuţa
Bendery
Chisinau
Ialoveni
Dubasari
Briceni
Soroca
Rîşcani
Floresti
Camenca
Balta
Balti
Ribnita
Kotovs'k



Drive 7 h 8 min
557 km
Drive 7 h 17 min
597 km
A
Arad
B
Bucharest
Moldova
Mukacheve
Chernivtsi
Miskolc
Eger
Nyíregyháza
Vynohradiv
Sighetu Marmației
Satu Mare
Baia Mare
Vișeu de Sus
Borșa
Debrecen
Carei
Jászberény
Marghita
Zalău
Dej
Bistrița
Aleșd
Oradea
Gherla
Cluj-Napoca
Reghin
Békéscsaba
Beiuș
Gyula
Târgu Mureș
Szeged
Deva
Sibiu
Timișoara
Lugoj
Hunedoara
Kikinda
Caransebeș
Petroșani
Zrenjanin
Reșița
Lupeni
Târgu Jiu
Râmnicu Vâlcea
Pančevo
Belgrade
Smederevo
Orșova
Drobeta-Turnu Severin
Slatina
Pitești
Mioveni
Târgoviște
Negotin
Zajecar
Jagodina
Čačak
Kraljevo
Ivanjica
Krusevac
Alexandria
Giurgiu
Ruse
Svishtov
Câmpulung
Câmpina
Ploiești
Mizil
Buzău
Sâcele
Sfântu Gheorghe
Odorheiu Secuiesc
Miercurea Ciuc
Onești
Focșani
Bacău
Roman
Piatra Neamț
Pașcani
Fălticeni
Suceava
Gura Humorului
Rădăuți
Dorohoi
Botoșani
Iași
Vaslui
Huși
Bârlad
Galați
Brăila
Slobozia
Cernavodă
Constanța
Eforie Nord
Mangalia
Călărași
Dobrich
Tulcea
Izmail
Reni
Kiliya
Vylkove
Sulina
Babadag
Chisinau
Ialoveni
Bendery
Comrat
Cahul
Taraclia
Ceadîr-Lunga
Balti
Soroca
Briceni
Rîșcani
Floresti
Camenca
Ribnita
Dubasari
Balta
Kotovs'k
Sălcuța
Năvodari
Balta Comana
Molhașurile Căpățânei
Bihari legelő Természetvédelmi terület
Tiszatelek-Tiszaberceli ártér Természetvédelmi...
E79
E81
E-761
E-75

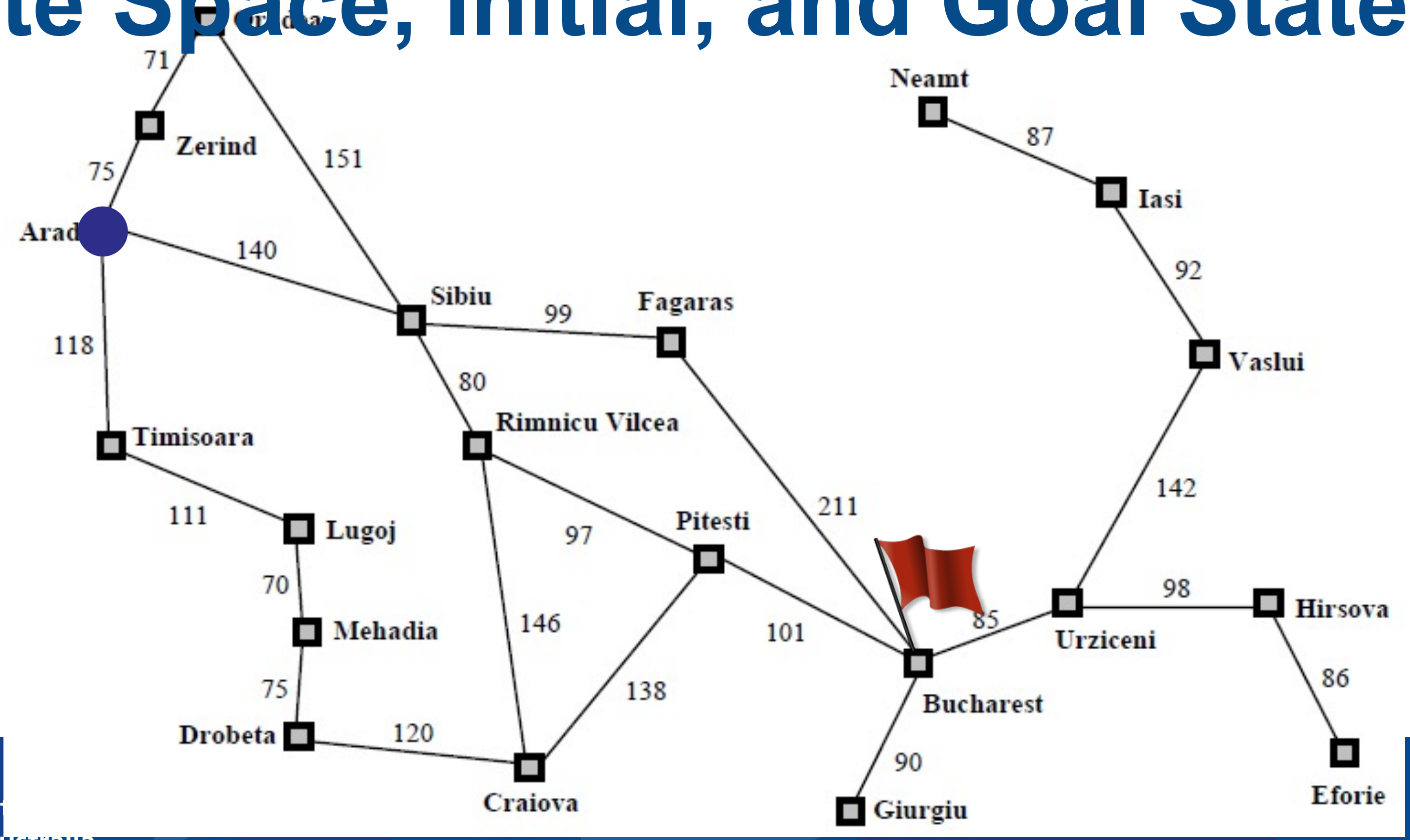
Search-based Problem Solving



Problem Definition (1)

- State Space
  - Set of all possible situations
- Initial State
  - Describes the situation from where the search for a solution begins.
- Goal State(s)
  - One or more states that exhibit some desirable property we would like to achieve. Goal states can be listed explicitly or be defined implicitly by a goal test.

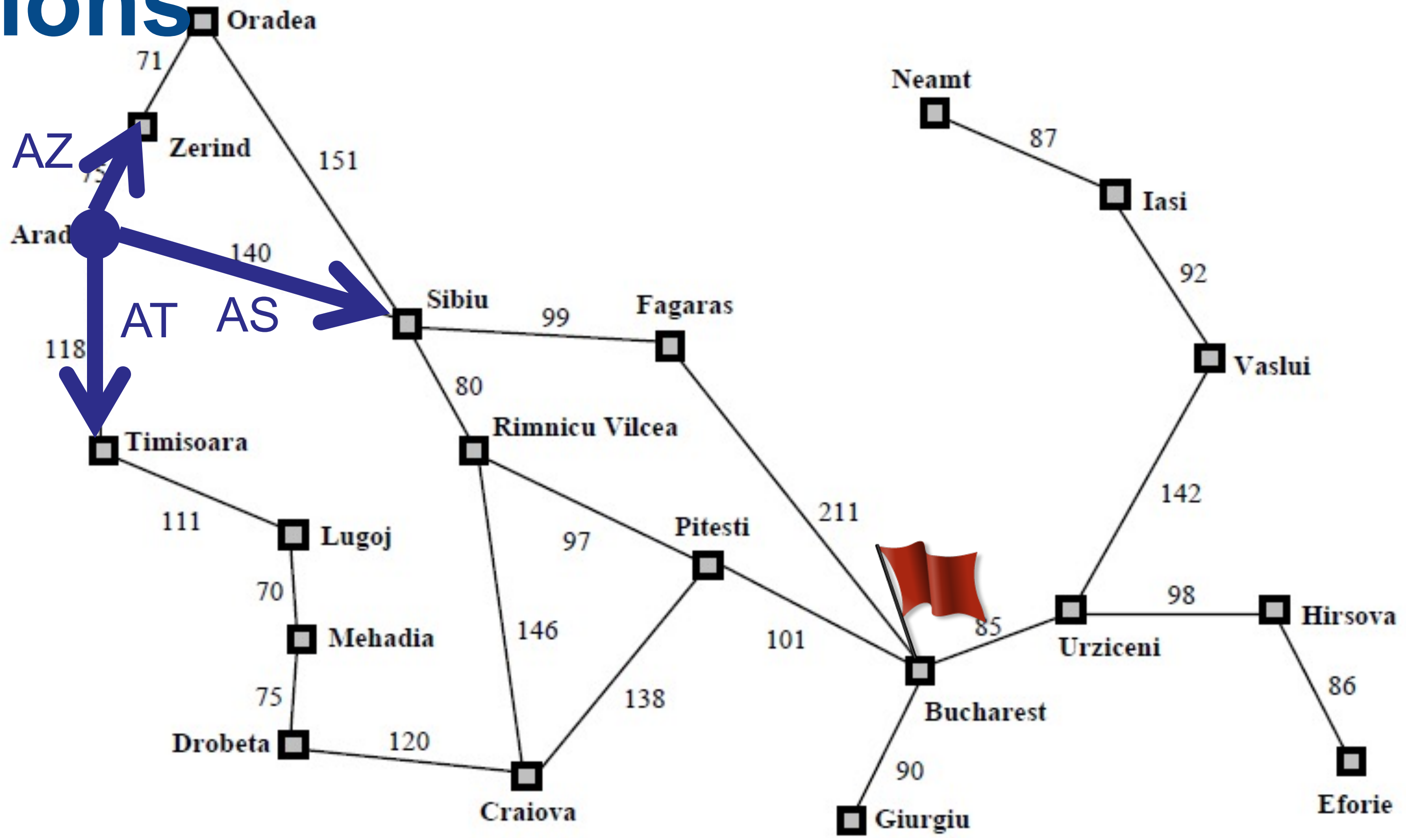
State Space, Initial, and Goal States



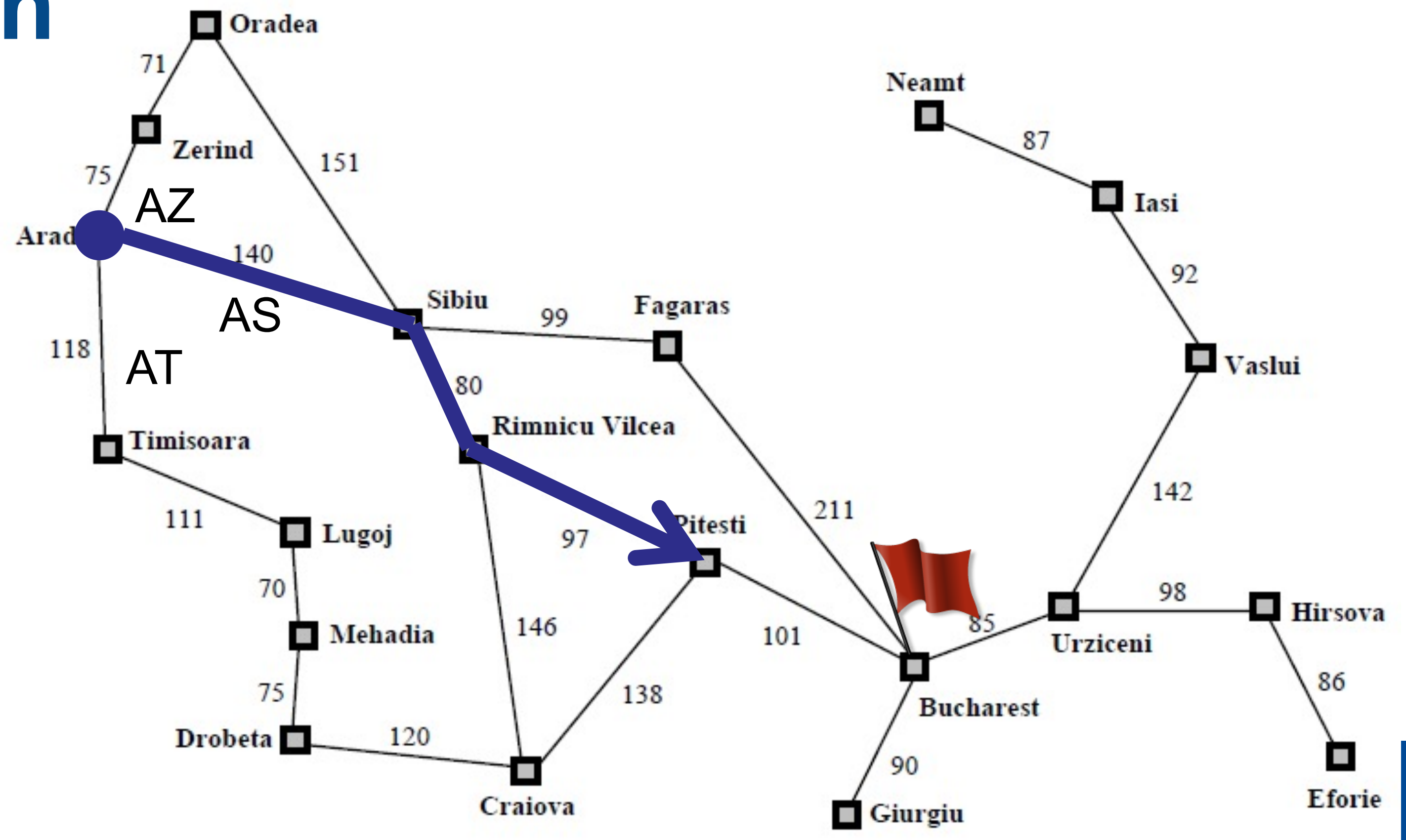
Problem Definition (2)

- Actions
  - Describe the transitions from one state to the next.
  - Define which states can be reached from the current state.
- Step Cost
  - The cost associated with an action.
  - Always nonnegative.
- Path Cost
  - The sum of all step costs of actions applied on a path from the initial state to the current state

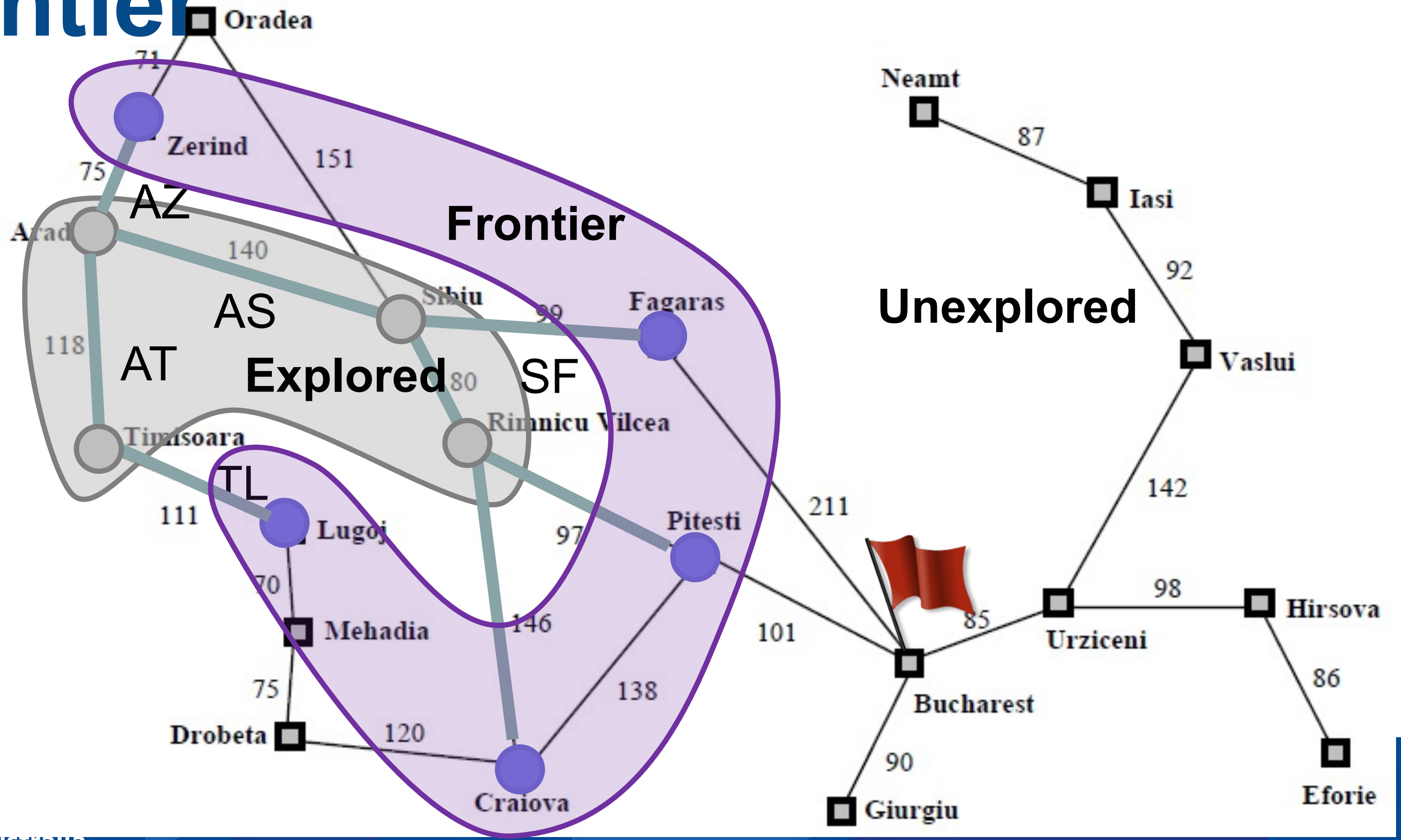
Actions


Oradea
71
Zerind
151
AZ
Arad
140
AT
AS
118
Sibiu
99
Fagaras
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Urziceni
Hirsova
85
Bucharest
86
90
Giurgiu
Eforie

Path

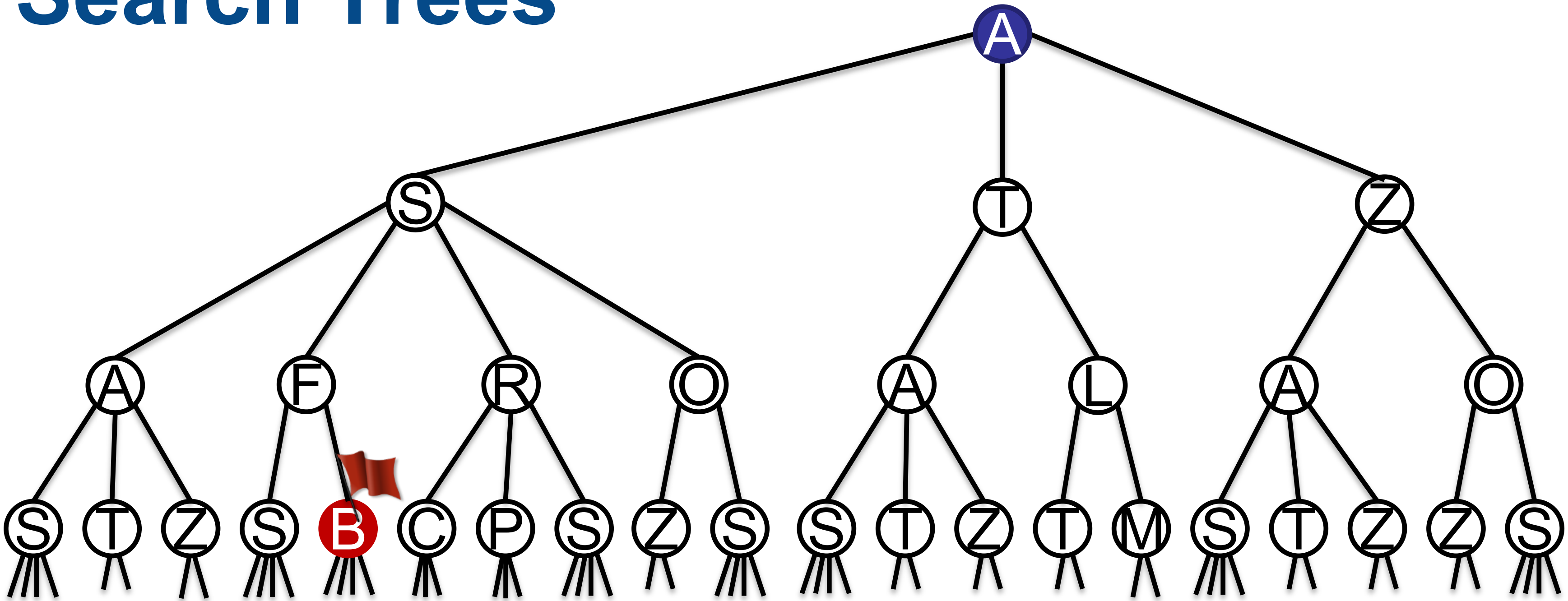

Oradea
71
Zerind
151
75
AZ
Arad
140
AS
Sibiu
99
Fagaras
118
AT
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Urziceni
Hirsova
85
86
Bucharest
90
Giurgiu
Eforie

Frontier



Oradea
71
Neamt
Zerind
151
87
75
Iasi
AZ
Arad
Frontier
140
92
Sibiu
AS
99
Fagaras
Unexplored
118
Vaslui
AT
Explored
80
SF
Rimnicu Vilcea
Timisoara
TL
142
111
211
Lugoj
97
Pitesti
70
98
85
Hirsova
146
101
Mehadia
Urziceni
86
75
138
Bucharest
Drobeta
120
90
Craiova
Giurgiu
Eforie

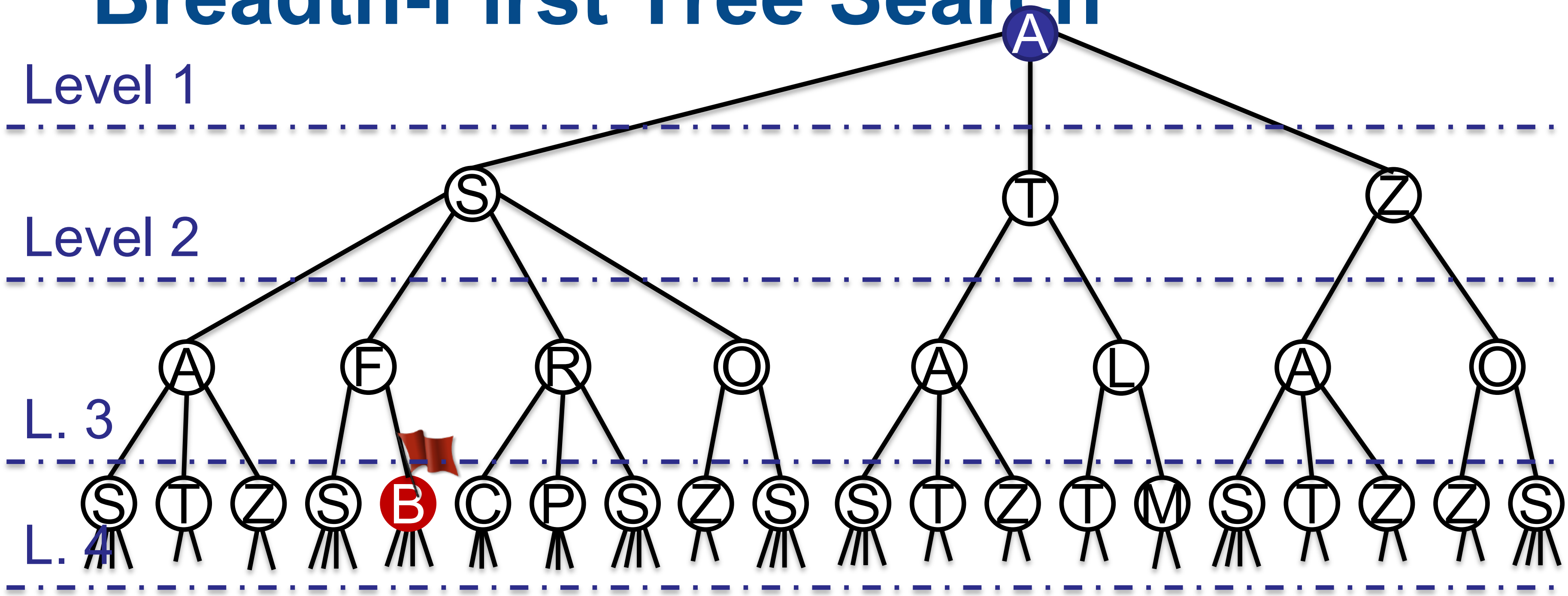
Search Trees



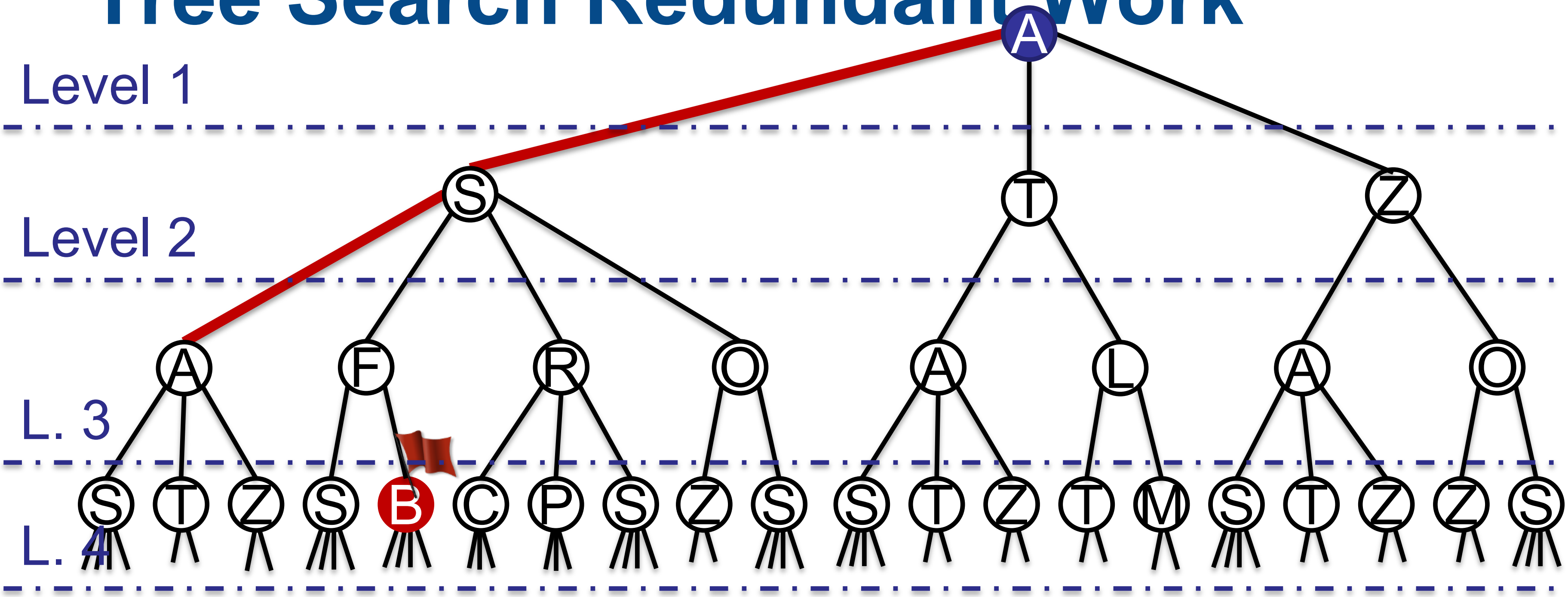
Tree Search

```
frontier = [ InitialState ]
loop:
        if frontier is empty then: return Fail
        path = Remove-Path(frontier)
        state = path.end
        if IsGoal(state) then: return path
        for a in Actions(state):
                newpath = (path, Result(state,a))
                add newpath to frontier
```

Breadth-First Tree Search



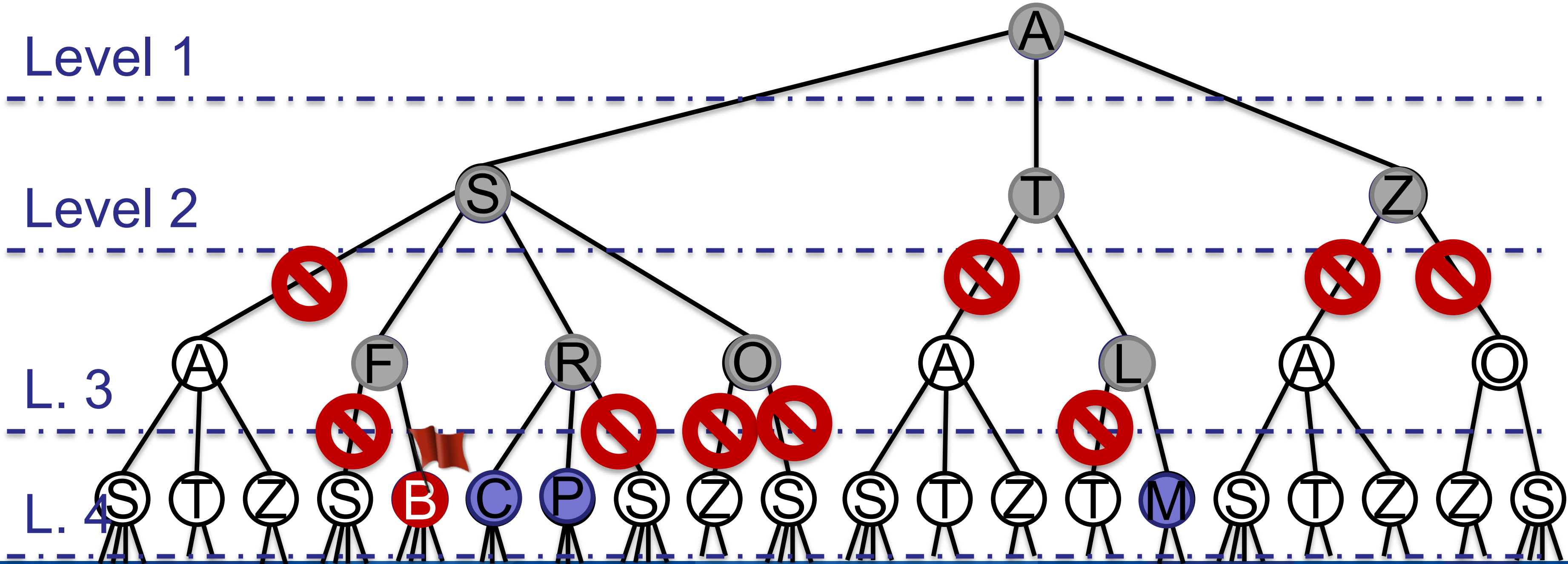
Tree Search Redundant Work



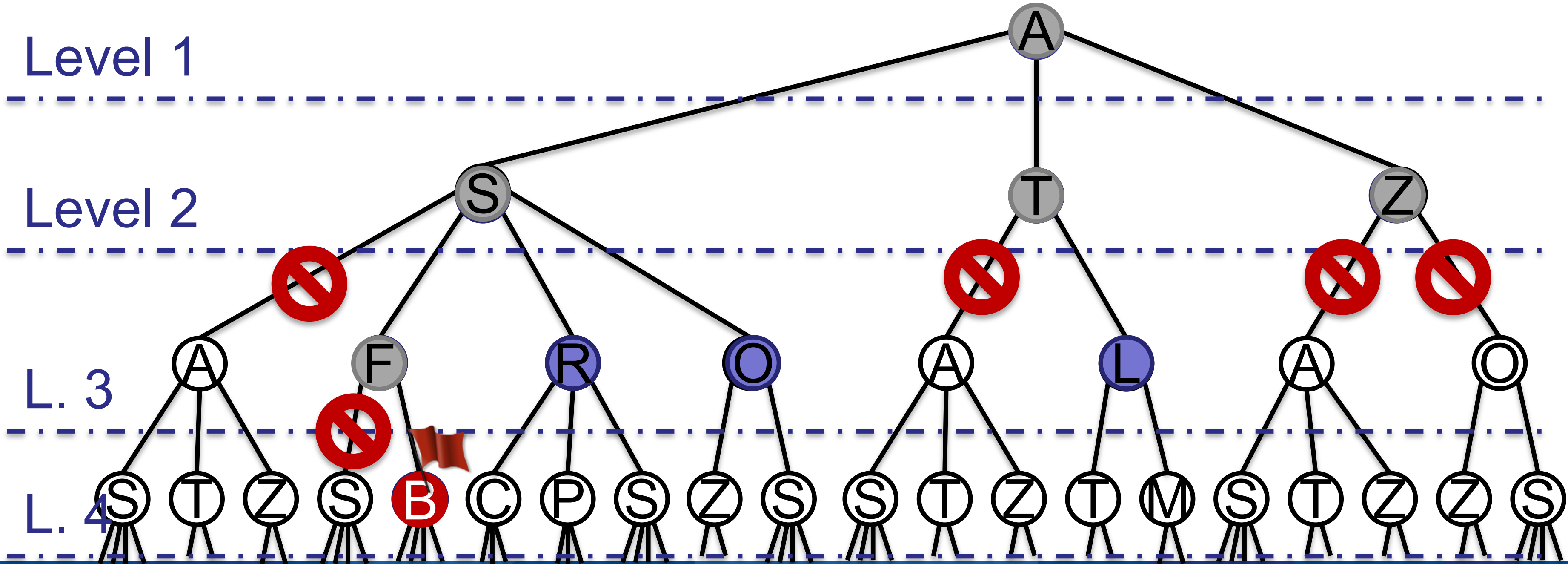
Graph Search

```
frontier = [ InitialState ]
explored = { }
loop:
        if frontier is empty then: return Fail
        path = Remove-Path(frontier)
        state = path.end
        add state to explored
        if IsGoal(state) then: return path
        for a in Actions(state):
                newpath = (path, Result(state,a))
                if newpath.end not in frontier + explored then:
                        add newpath to frontier
```

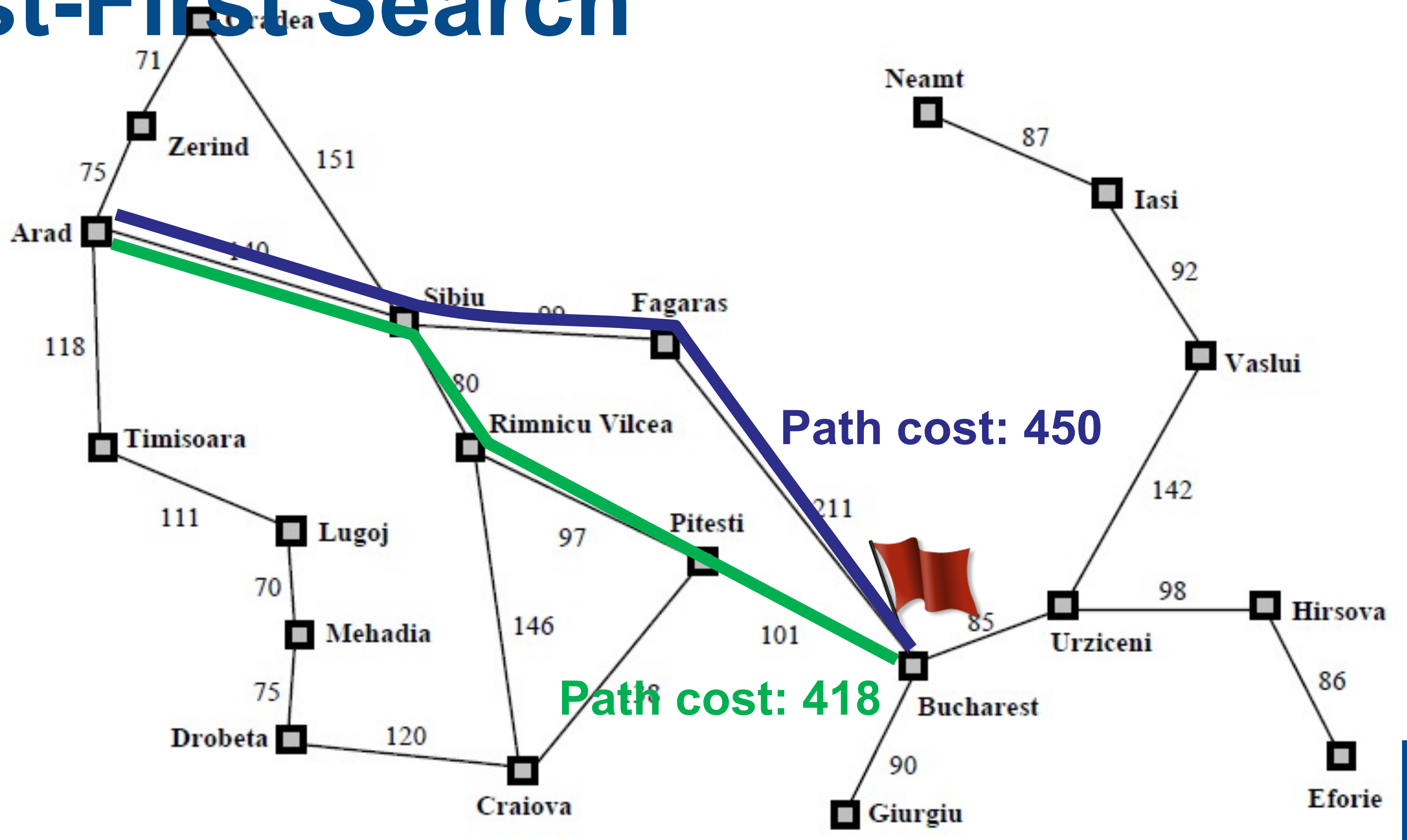
Breadth-First Graph Search



Breadth-First Graph Search Shortcut



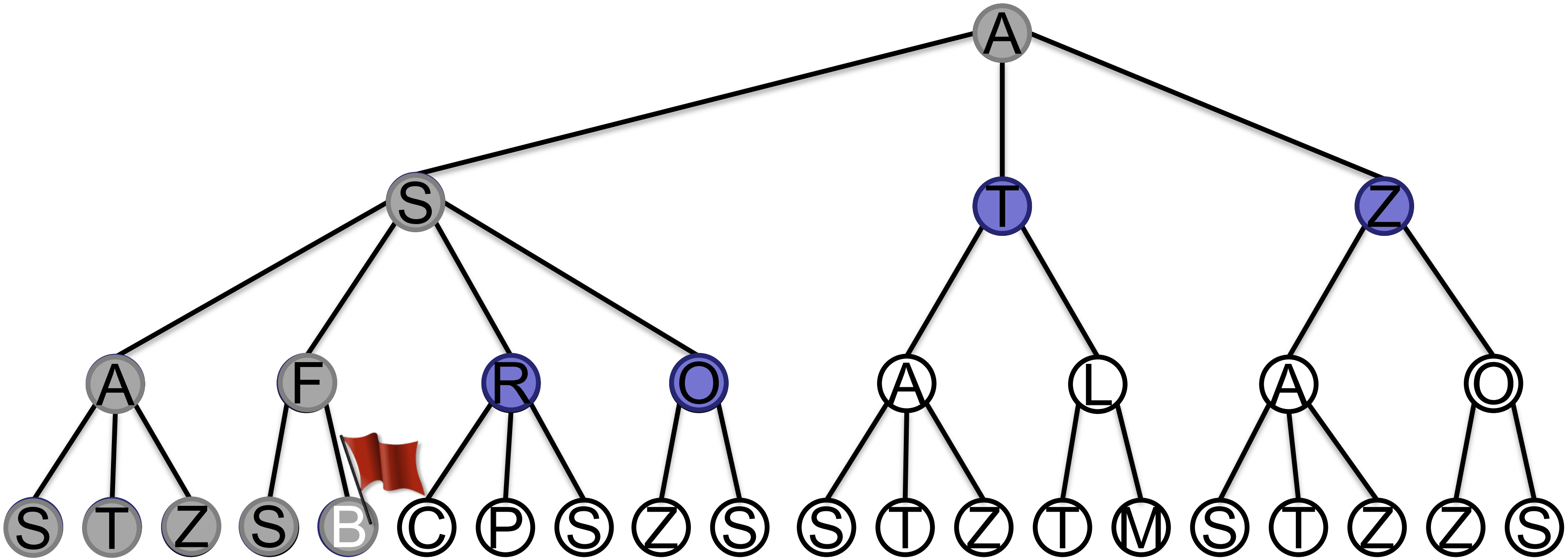
Best-First Search



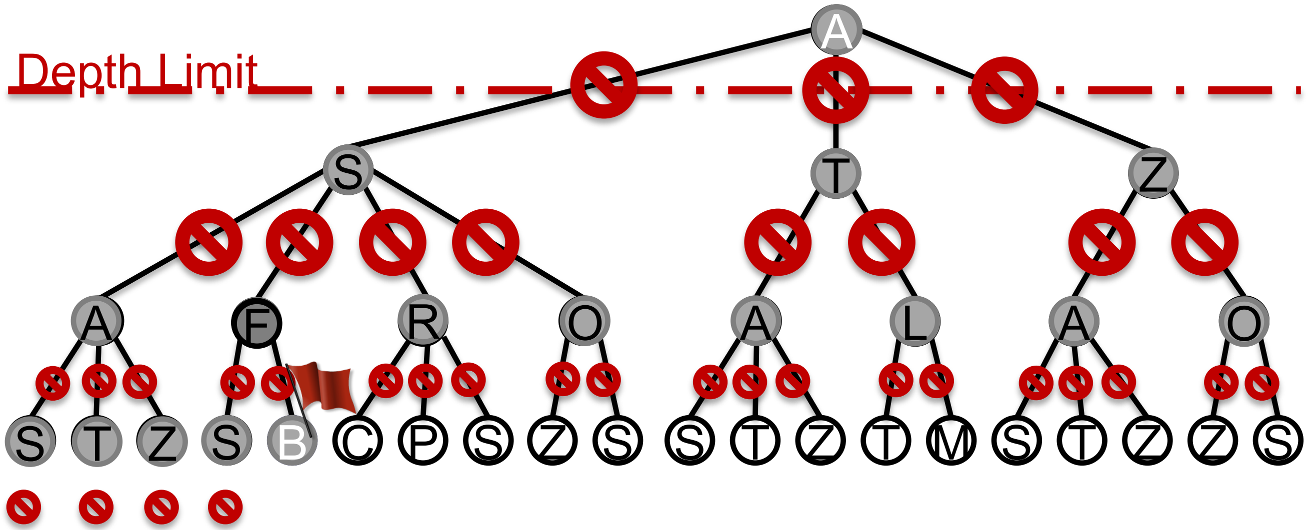
Best First Search

```
frontier = [ InitialState ]
explored = { }
loop:
        if frontier is empty then: return Fail
        path = Remove-Path(frontier)
        state = path.end
        add state to explored
        if IsGoal(state) then: return path
        for a in Actions(state):
                newpath = (path, Result(state,a))
                if newpath.end not in explored then:
                        update frontier with newpath
```

Depth-First Search



Iterative Deepening Depth-First (IDF)



IDF Algorithm

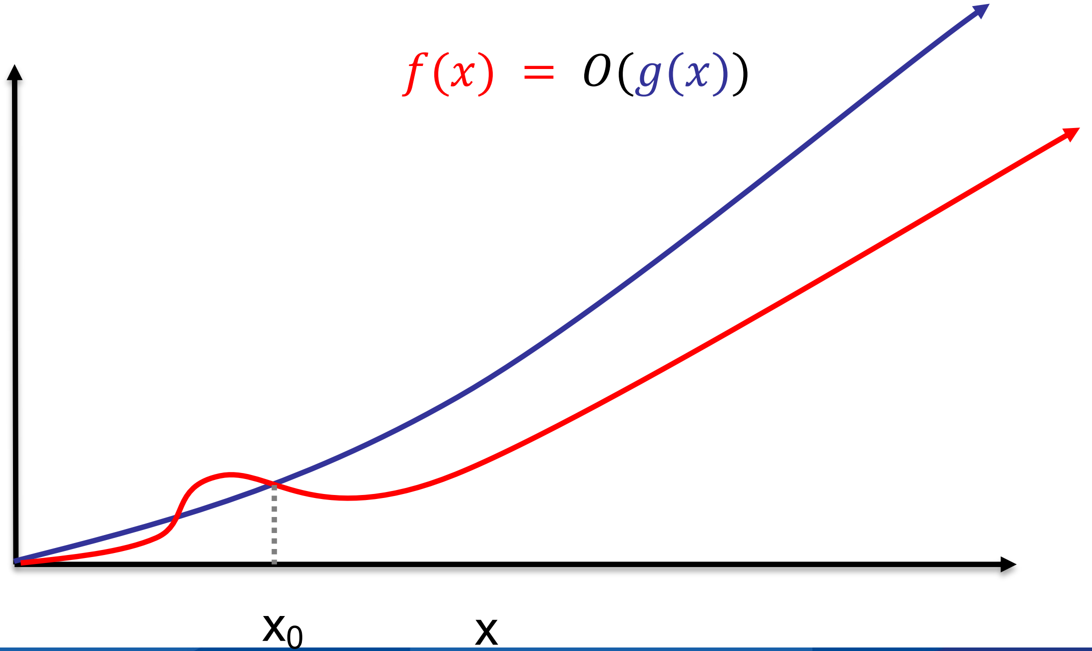
limit = 0
loop:
    limit = limit + 1
    result = DFS-limited((node),limit)
until result ≠ FAIL
return result

DFS-limited(path,depth):

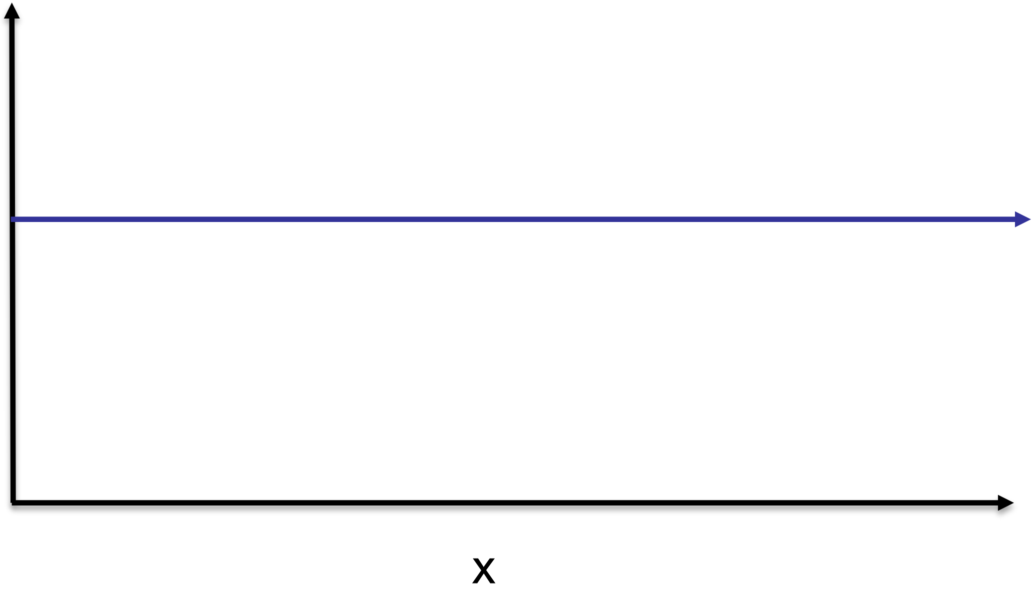
if depth=0 **then**: **return** FAIL
state = path.end
if IsGoal(state): **return** path
for a **in** Actions(state):
    newpath = (path, Result(state,a))
    result = DFS-limited(newpath, depth-1)
    if result ≠ FAIL: **return** result
return FAIL

Big O Notation

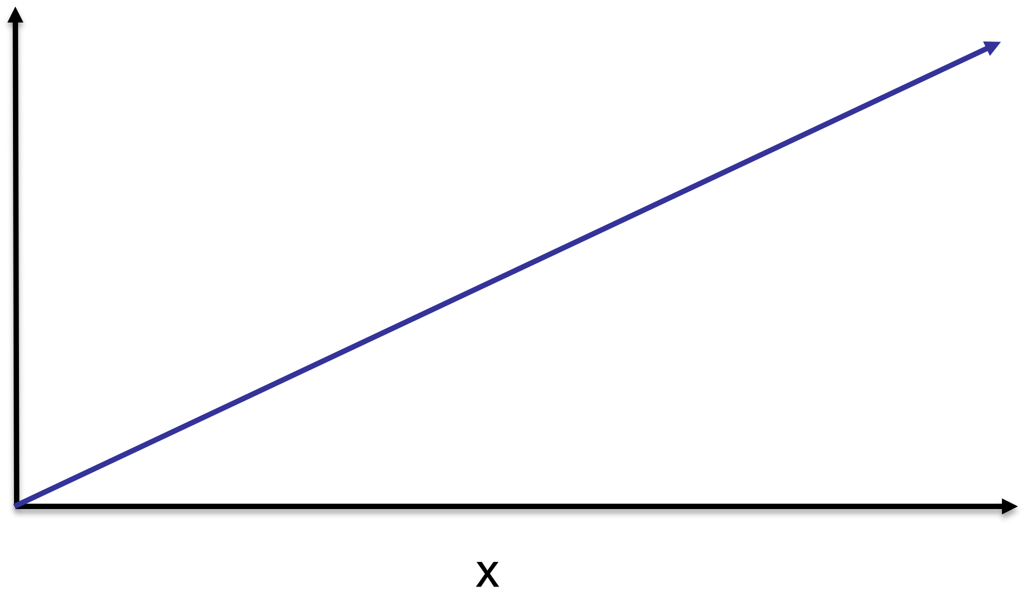
Describes the **worst-case** execution scenario by an algorithm.



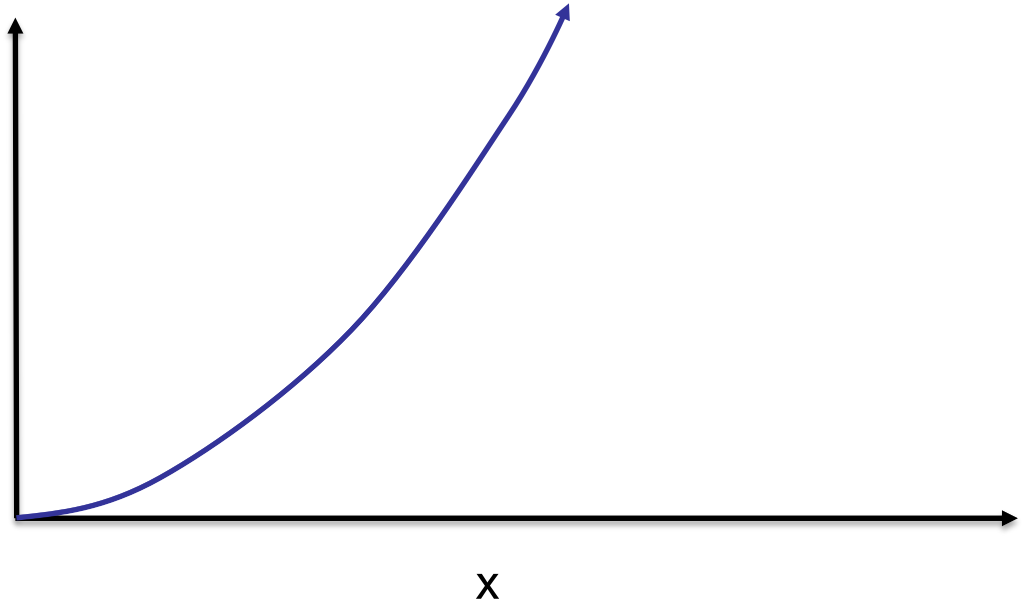
O(1): constant



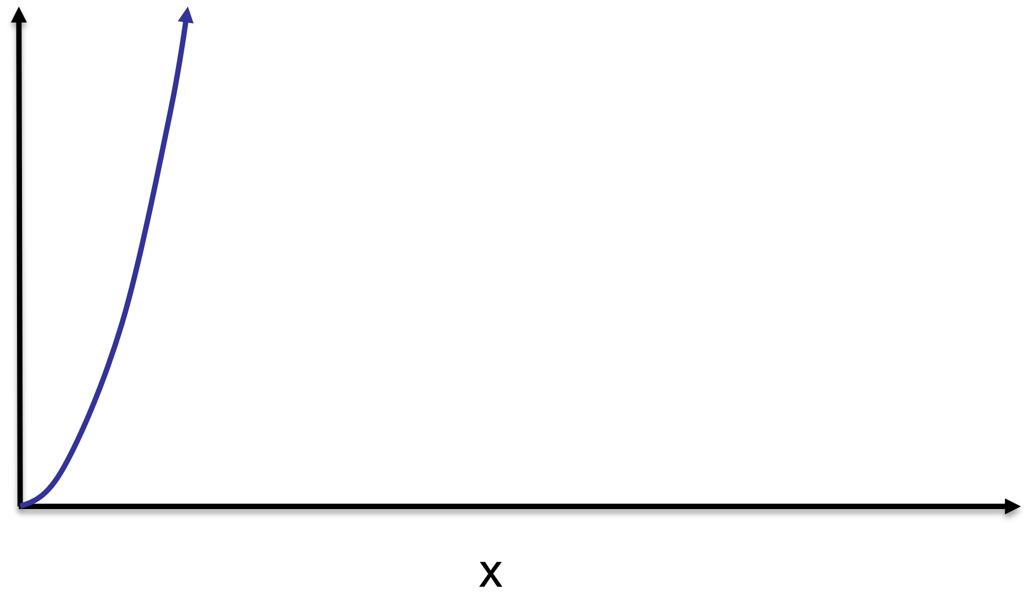
O(x): linear



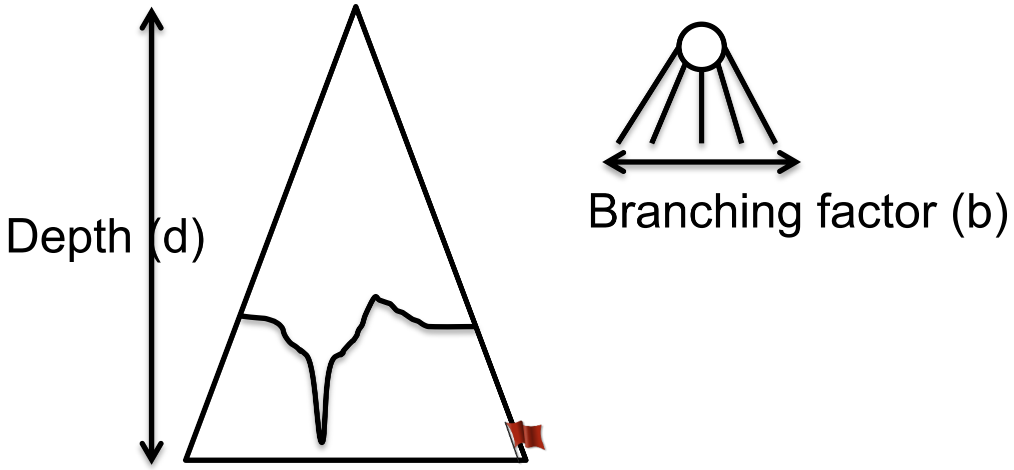
$O(x^2)$: quadratic



$O(c^x)$: exponential ($c>1$)

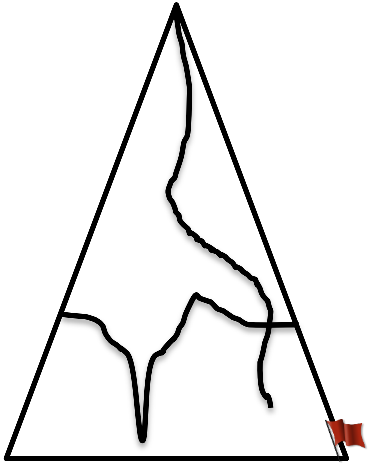


Comparison: States Traversed



Breadth FS	Best FS	Depth FS
$O(b^d)$	$O(b^d)$	$O(b^d)$

Comparison: Memory Consumption



1 GB of RAM.
10 bytes/state.
Search 10 million states/sec.

How long can we search using Breadth-/Best First Search?

Memory exhausted in 10 s!

Breadth FS	Best FS	Depth FS
$O(b^d)$	$O(b^d)$	$O(d)$

Summary

- Search algorithms are at the heart of many scientific and AI problems in practice
- Breadth-/Best-first search guarantee to find the shortest/cost-optimal solution, but suffer from memory exhaustion
- Depth-first search is memory-efficient, but may not find a solution
- Iterated Depth-first search can find solutions without memory exhaustion
- For large problems, more informed techniques are needed (next week’s topic)

University of South Australia

Questions?