STEVENS
INSTITUTE OF TECHNOLOGY
1870

Artificial Intelligence

Computer Science, CS541 - A

Jonggi Hong

Outline

1. Course administration
2. What is AI? (Chapter 1)
    1. Definitions
    2. History
3. Rational agents (Chapter 2)
4. Uninformed Search (Chapter 3)

Introduction

Welcome to CS541!

Course administration

- CS541: Artificial Intelligence
  - Prerequisites (with grade greater than or equal to D)
    - MA 222 - Probability & Statistics
    - MA 232 - Linear Algebra
  - Completed or is in process of completing 60 Units

Course administration

- Course Personnel

Jonggi Hong

jhong8@stevens.edu

Instructor

Xuting Tang

xtang18@stevens.edu

Teaching assistant

Madhuri Paluri

mpaluri@stevens.edu

Teaching assistant

Course administration

- Time
  - Lecture
    - 3:30 pm – 6:00 pm, Tuesday
  - Office hours
    - **Mon**: 1-2 pm, **Room 251** (Jonggi)
    - **Tue**: 11:30 am -12:30 pm, **Room 251** (Jonggi), 1:30-2:30 pm, **Room 226** (Madhuri)
    - **Wed**: 12:30-1:30 pm, **Room 224** (Madhuri)
    - **Thu**: 10-11 am, **Room 224** (Xuting)
    - **Fri**: 10-11 am, **Room 224** (Xuting)
  - Office hours may be held remotely in some circumstances.
    - We will make announcement in advance through Canvas.

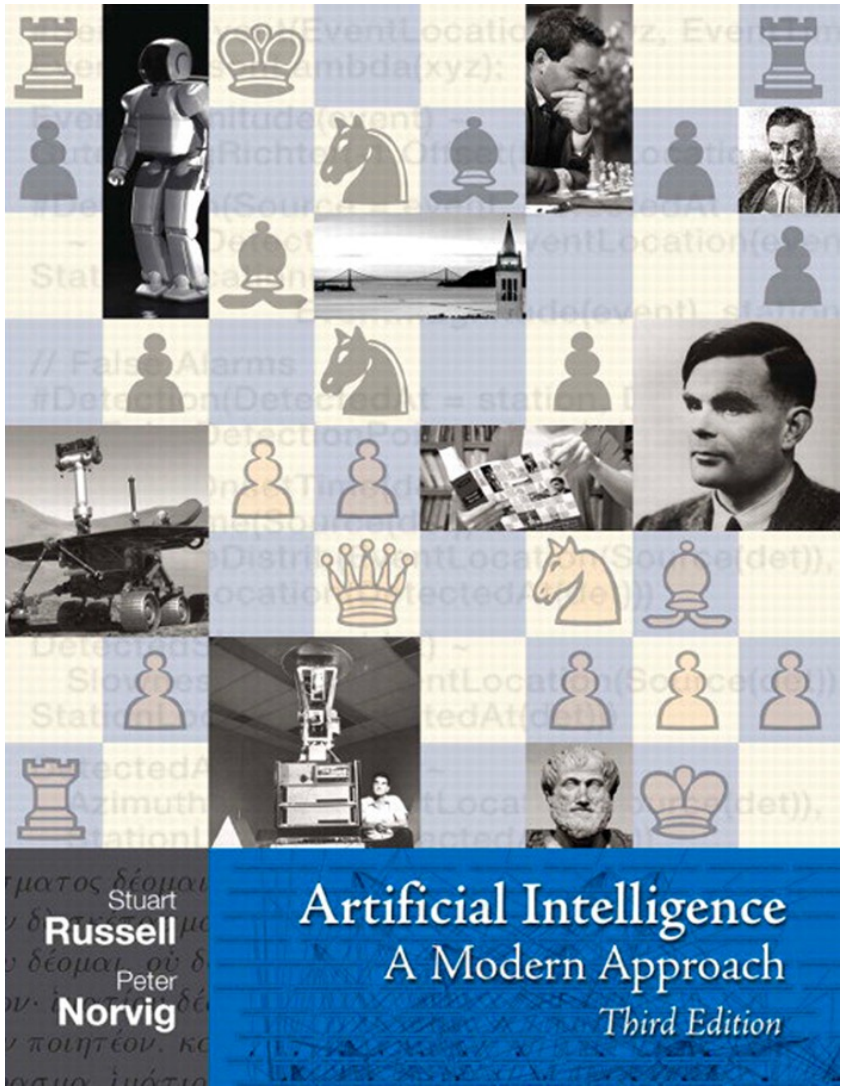
Course administration

- Textbook

  Artificial Intelligence: A Modern Approach (3rd Edition)

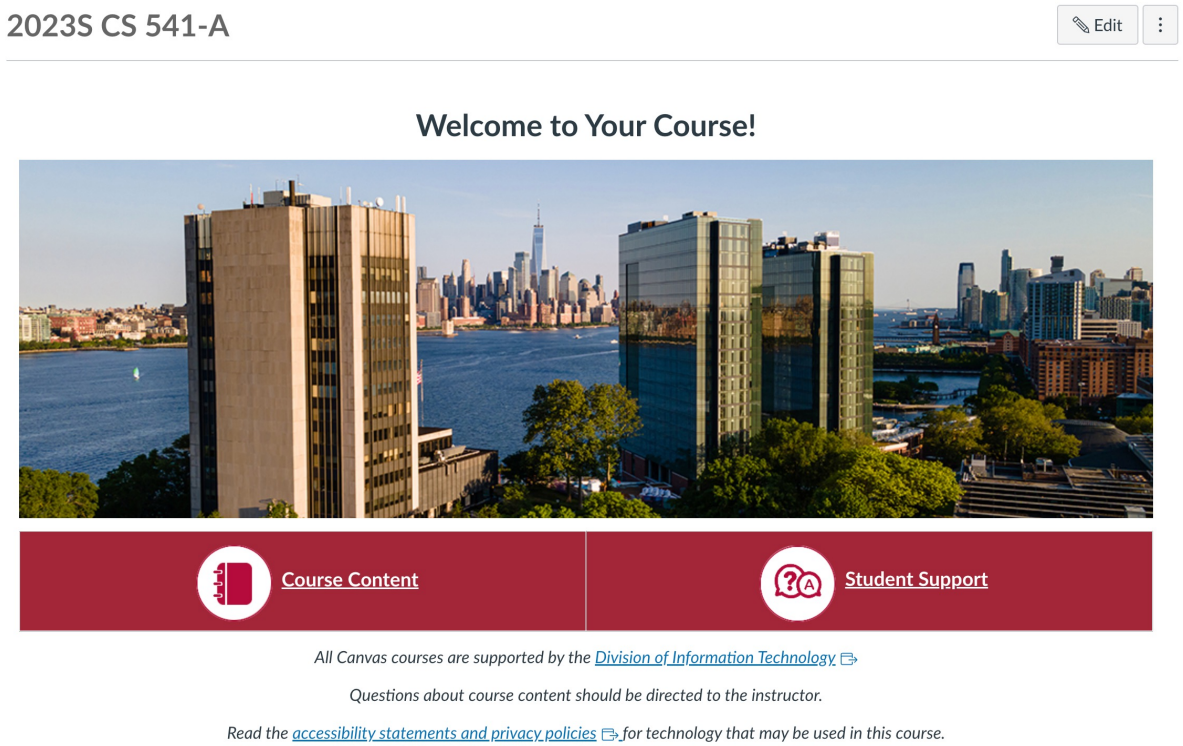
  Stuart Russell and Peter Norvig

  not required but those who want to read more



Course administration

- Website
  - Canvas (https://sit.instructure.com/courses/64281)



Course administration

- Grade
  - Homework: 40% (10% x 4)
  - Final exam: 30%
  - Project: 30%
    - Midterm report: 10%
    - Final presentation: 10%
    - Final report: 10%

Course administration

- Policy
  - Submit your homework and project files through Canvas.
  - All submissions are due at **11:59pm EST** at the due date.
  - Submit a **pdf or single zip file** containing everything including code, data, readme etc.
  - **Five days** allowed for late submissions (10% penalty per day)
  - Academic integrity
  - Bring your computer or smartphone for communication during the lecture

Schedule

Month	January		February				March				April				May		
Date	24	31	7	14	21	28	7	14	21	28	4	11	18	25	2	9	16
Lecture	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Spring Recess	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13			
Home-work		HW #1		HW #1 due	HW #2		HW #2 due		HW #3		HW #3 due	HW #4		HW #4 due			
Project			Start							Midterm report due					Presen-tation		Final report due
Exam																Final exam	

Schedule

Week 1	Introduction and intelligent agent
Week 2	Search: search strategy and heuristic search
Week 3	Search: constraint satisfaction & adversarial search
Week 4	Logic: logic agent and first order logic
Week 5	Logic: inference on first order logic
Week 6	Uncertainty and Bayesian network
Week 7	Knowledge representation
Week 8	Inference in Baysian network
Week 9	Machine learning
Week 10	Markov decision process
Week 11	Decision theory
Week 12	Decision trees and information theory
Week 13	Reinforcement Learning

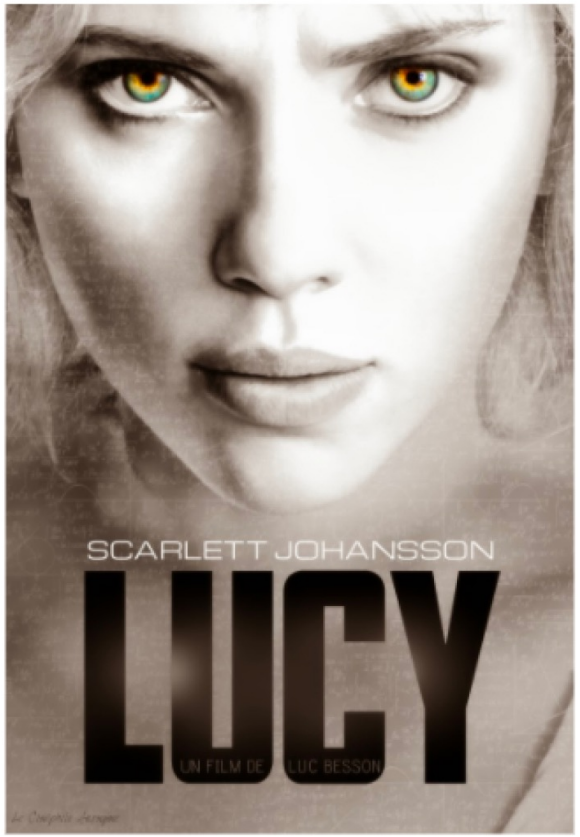
STEVENS
INSTITUTE OF TECHNOLOGY
1870

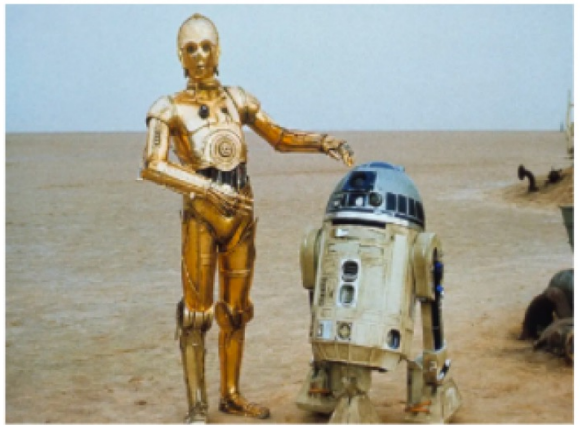
What is AI?

Chapter 1

What AI applications do you know?

What is Artificial Intelligence (AI)?





What is Artificial Intelligence (AI)?

The systems that

- think like humans.
- think rationally.
- act like humans.
- act rationally.

Acting humanly: the Turing Test approach

A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer.

Alan Turing (1950)

The computer would need to possess the following capabilities:

- **natural language processing** to enable it to communicate successfully in English
- **knowledge representation** to store what it knows or hears
- **automated reasoning** to use the stored information to answer questions and to draw
- **machine learning** to adapt to new circumstances and to detect and extrapolate patterns

Thinking humanly: The cognitive modeling approach

If we are going to say that a given program thinks like a human, we must have some way of determining how humans think.

There are three ways to do this:

- **Introspection.** Trying to catch our own thoughts as they go by
- **Psychological experiments.** Observing a person in action
- **Brain imaging.** Observing the brain in action

The interdisciplinary field of **cognitive science** brings together computer models from AI and experimental techniques from psychology to construct precise and testable theories of the human mind.

Thinking rationally: The “laws of thought” approach

The Greek philosopher Aristotle was one of the first to attempt to codify “right thinking,” that is, irrefutable reasoning processes.

“Socrates is a man; all men are mortal; therefore, Socrates is mortal.”

These laws of thought were supposed to govern the operation of the mind; their study initiated the field called **logic**.

Acting rationally: The rational agent approach

Goals are expressed in terms of the utility of outcomes.

Being rational means maximizing your **expected utility**.

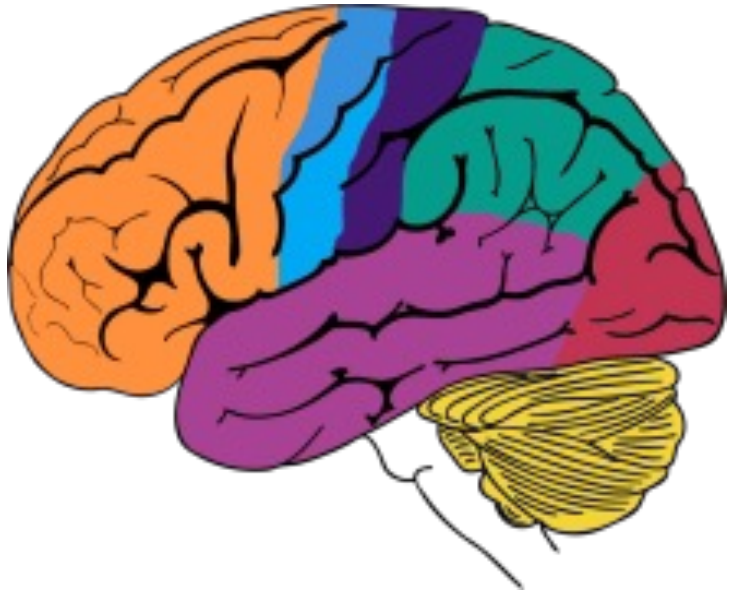
A **rational agent** is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.

Making correct (*i.e.*, logical) inferences is sometimes part of being a rational agent.

Recoiling from a hot stove is a reflex action that is usually more successful than a slower action taken after careful deliberation.

(Image from Jameica Observer)

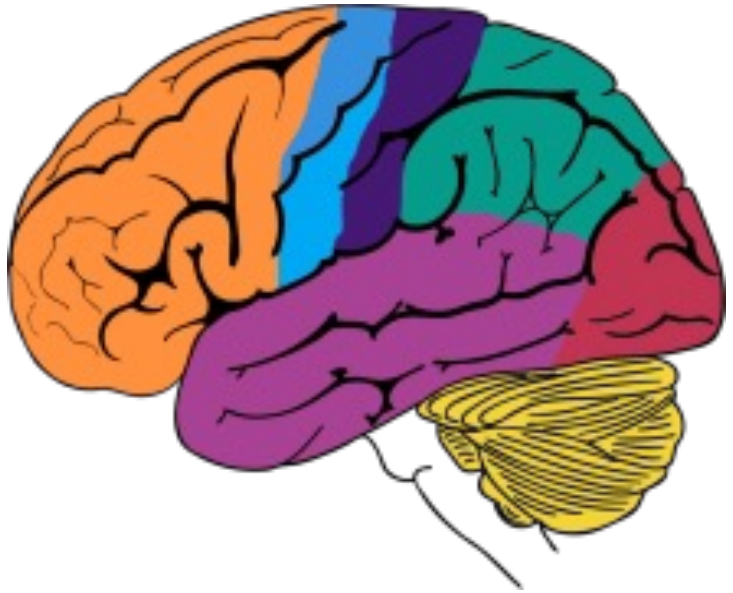
What about the brain?



(Image from NINDS)

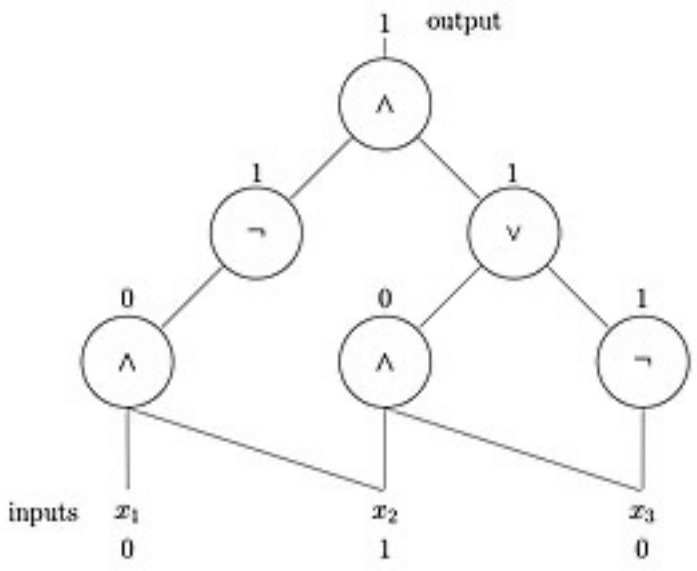
What about the brain?

- Brains (human minds) are very good at making rational decisions, but not perfect
- Brains aren’t as modular as software, so hard to reverse engineer!
- “Brains are to intelligence as wings are to flight”
- Lessons learned from the brain: memory and simulation are key to decision making



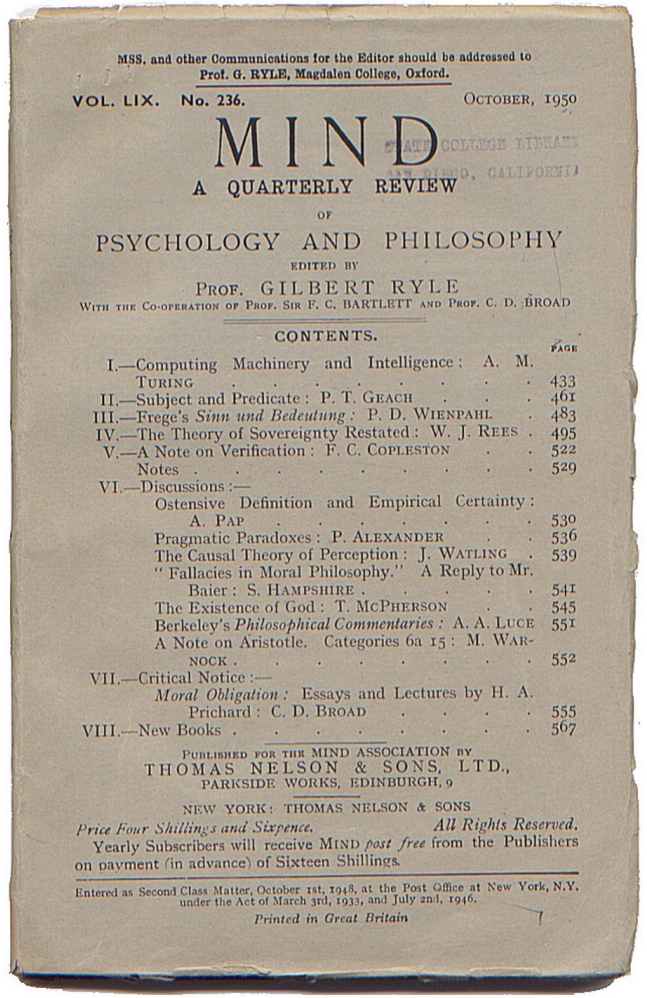
Abridged History of AI

- 1943 McCulloch & Pitts: Boolean circuit of the brain



Abridged History of AI

- 1943 McCulloch & Pitts: Boolean circuit of the brain
- 1950 Turing's "Computing machinery and intelligence"



MSS. and other Communications for the Editor should be addressed to
Prof. G. RYLE, Magdalen College, Oxford.

VOL. LIX. No. 236. OCTOBER, 1950

MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

EDITED BY

PROF. GILBERT RYLE

WITH THE CO-OPERATION OF PROF. SIR F. C. BARTLETT AND PROF. C. D. BROAD

CONTENTS.

PUBLISHED FOR THE MIND ASSOCIATION BY
THOMAS NELSON & SONS, LTD.,
PARKSIDE WORKS, EDINBURGH, 9

NEW YORK: THOMAS NELSON & SONS

Price Four Shillings and Sixpence.

Yearly Subscribers will receive MIND *post free* from the Publishers on payment (in advance) of Sixteen Shillings.

Entered as Second Class Matter, October 1st, 1948, at the Post Office at New York, N.Y. under the Act of March 3rd, 1933, and July 2nd, 1946.

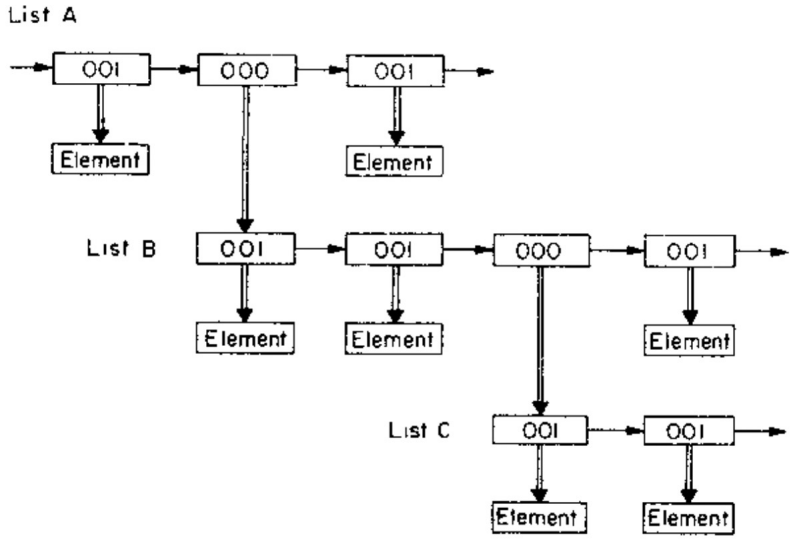
Printed in Great Britain

(Image from Manhattan Rare Book Company)

Abridged History of AI

- 1943 McCulloch & Pitts: Boolean circuit of the brain
- 1950 Turing's "Computing machinery and intelligence"
- 1950s Early AI Programs including Samuel's checkers, Newell and Simon's Logic Theorist

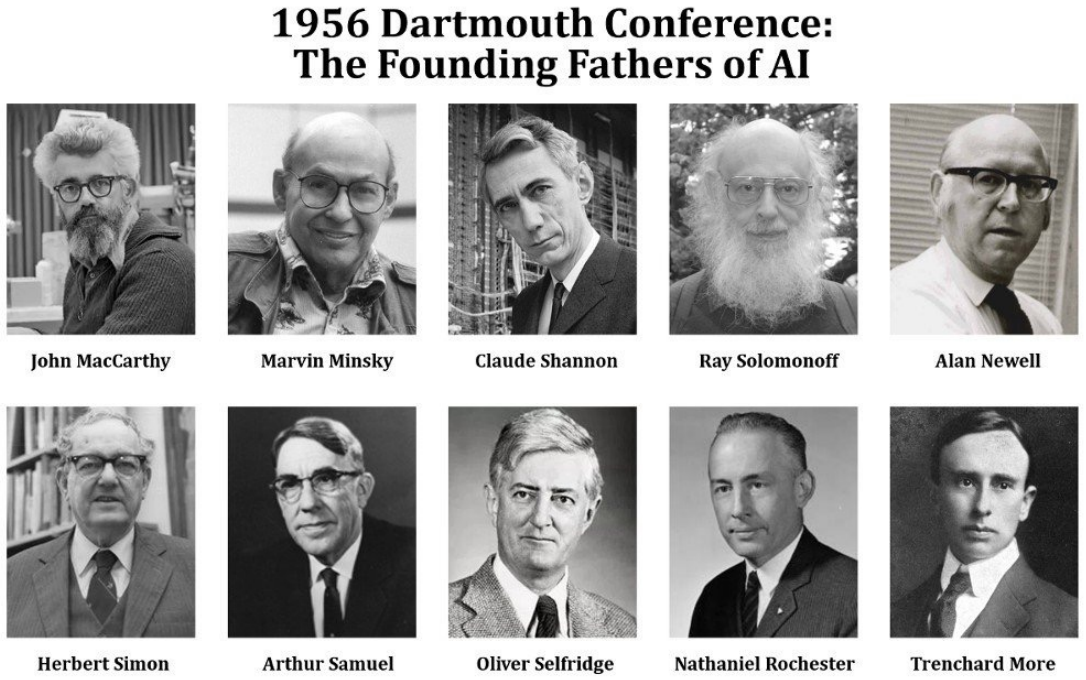
(Image from Aishwarya Srinivasan)



(A. Newell and J. Shaw, 1957)

Abridged History of AI

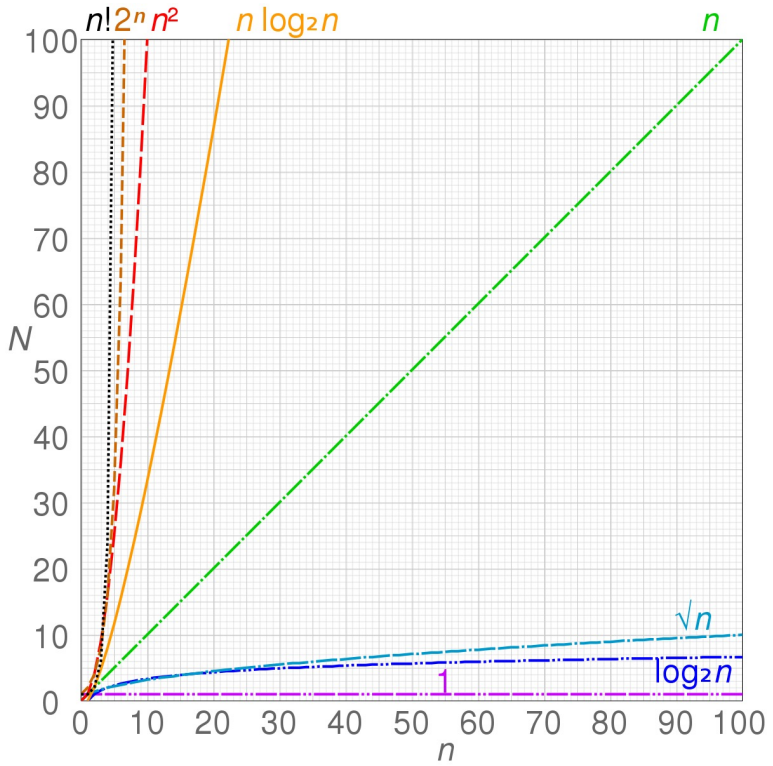
- 1943 McCulloch & Pitts: Boolean circuit of the brain
- 1950 Turing's "Computing machinery and intelligence"
- 1950s Early AI Programs including Samuel's checkers, Newell and Simon's Logic Theorist
- 1956 Dartmouth meeting: "Artificial Intelligence"

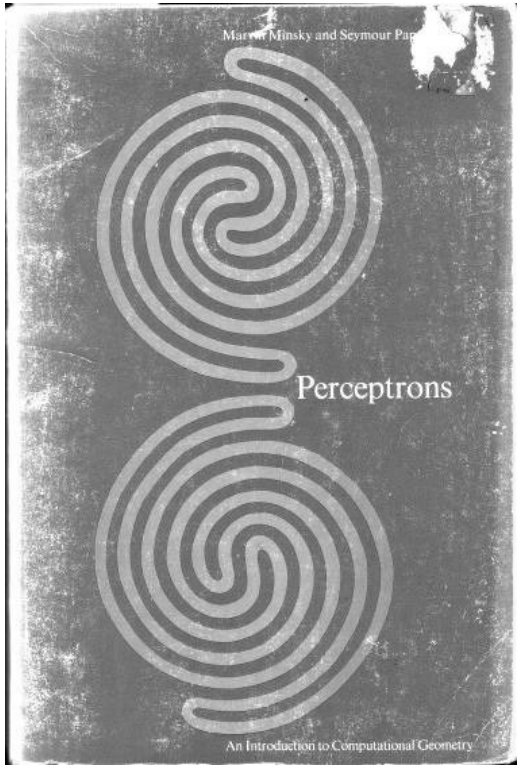


(Image from scienceabc.com)

Abridged History of AI

- 1966-1973 Problems with scaleability, Perceptron book





Abridged History of AI

- 1966-1973 Problems with scaleability, Perceptron book
- 1970s Knowledge-based systems (e.g., DENDRAL by Buchanan *et al.*, 1969)

Abridged History of AI

- 1966-1973 Problems with scaleability, Perceptron book
- 1970s Knowledge-based systems (e.g., DENDRAL by Buchanan *et al.*, 1969)
- 1980s Expert-systems industry

The **AI industry** boomed from a few million dollars in 1980 to billions of dollars in 1988, including hundreds of companies building expert systems, vision systems, robots, and software and hardware specialized for these purposes.

Soon after that came a period called the “AI Winter,” in which many companies fell by the wayside as they failed to deliver on extravagant promises.

Abridged History of AI

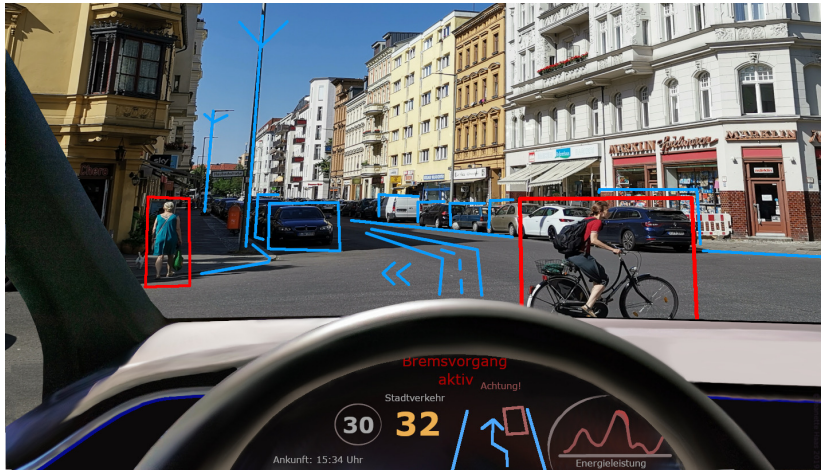
- 1966-1973 Problems with scaleability, Perceptron book
- 1970s Knowledge-based systems (e.g., DENDRAL by Buchanan *et al.*, 1969)
- 1980s Expert-systems industry
- 1988-now Probabilistic and decision theoretic methods

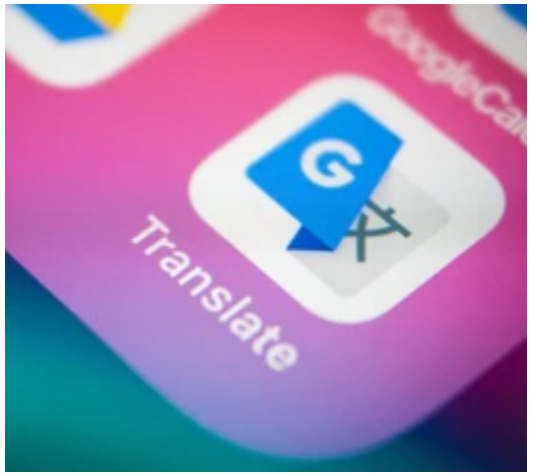
build on existing theories than to propose brand-new ones, to base claims on **rigorous theorems or hard experimental evidence** rather than on intuition, and to show relevance to **real-world applications** rather than toy examples.

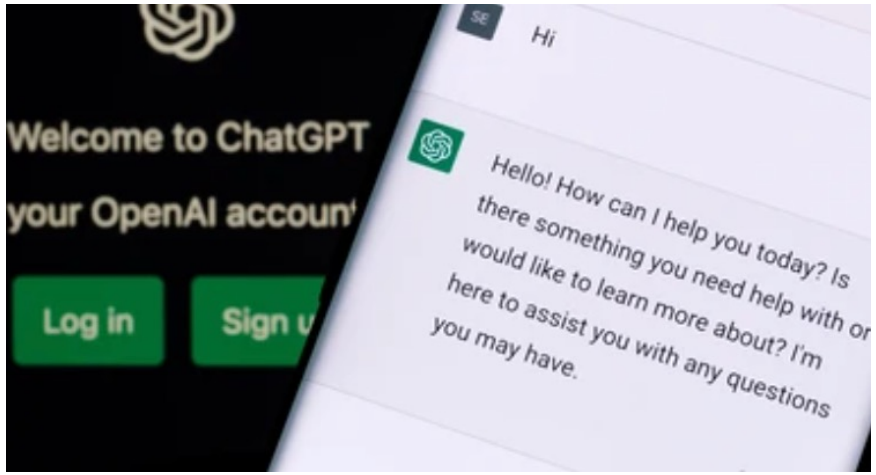
Abridged History of AI

- Now Significant progress in machine learning and industry interest
  - Focus on solving specific problems
  - Heavy use of probability theory, decision theory, statistics,...
  - Very large datasets
    - It makes more sense to worry about the **data** and be less picky about what algorithm to apply.
    - A mediocre algorithm with 100 million words of unlabeled training data outperforms the best-known algorithm with 1 million words.
    - An image processing algorithm (Hays and Efros, 2007) was poor when they used a collection of only ten thousand photos, but crossed a threshold into excellent performance when they grew the collection to two million photos.
  - AI Spring?

The state of the art







STEVENS
INSTITUTE OF TECHNOLOGY
1870

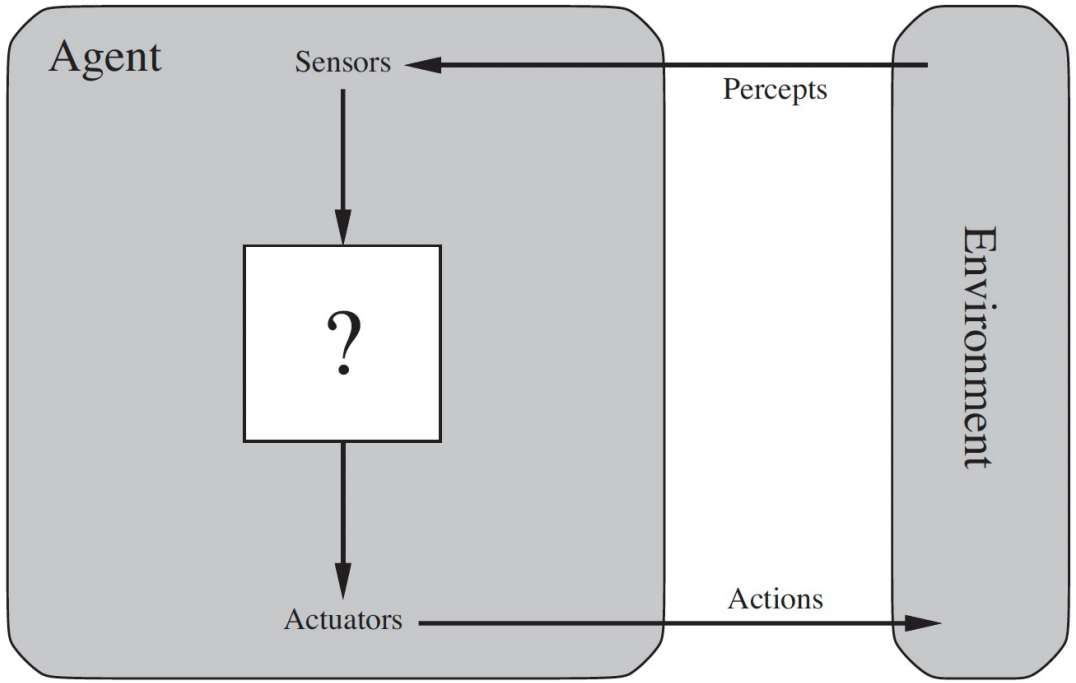
Intelligent Agents

Chapter 2

Rational agents

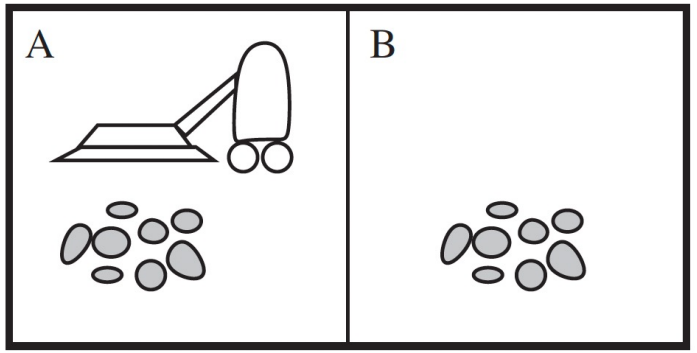
- Agent
  - An entity that perceives and acts.
  - **Agent function** is a function from percept to actions.
  - It generates a sequence of actions according to the percepts it receives.

- Rational agent
  - An agent that does the right thing.
  - The sequence of actions is desirable.
  - This notion of desirability is captured by a **performance measure**.



Example: vacuum-cleaner agent

- Percept
  - Which square it is in
  - Whether there is dirt in the square
- Action
  - move left
  - move right
  - suck up the dirt
  - do nothing



Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
⋮	⋮

What is the right way to fill out the table?

Task environment

- PEAS description
  - **P**erformance
  - **E**nvironment
  - **A**ctuators
  - **S**ensors

Task environment: automated taxi driver

- PEAS description
  - **P**erformance: safe, fast, legal, comfortable trip, profits
  - **E**nvironment: roads, other traffic, pedestrians, customers
  - **A**ctuators: steering, accelerator, brake, signal, horn, display
  - **S**ensors: cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, ...

Properties of task environments

- Fully observable vs. Partially observable

Properties of task environments

- Fully observable vs. Partially observable
- Deterministic vs. Stochastic

Properties of task environments

- Fully observable vs. Partially observable
- Deterministic vs. Stochastic
- Discrete vs. Continuous

Properties of task environments

- Fully observable vs. Partially observable
- Deterministic vs. Stochastic
- Discrete vs. Continuous
- Single agent vs. Multiagent
- …

STEVENS
INSTITUTE OF TECHNOLOGY
1870

15 min. break

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Uninformed search

Chapter 3

Search

- Search was one of the first topics studied in AI
  - Newell and Simon (1961) General Problem solver
- Central component to many AI systems
  - Automated reasoning, theorem proving, robot navigation, scheduling, game playing, ...

Search problems

- A search problem can be defined formally by five components:
  - Initial state (states)
  - Actions
  - Transition model (actions, costs, successors)
  - Goal test
  - Path cost
- A solution to a problem is an action sequence that leads from the initial state to a goal state.
- An optimal solution has the lowest path cost among all solutions.

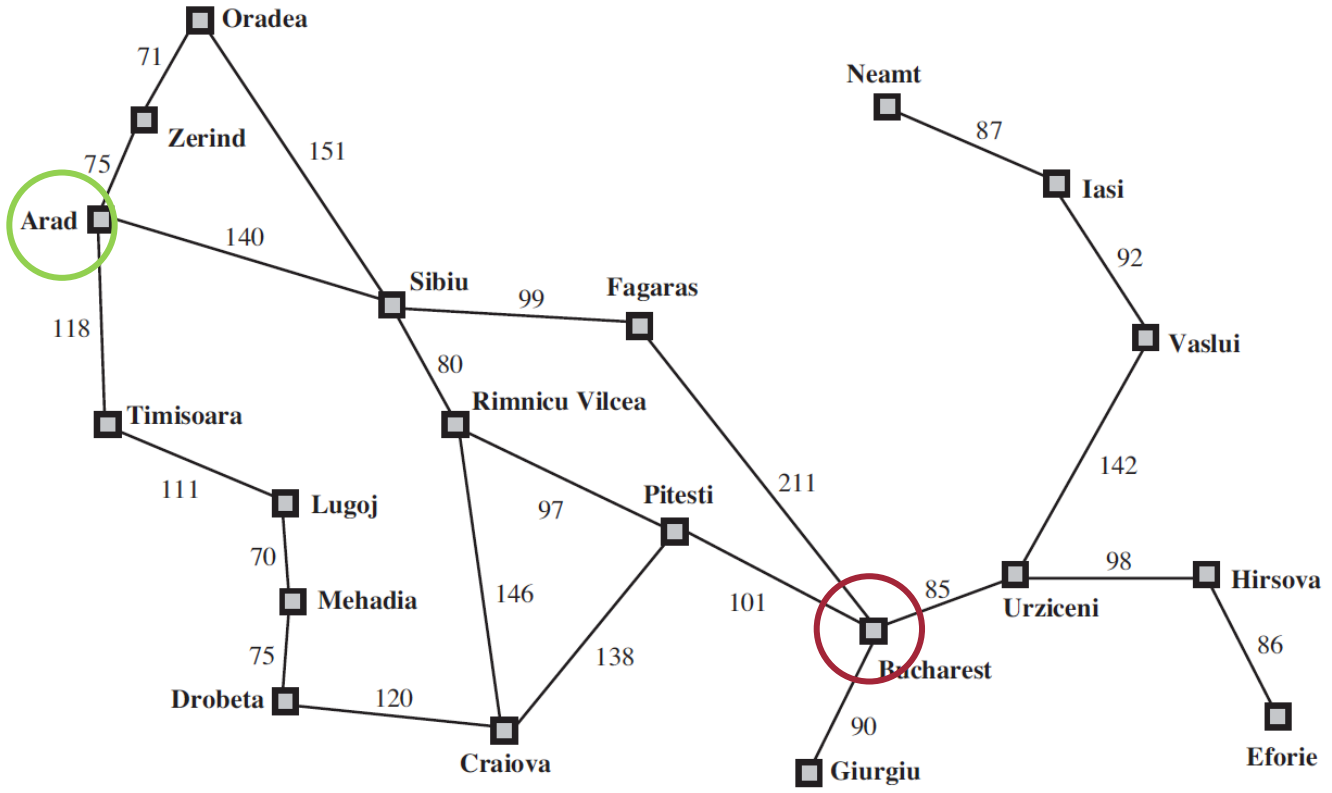
Uninformed search

- The strategies have no additional information about states beyond that provided in the problem definition.
  - Strategies that know whether one non-goal state is **more promising** than another are called informed search or heuristic search strategies

- All search strategies are distinguished by the order in which nodes are expanded.

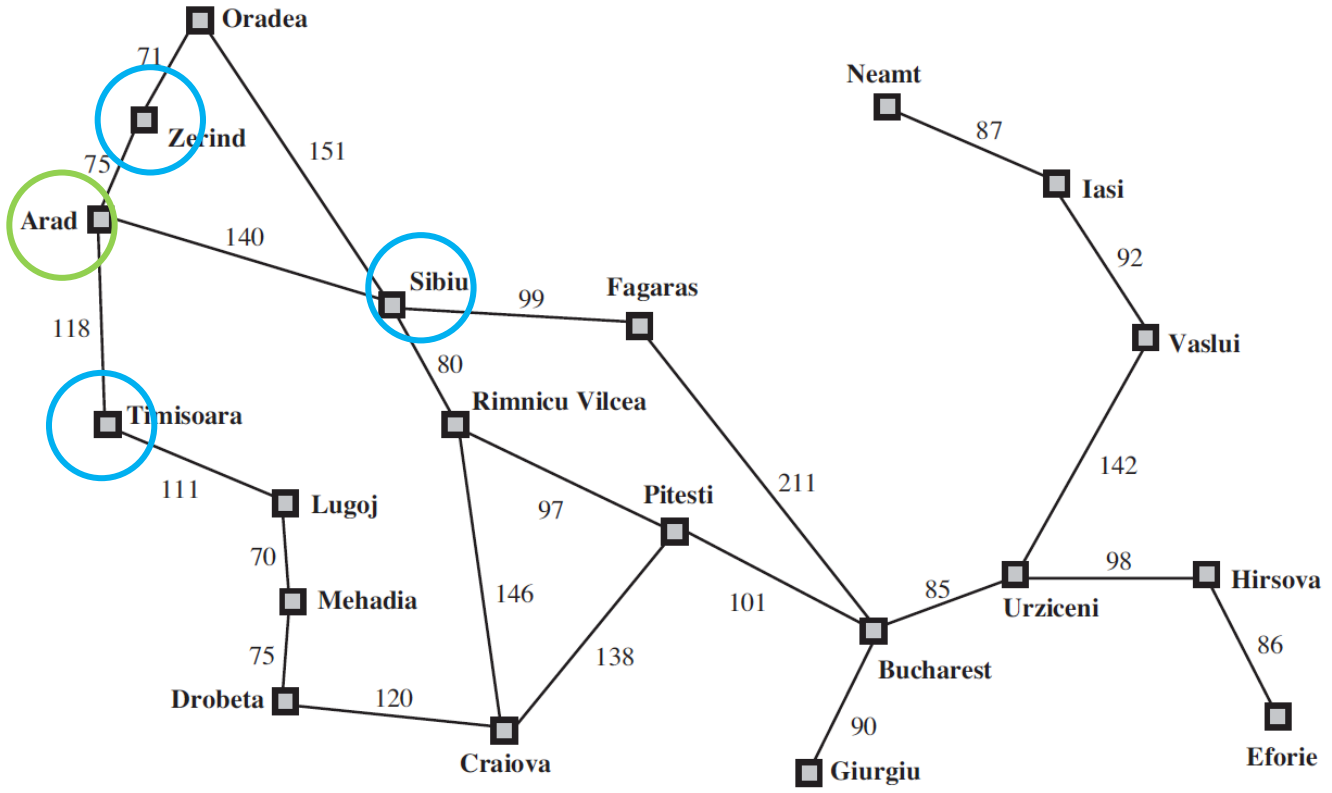
- Uninformed search techniques:
  - Breadth-First, Uniform-Cost, Depth-First, Depth-Limited, Iterative-Deepening

Example: Traveling in Romania



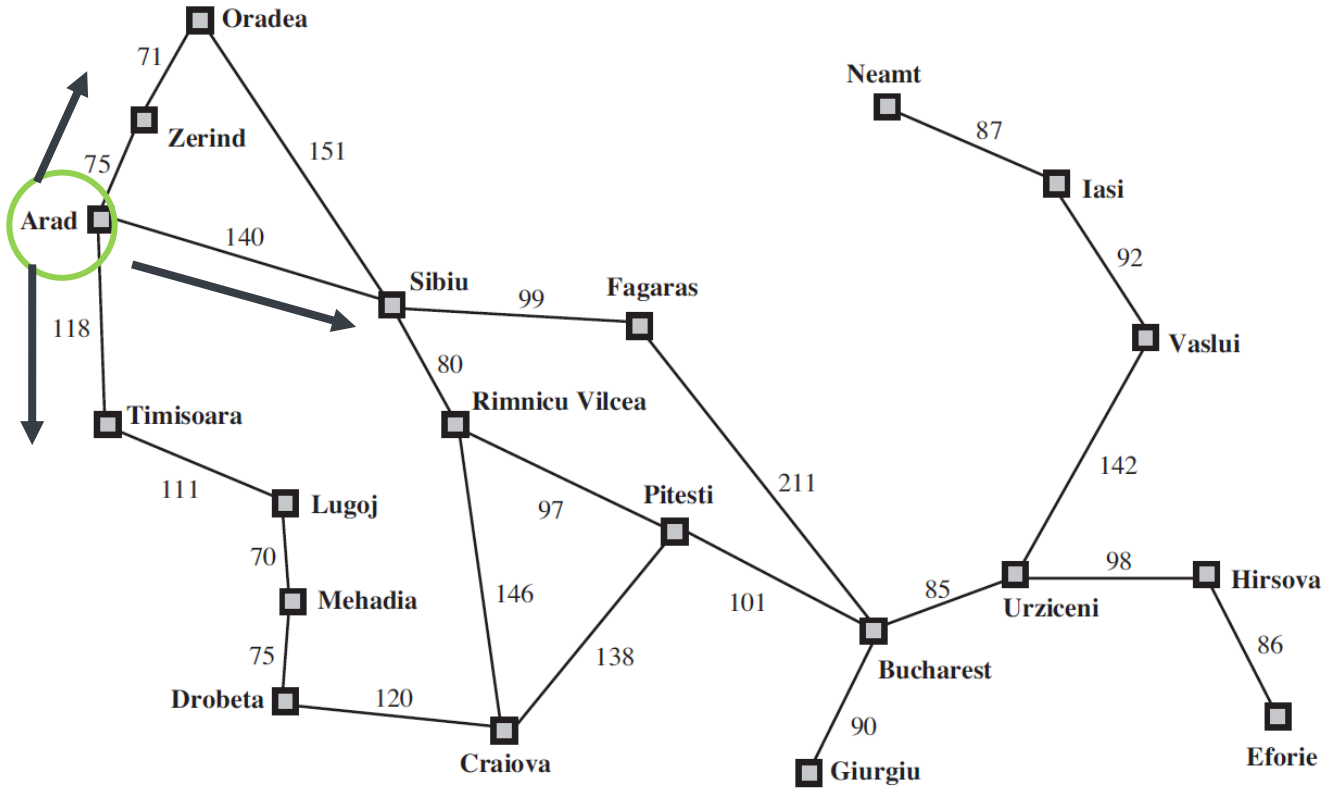
Example: Traveling in Romania

- Initial state
  - In(Arad)



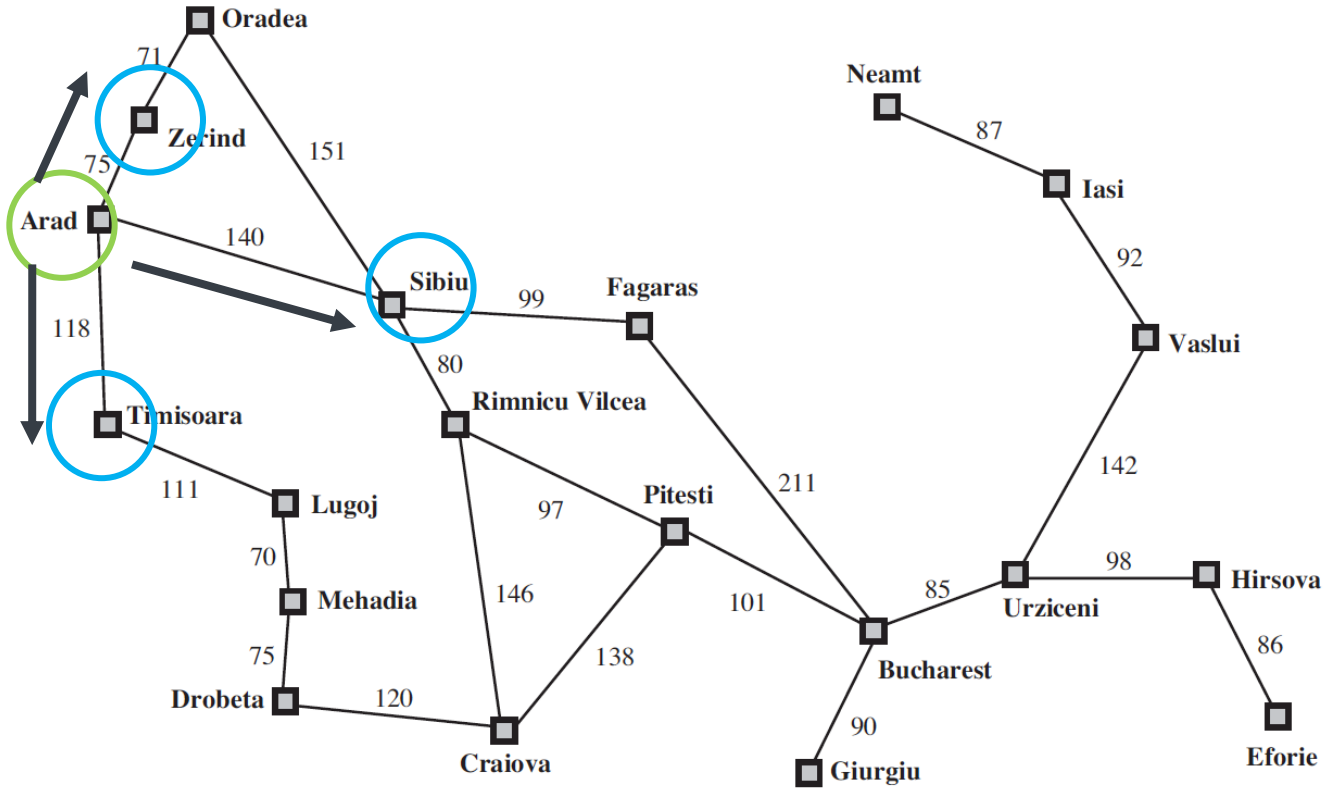
Example: Traveling in Romania

- Initial state
  - In(Arad)
- Actions
  - {Go(Sibiu), Go(Timisoara), Go(Zerind)}



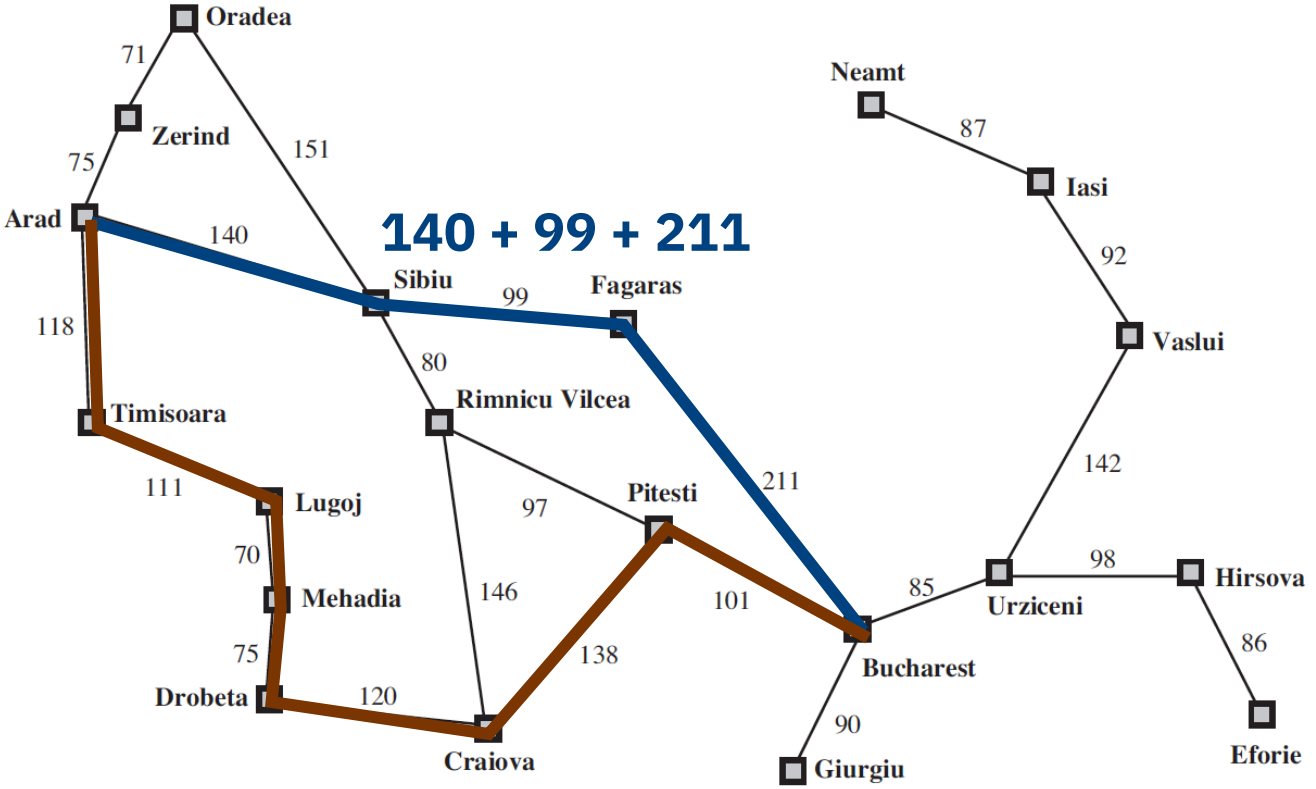
Example: Traveling in Romania

- Initial state
  - In(Arad)
- Actions
  - {Go(Sibiu), Go(Timisoara), Go(Zerind)}
- Transition model
  - RESULT(In(Arad),Go(Zerind)) = In(Zerind)
- Goal test
  - {In(Bucharest)}



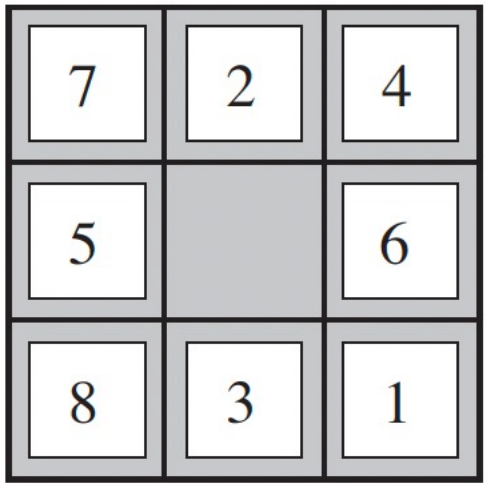
Example: Traveling in Romania

- Initial state
  - In(Arad)
- Actions
  - {Go(Sibiu), Go(Timisoara), Go(Zerind)}
- Transition model
  - RESULT(In(Arad),Go(Zerind)) = In(Zerind)
- Goal test
  - {In(Bucharest)}
- Path cost
  - The sum of route distances

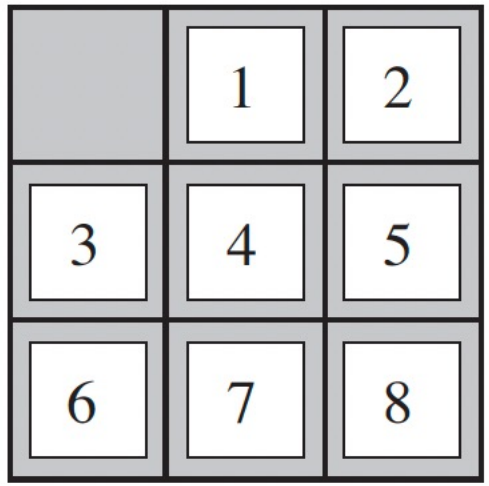


Example: 8-puzzle

- Initial state
  - Initial location of each of the eight tiles and the blank
- Actions
  - Movements of the blank space (left, right, up, down)
- Transition model
  - The resulting state
- Goal test
  - Whether the state matches the goal configuration
- Path cost
  - Each step costs 1

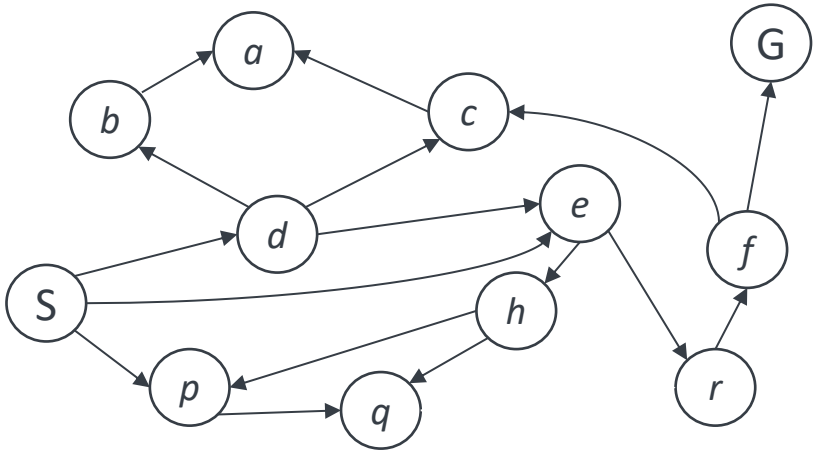


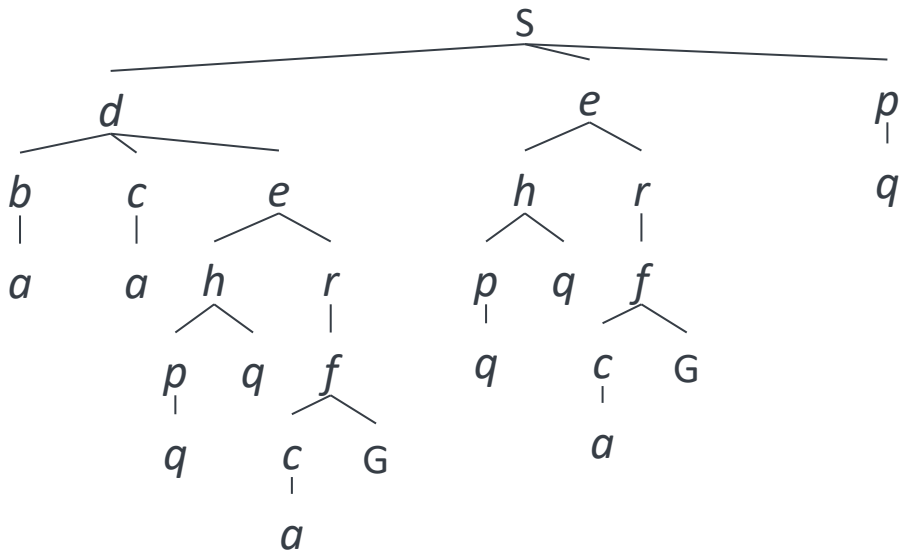
Start State



Goal State

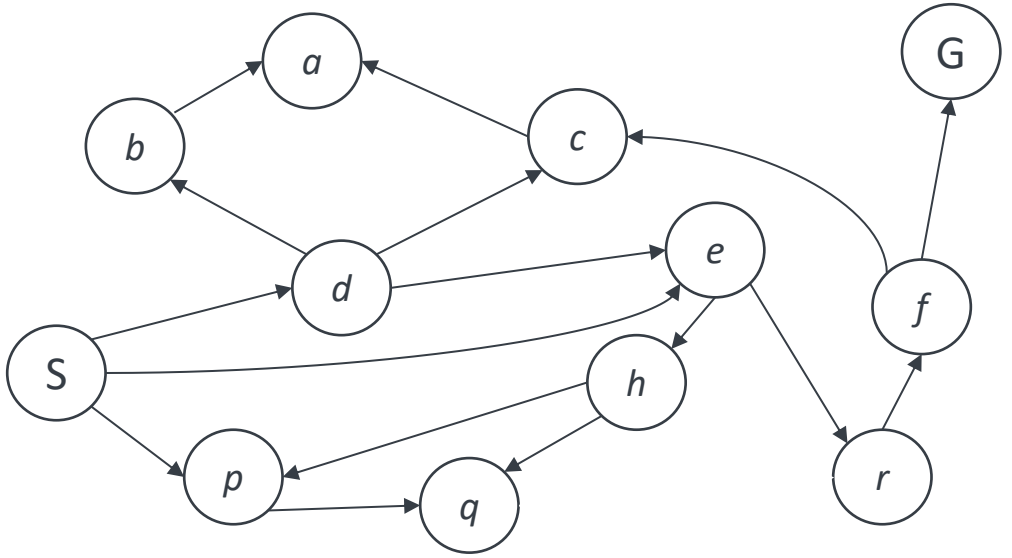
Representing search





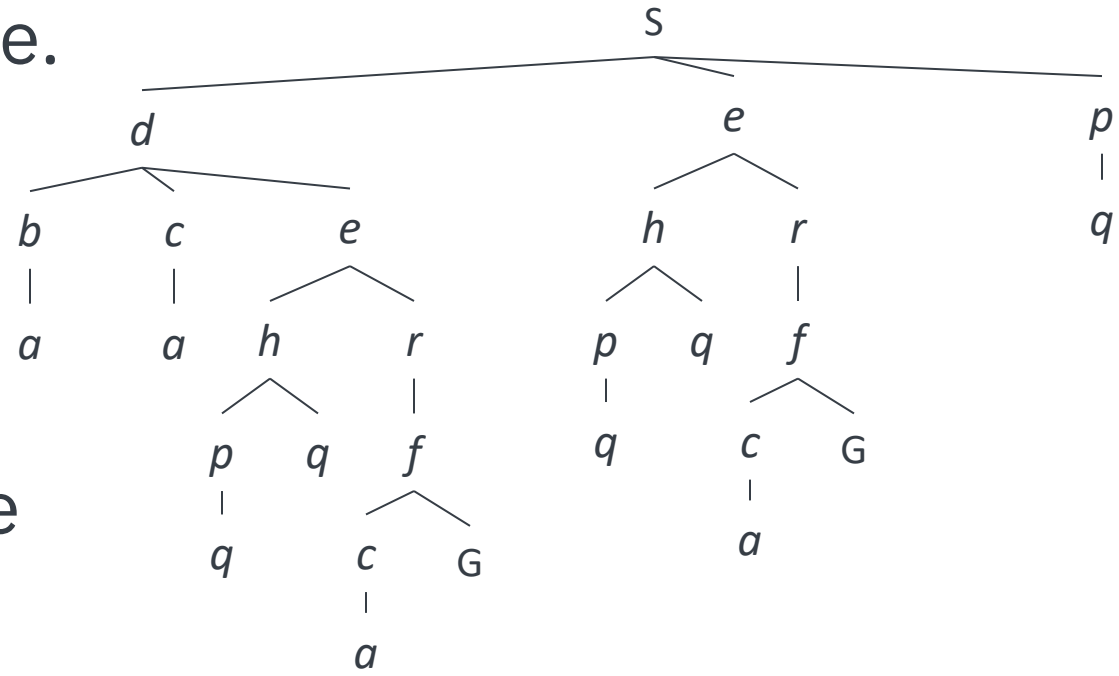
State space graph

- State space graph
  - Mathematical representation of a search problem.
  - Vertices are states (*i.e.*, world configurations).
  - Edges correspond to successors.
  - The goal test is a set of goal nodes.
- In a search graph, each state occurs only once!
- We can rarely build this full graph in memory (it's too big), but it's a useful idea.



Search tree

- A search tree
  - Start state is the root of the tree.
  - Children are successors.
  - A plan is a path in the tree.
  - For the most problems , we do not actually generate the entire tree.



Search terminology

- **Expansion** of nodes
  - As states are explored, the corresponding nodes are expanded by legal operators (*i.e.*, actions).

- The fringe (frontier) is the set of nodes that are newly **generated** and not yet visited.
  - Newly generated nodes are added to the fringe.

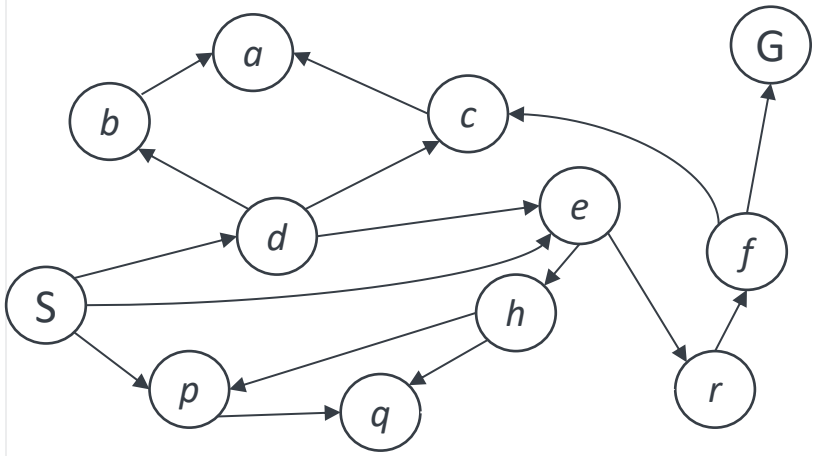
- Search strategy
  - determines the selection of the next node to be expanded.
  - can be achieved by ordering the nodes in the fringe.

Expanding nodes

- Apply legal operators to the state contained in the node.
- Generate nodes for corresponding successor states.
- Try to expand as few tree nodes as possible.

State Space Graph and Search Tree

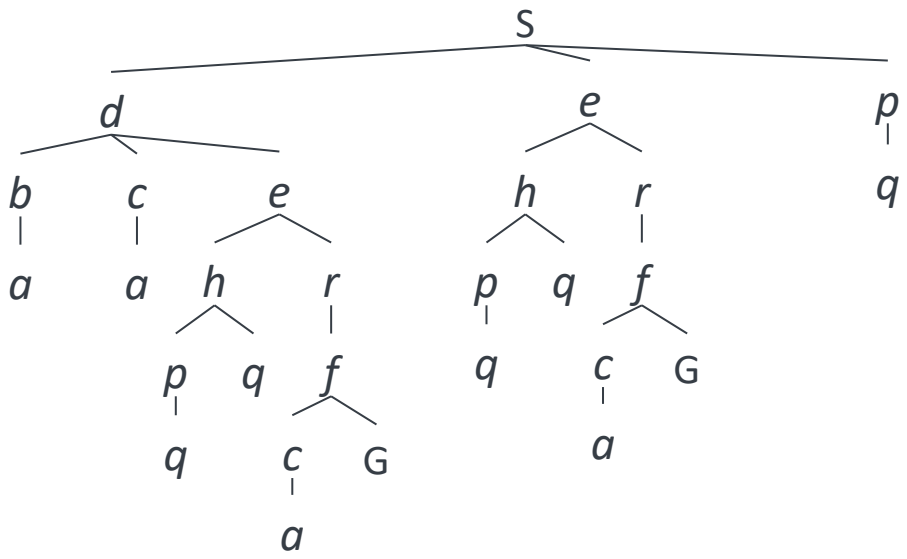
State Space Graph



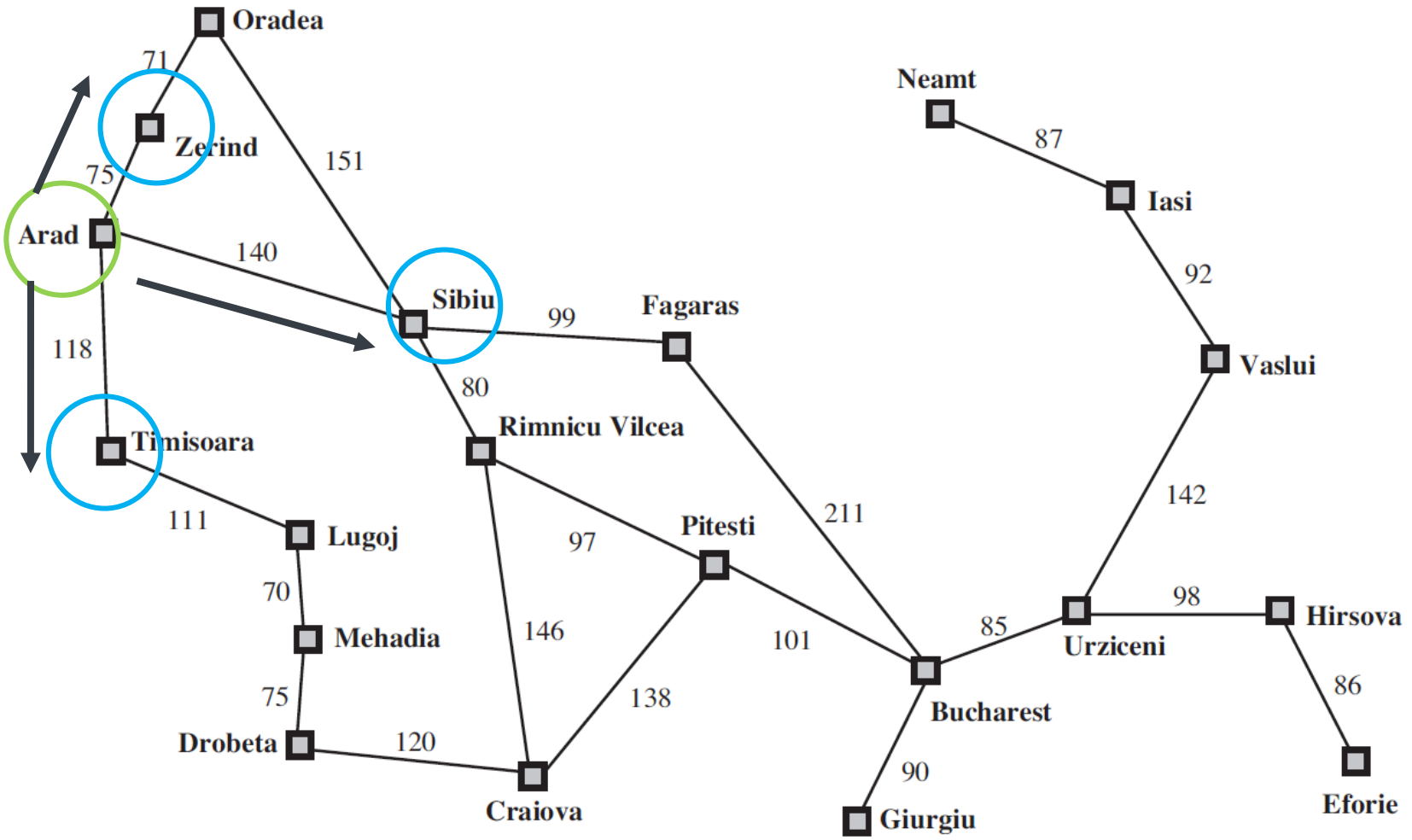
Each NODE in the search tree is an entire PATH in the state space graph.

We construct both on demand – and we construct as little as possible.

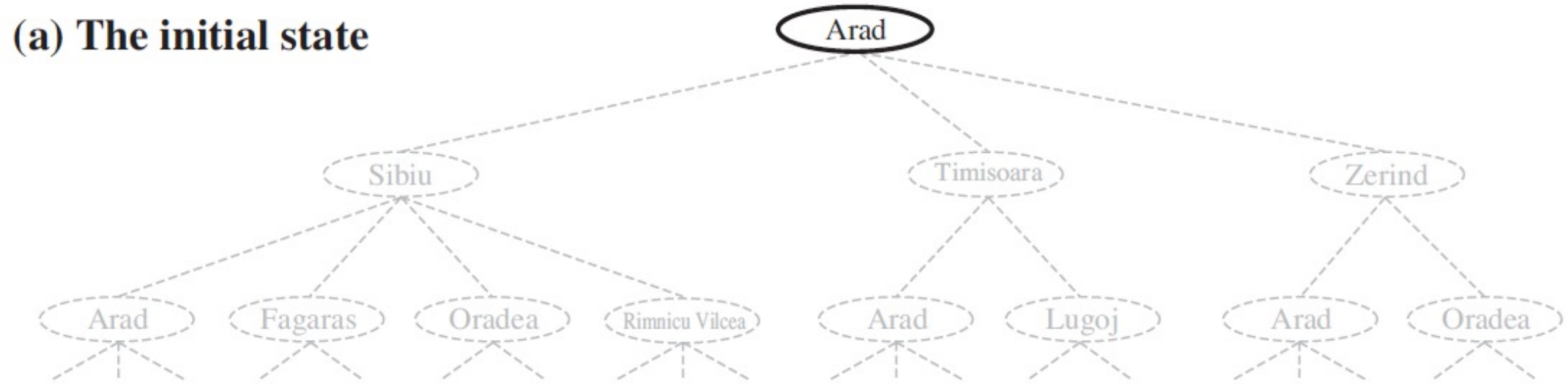
Search Tree



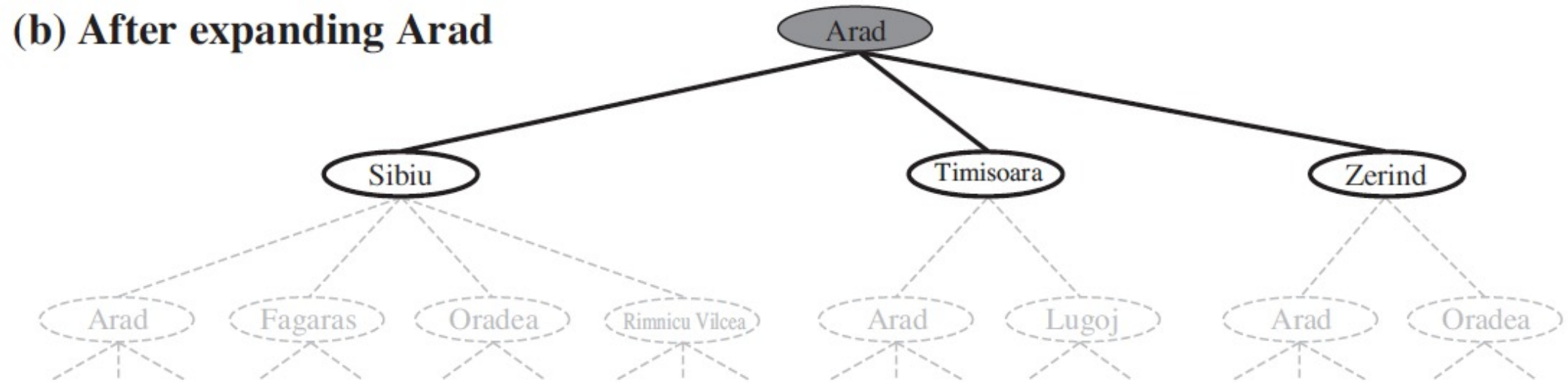
Search example: Romania



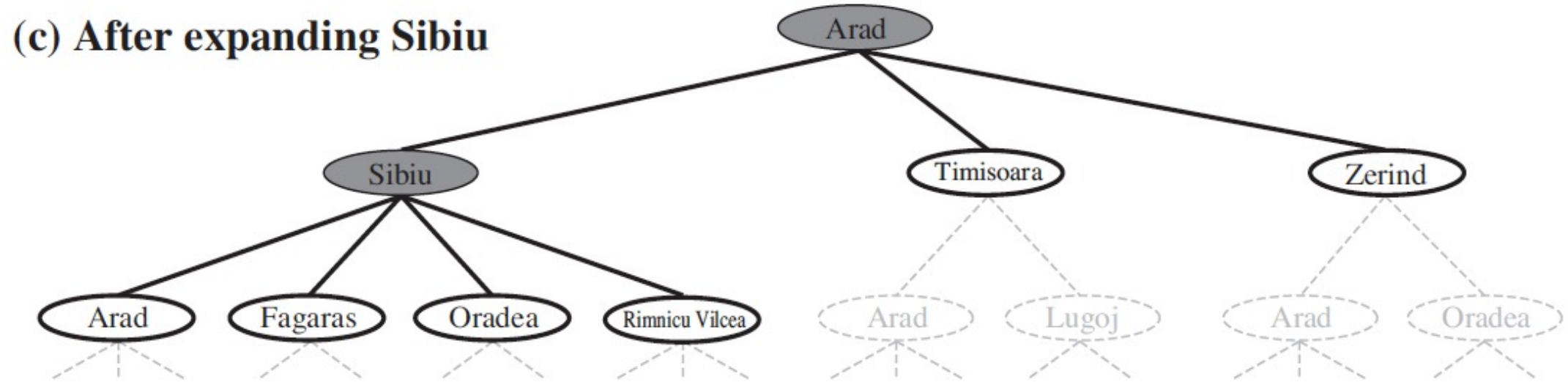
Search tree example: Romania



Search tree example: Romania



Search tree example: Romania



General tree search

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 (according to strategy)
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

Main question: which fringe nodes to explore?

This works in general but it can be more efficient when the actions have uniform costs.

General tree search

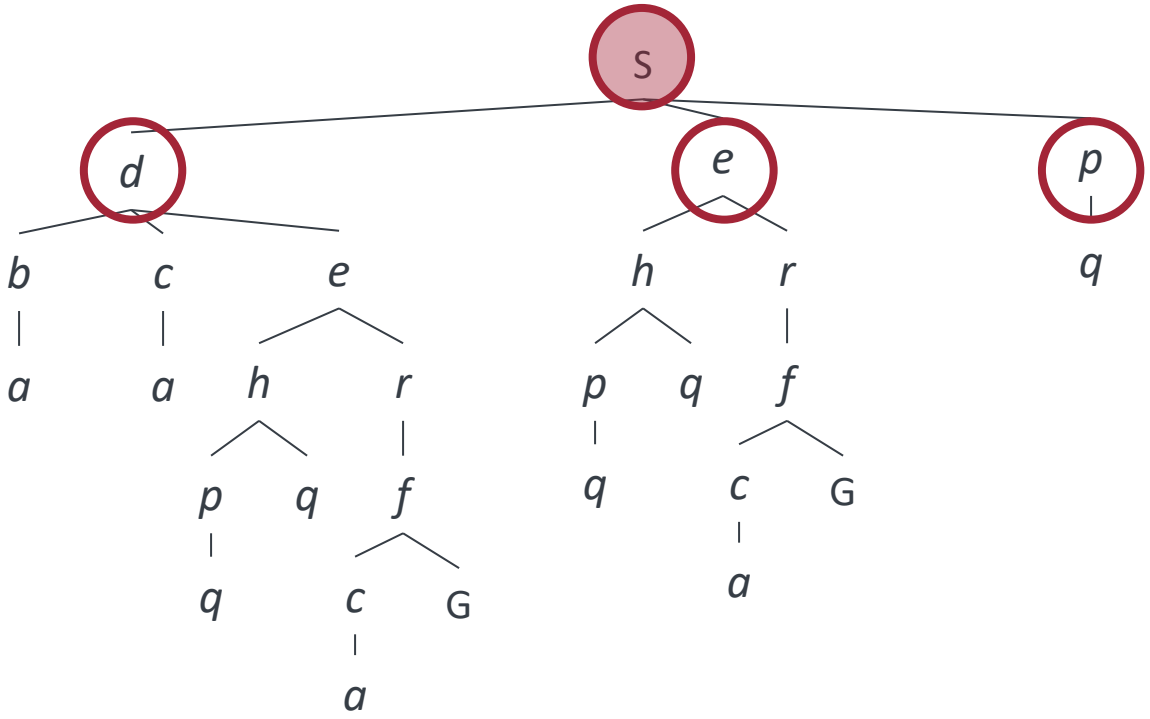
```
function TREE-SEARCH(problem) returns a solution, or failure
    initialize the frontier using the initial state of problem
    loop do
        if the frontier is empty then return failure
        choose a leaf node and remove it from the frontier   (according to strategy)
        for succ in successors(node):
                if succ is a goal state then return the corresponding solution
                frontier.insert(<node, succ>)
```

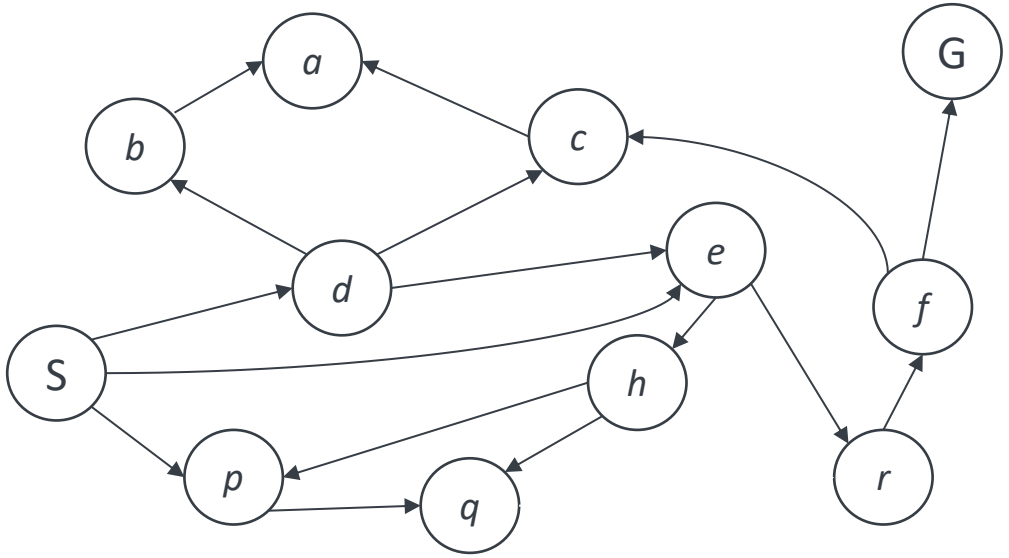
Main question: which fringe nodes to explore?

If the actions have uniform costs, the search can stop earlier.

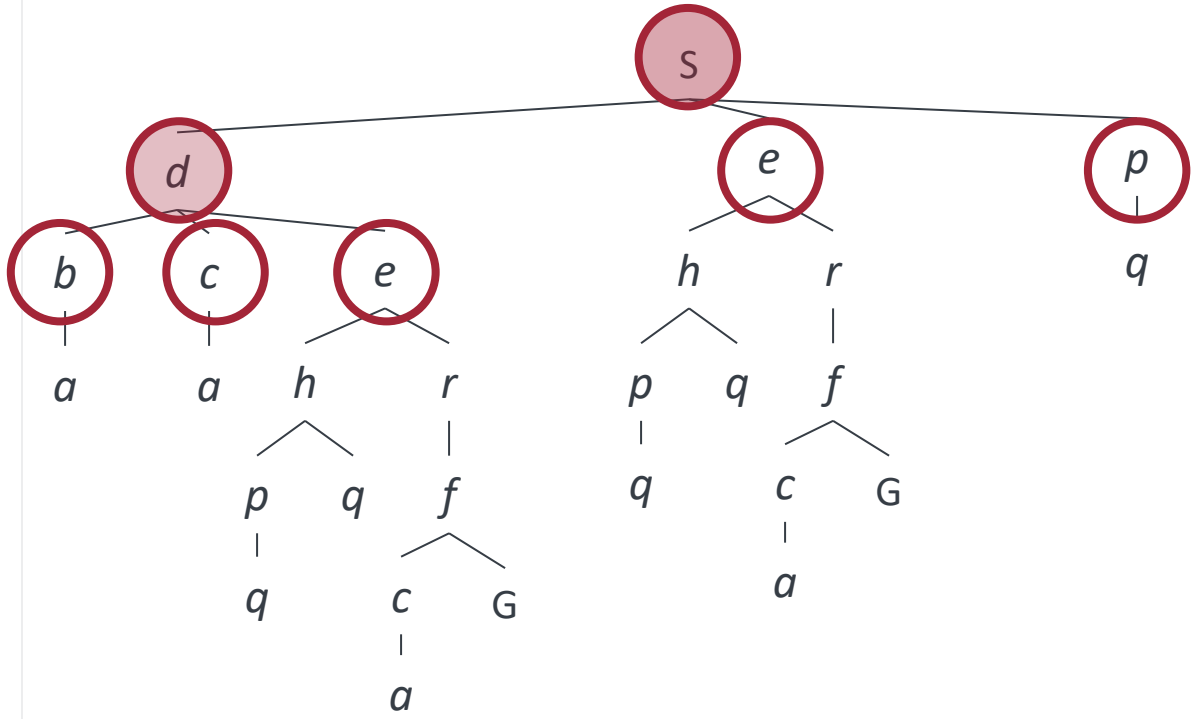
Depth-First Search (DFS)

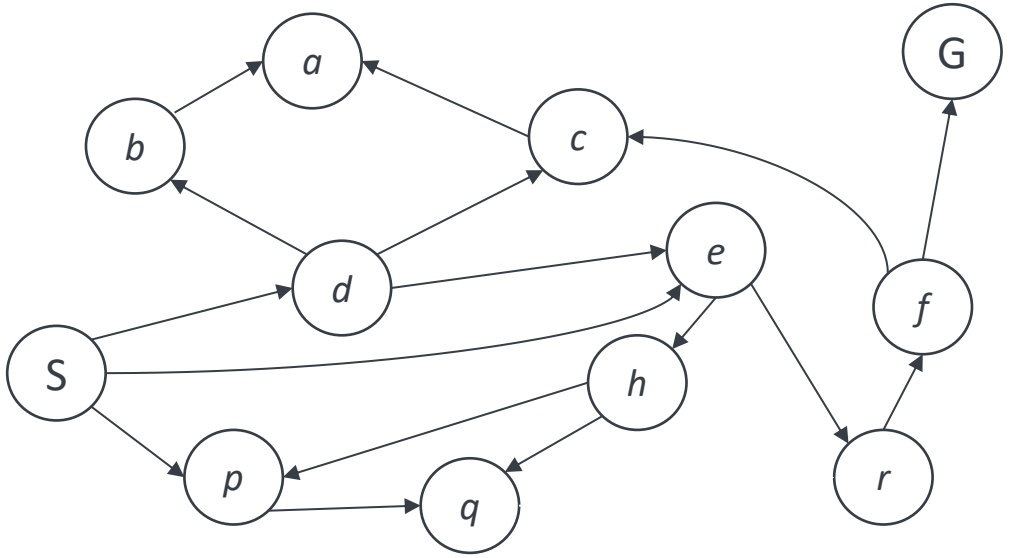
Depth-First Search (DFS)



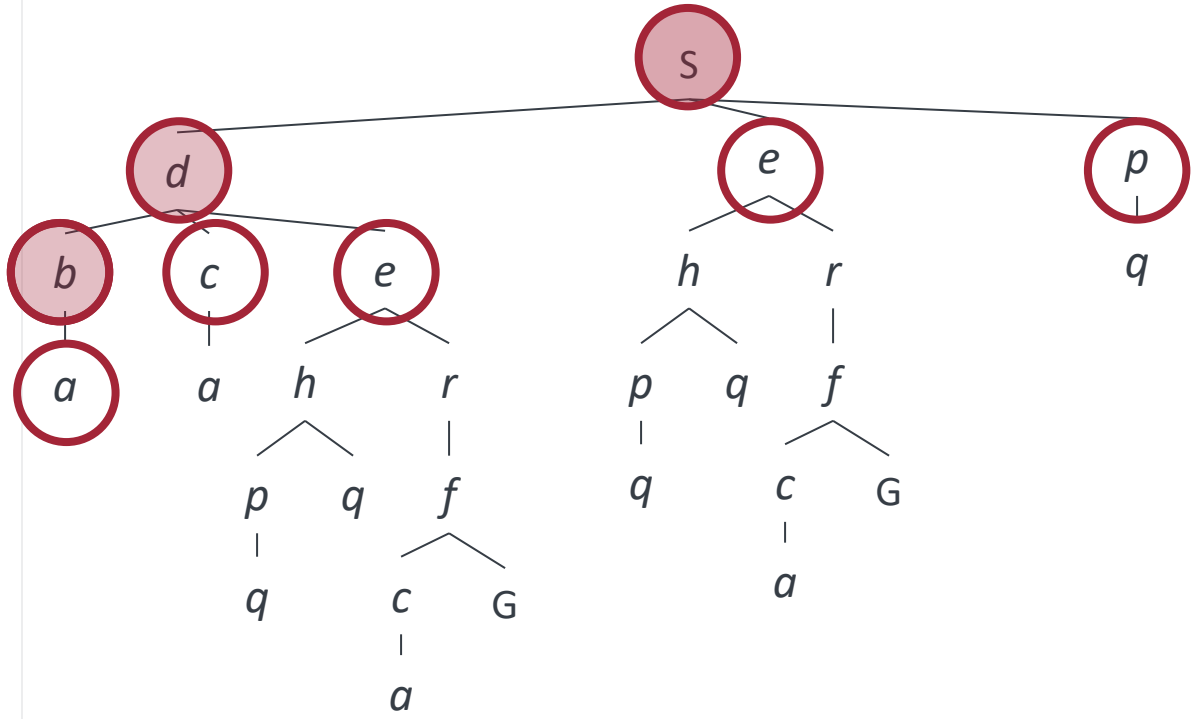


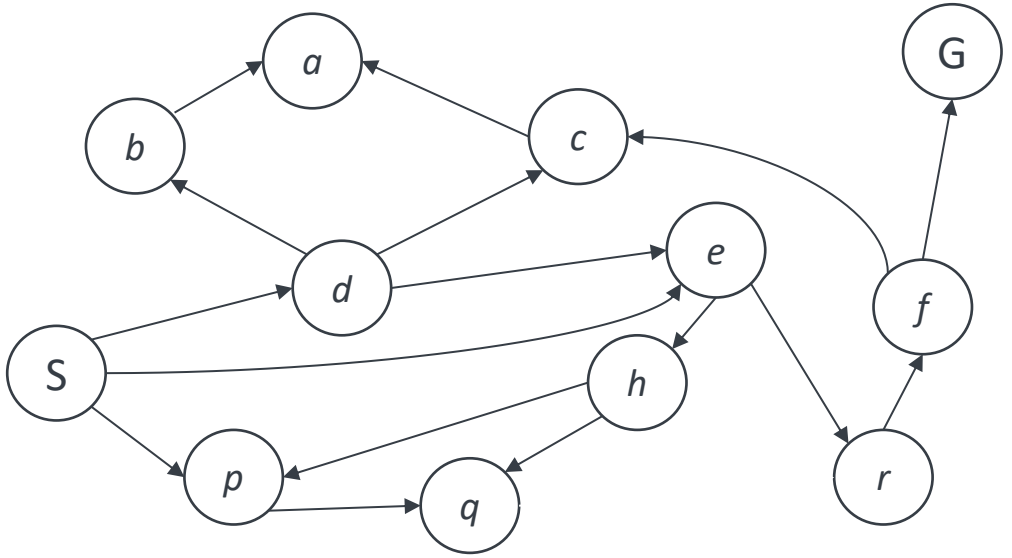
Depth-First Search (DFS)



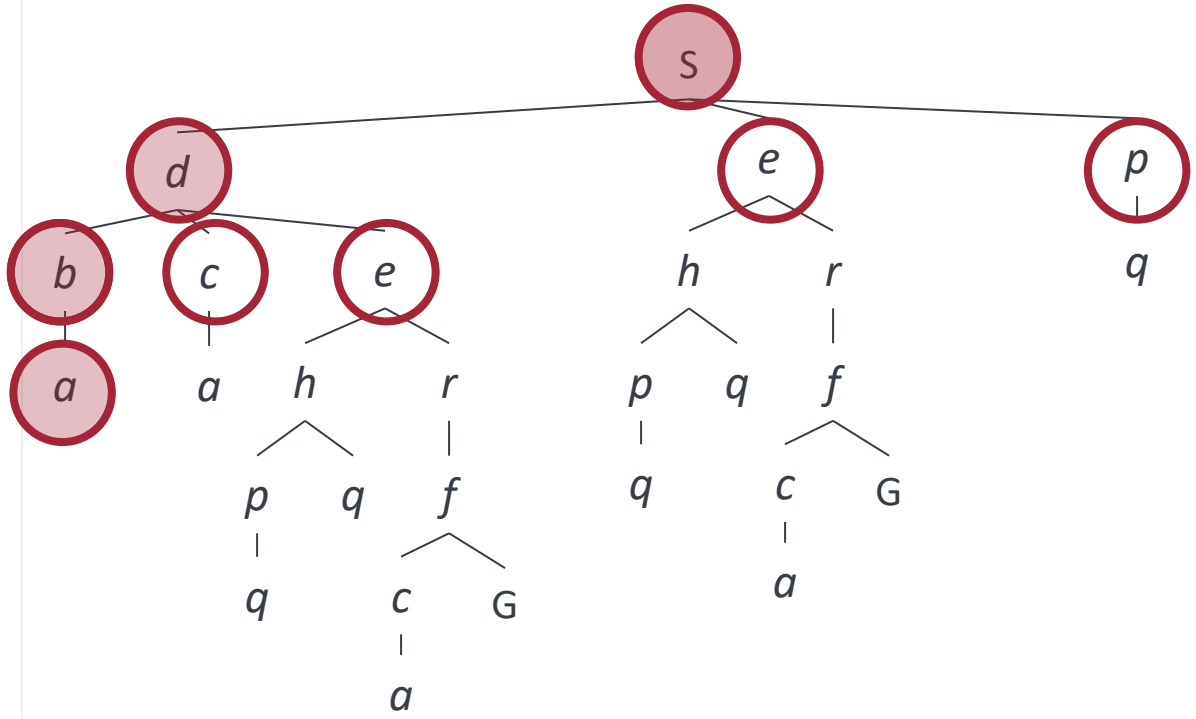


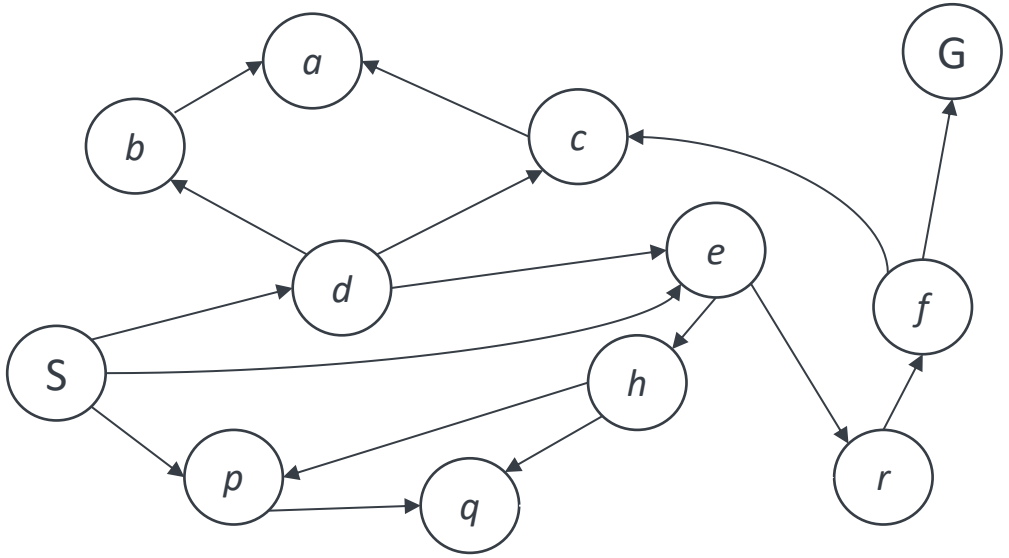
Depth-First Search (DFS)



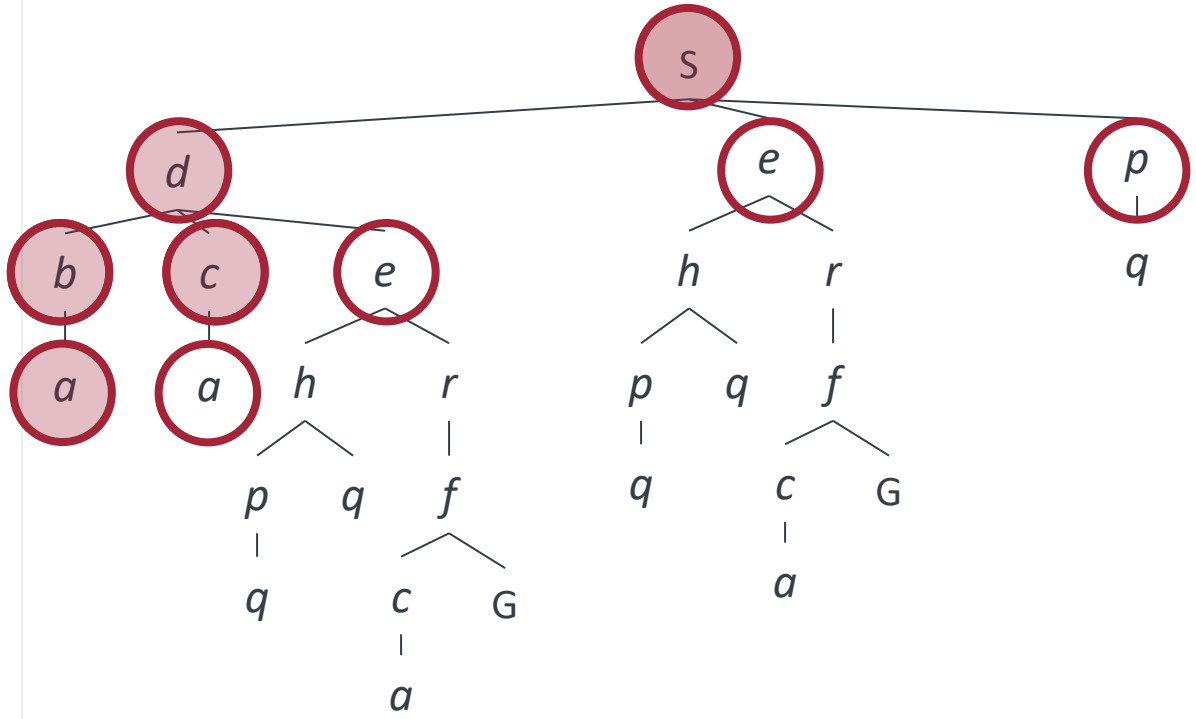


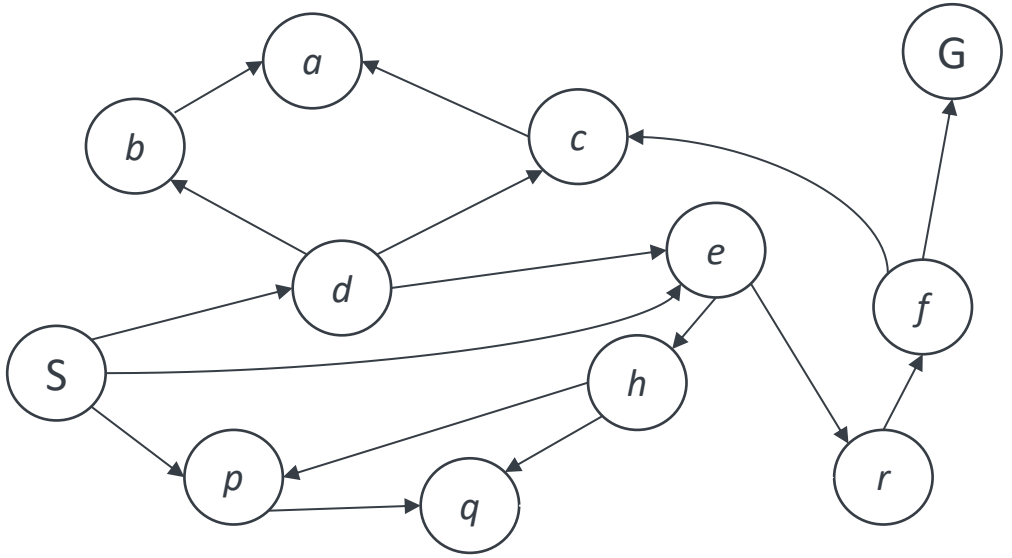
Depth-First Search (DFS)



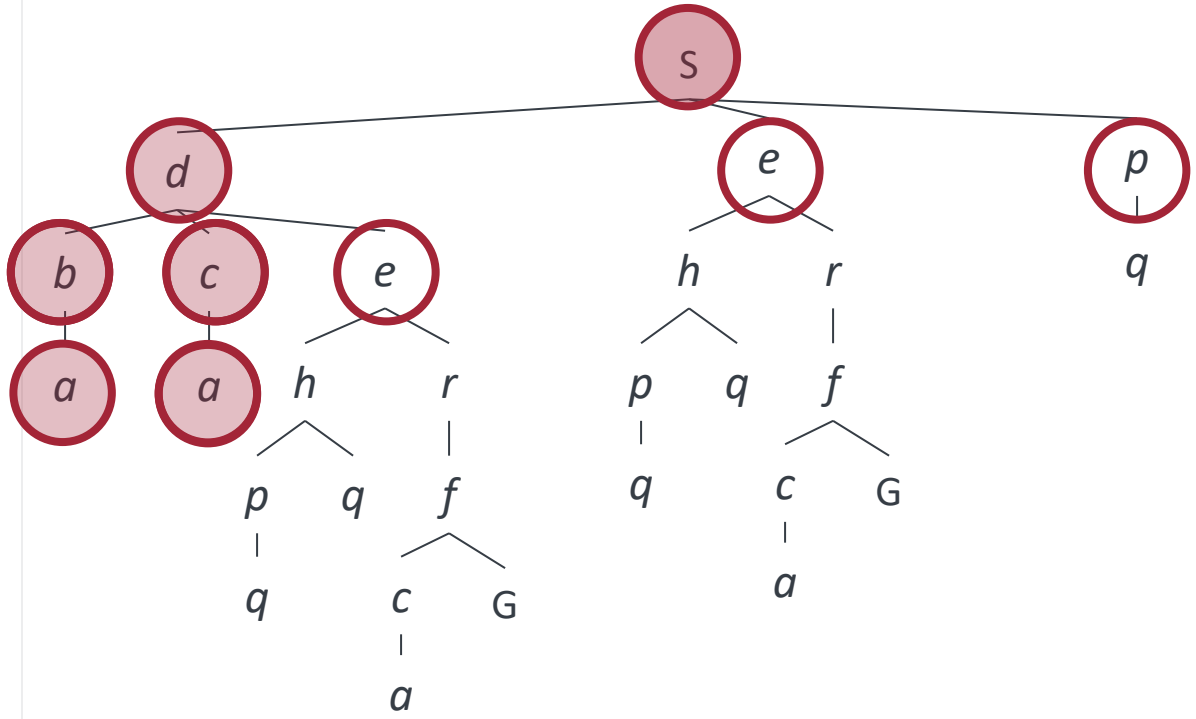


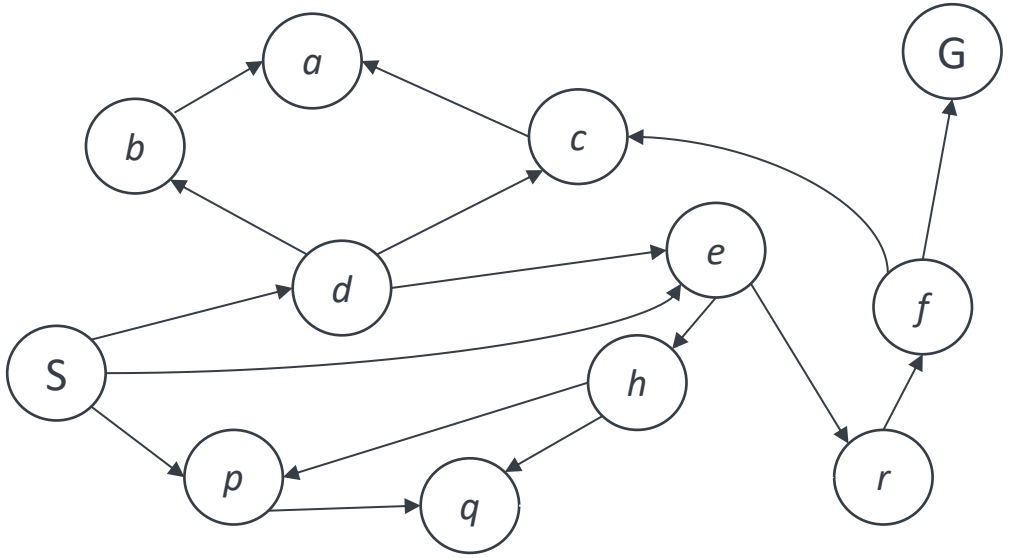
Depth-First Search (DFS)



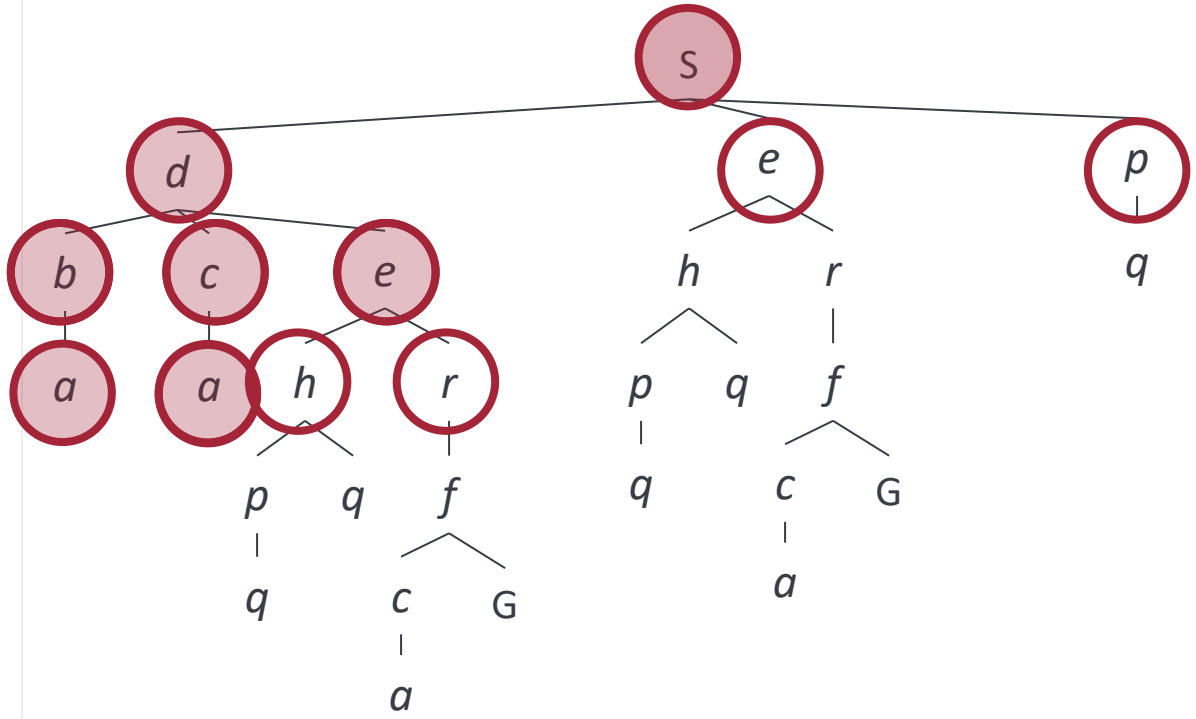


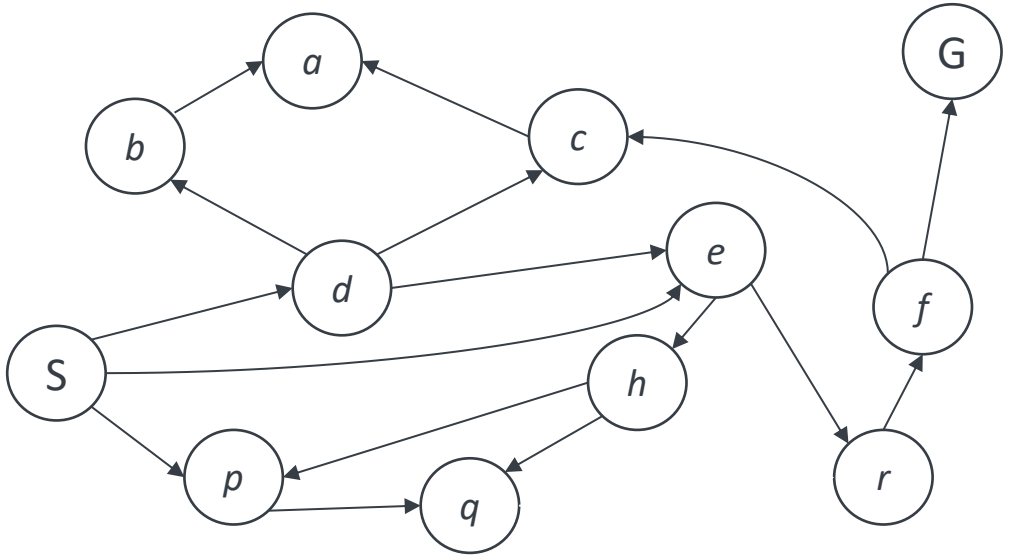
Depth-First Search (DFS)



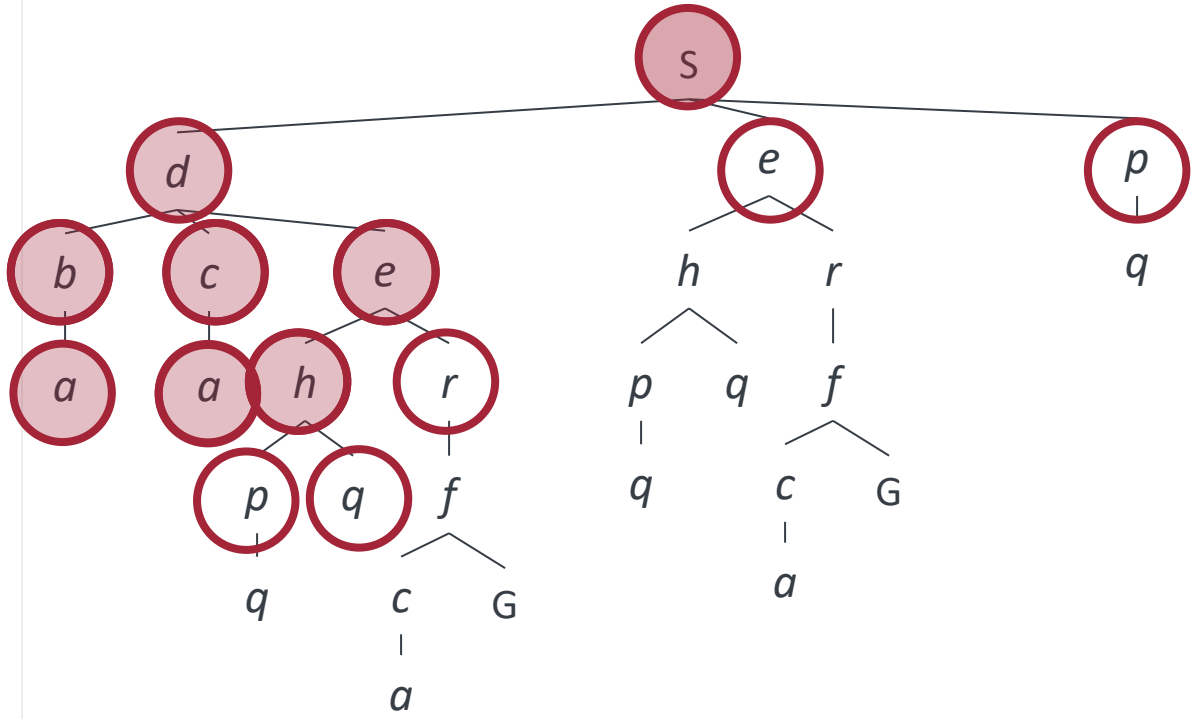


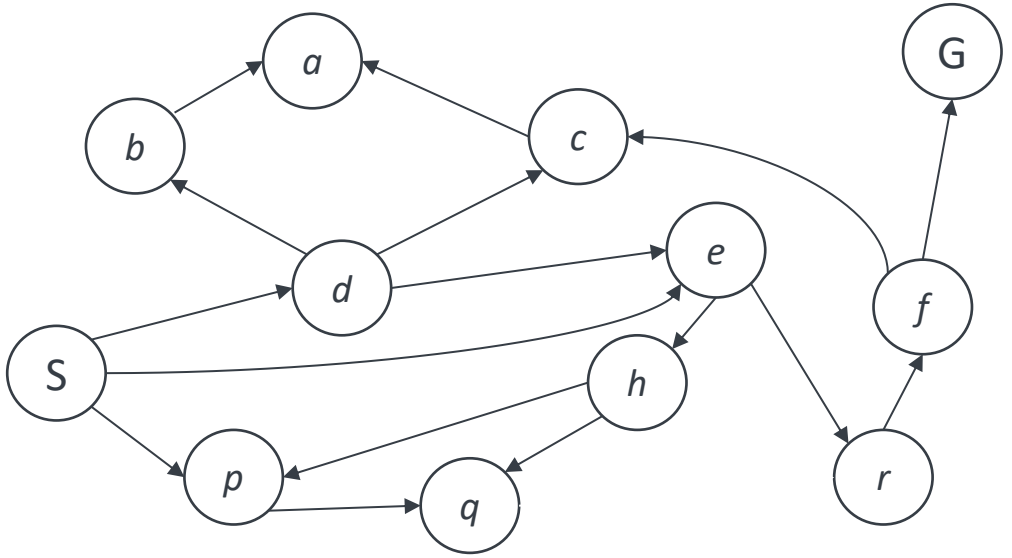
Depth-First Search (DFS)



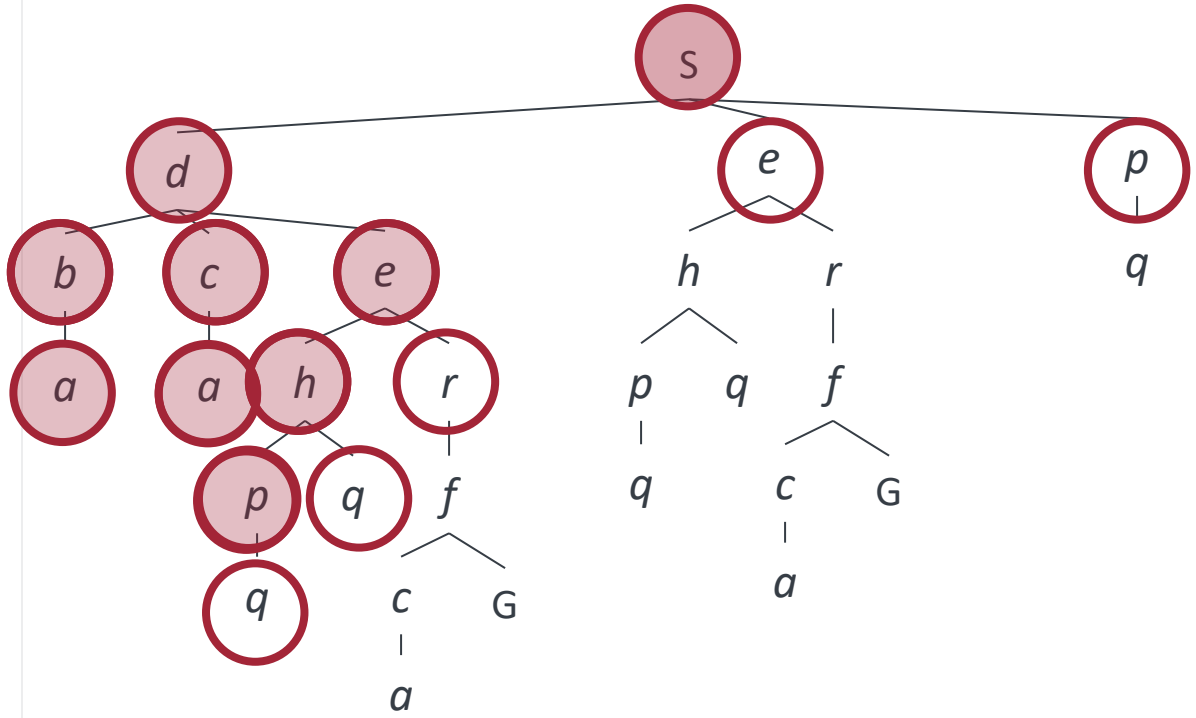


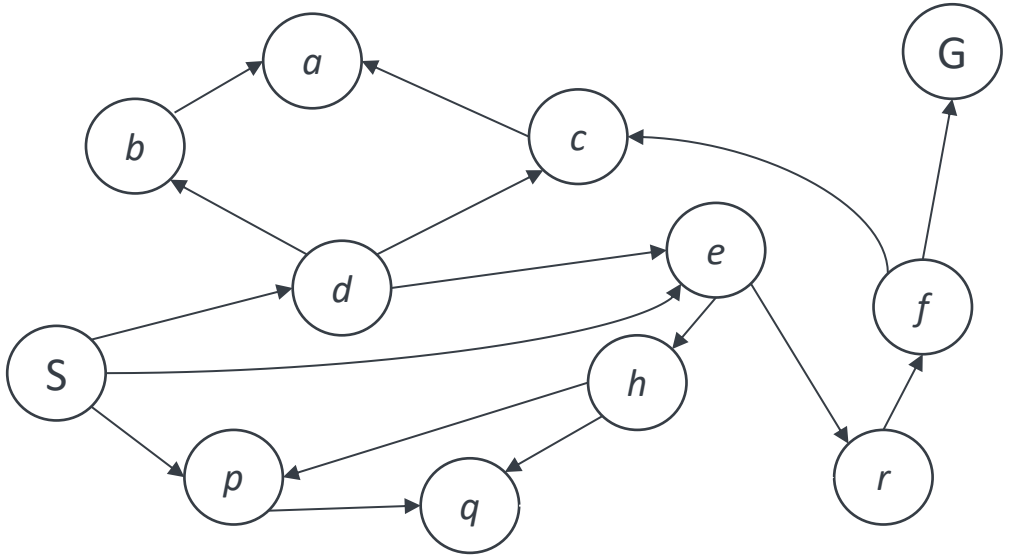
Depth-First Search (DFS)



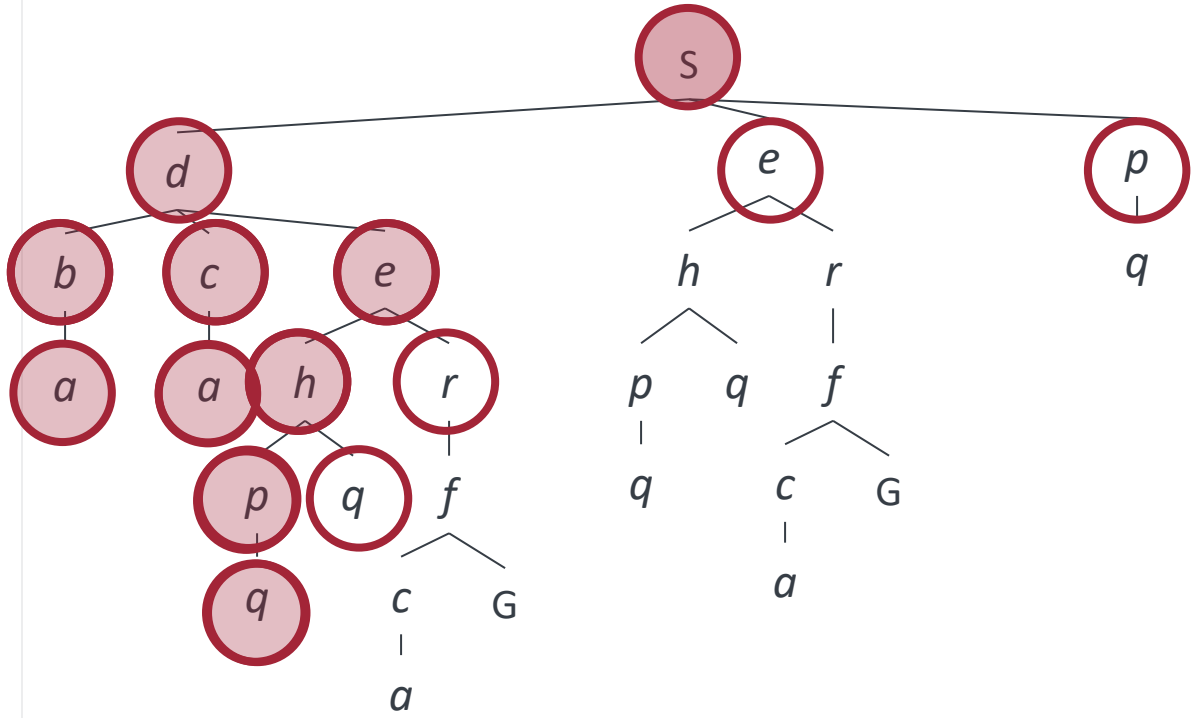


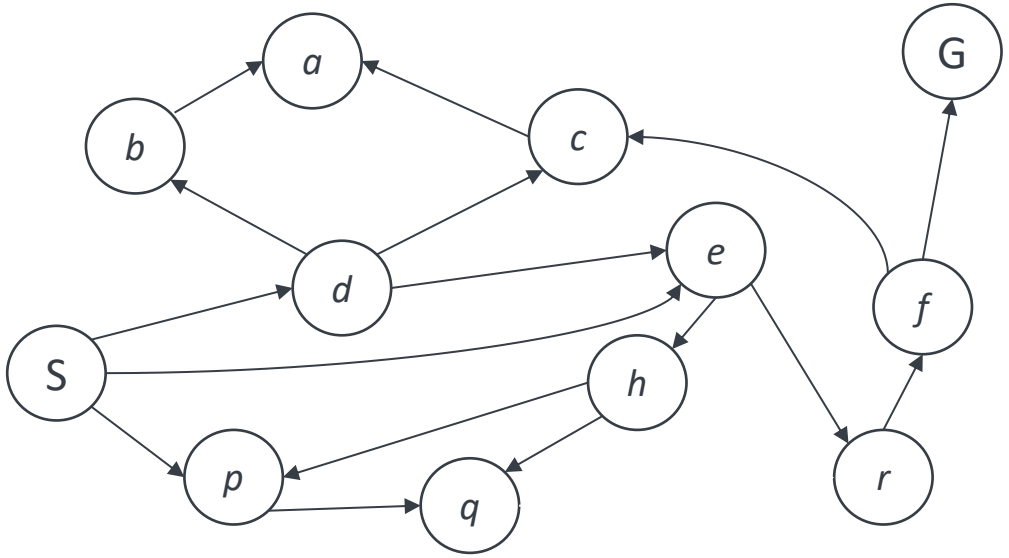
Depth-First Search (DFS)



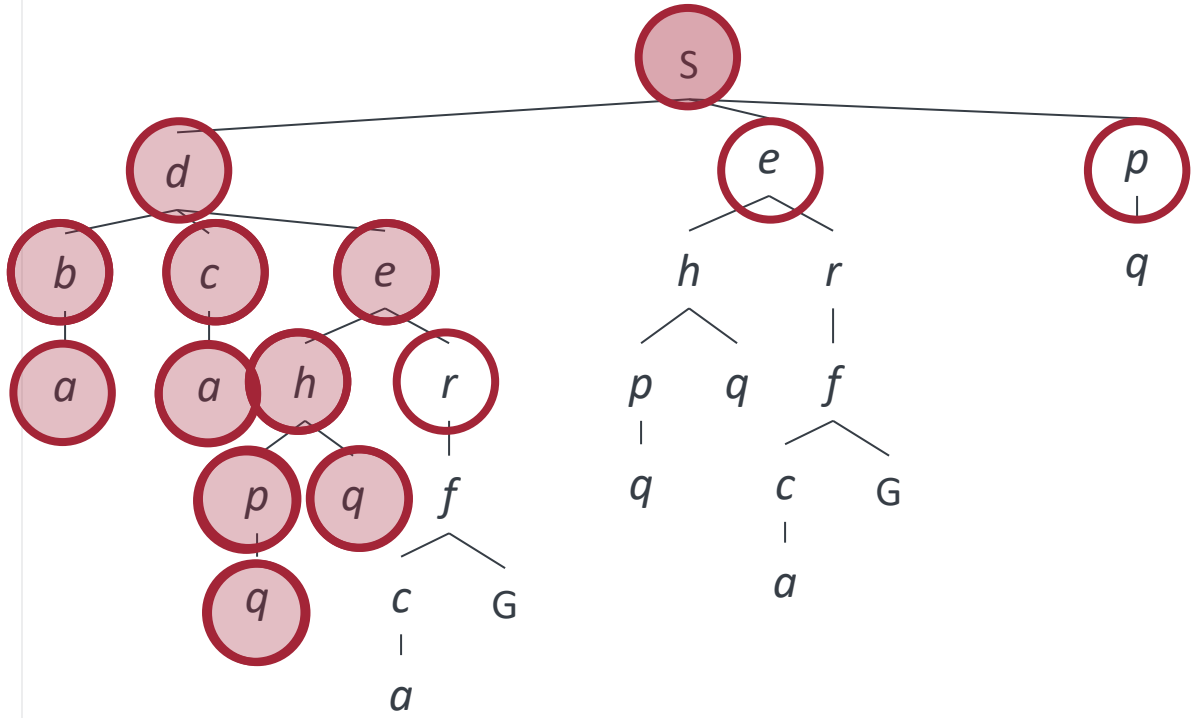


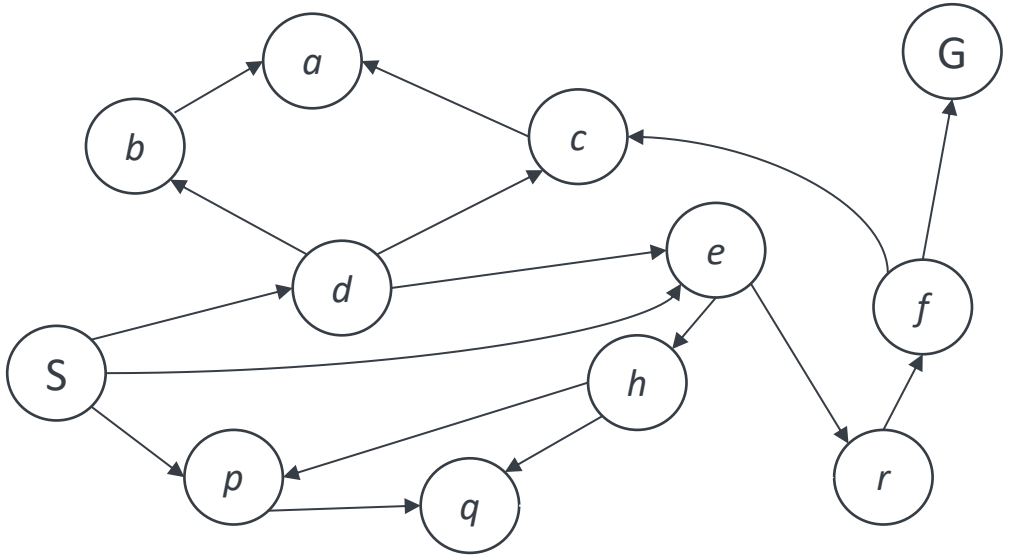
Depth-First Search (DFS)



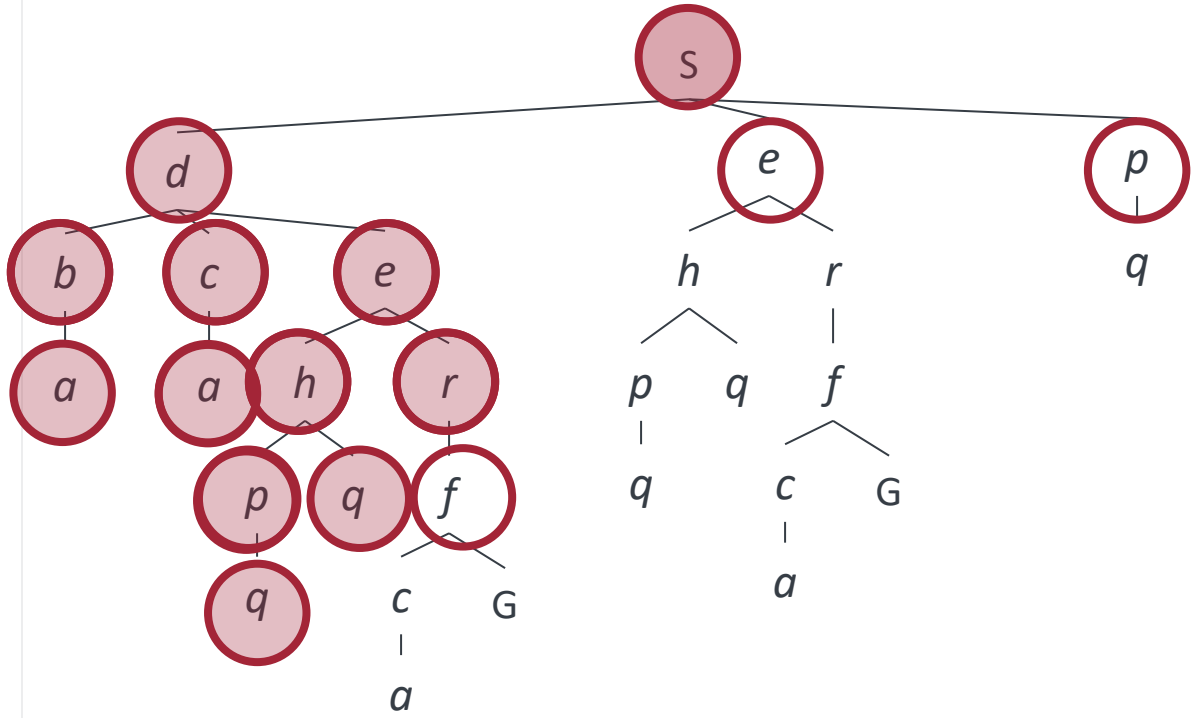


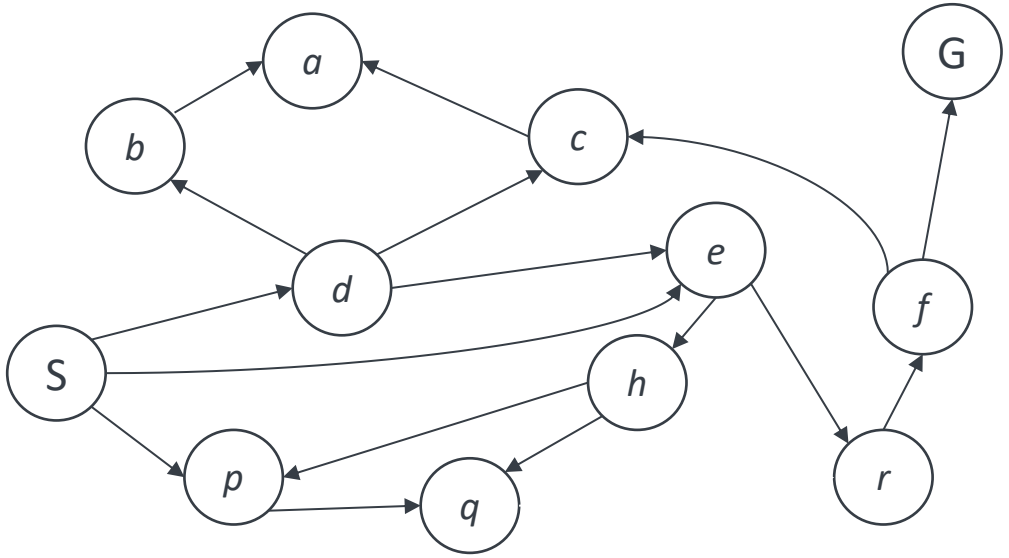
Depth-First Search (DFS)



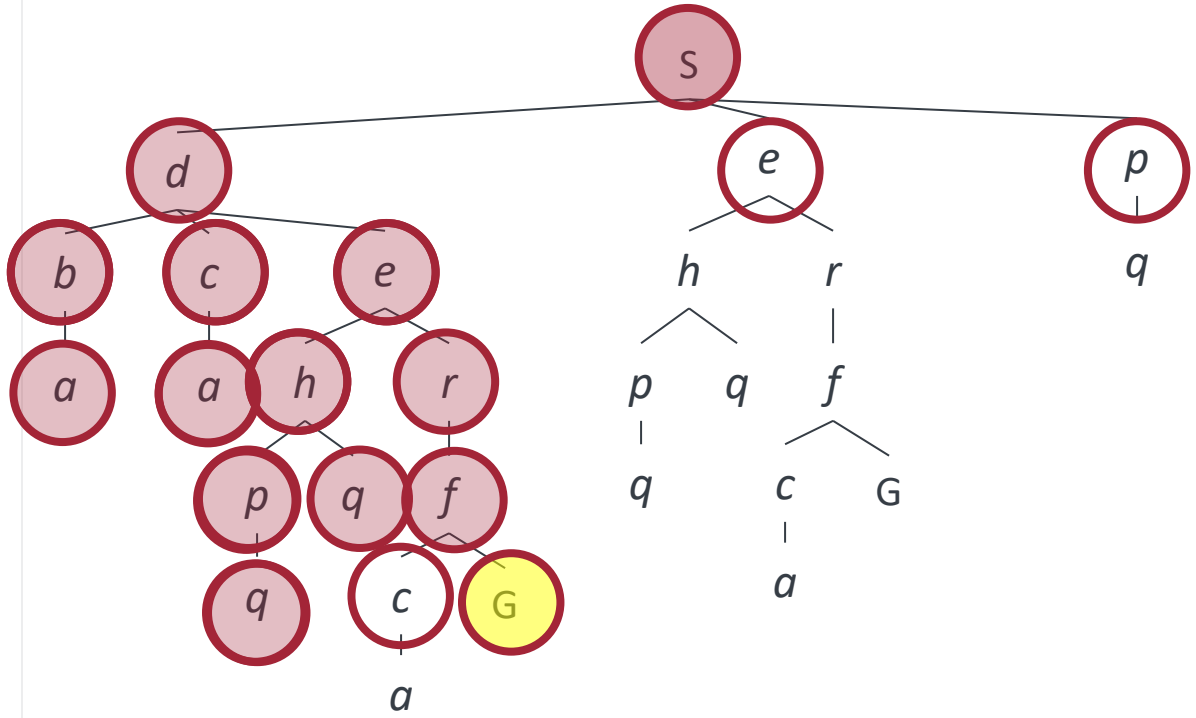


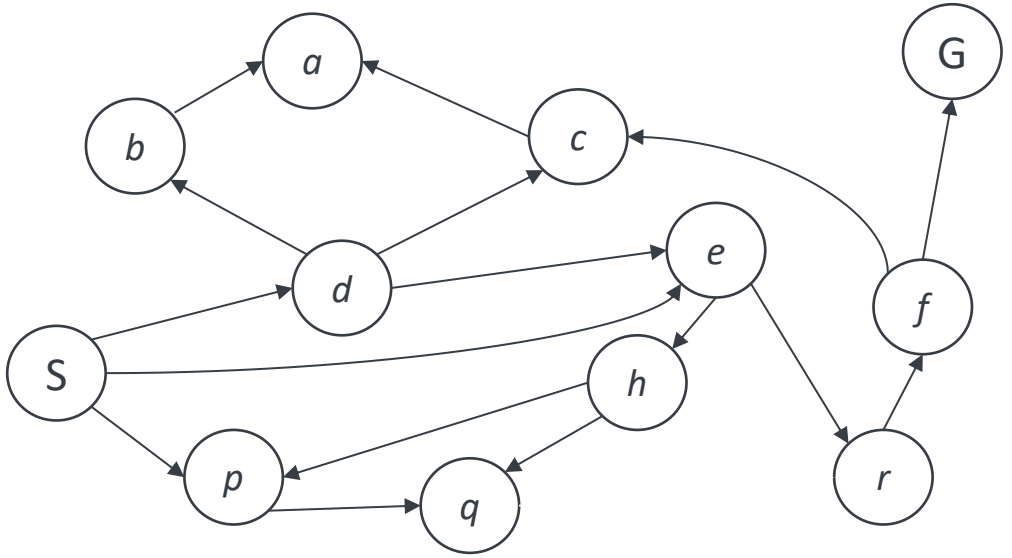
Depth-First Search (DFS)





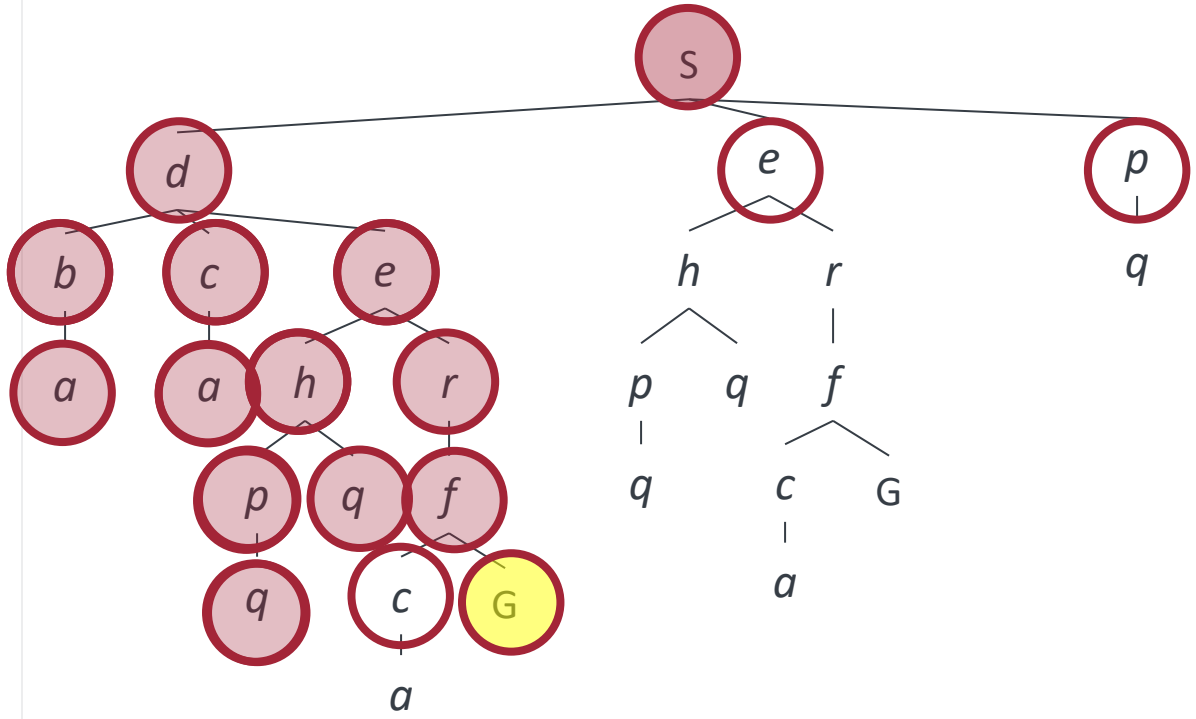
Depth-First Search (DFS)

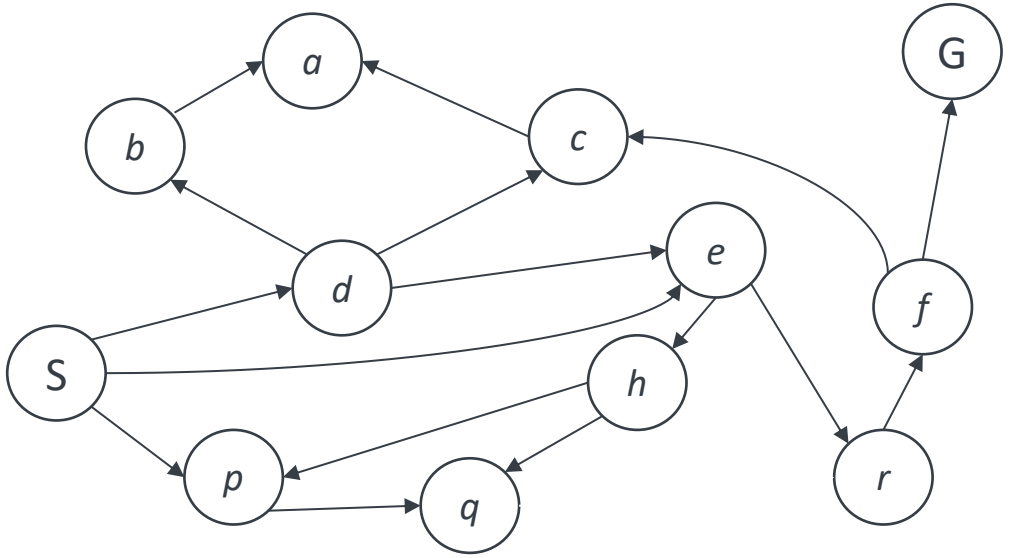




Note that the search ends when the goal node is **generated** (not expanded).

Depth-First Search (DFS)





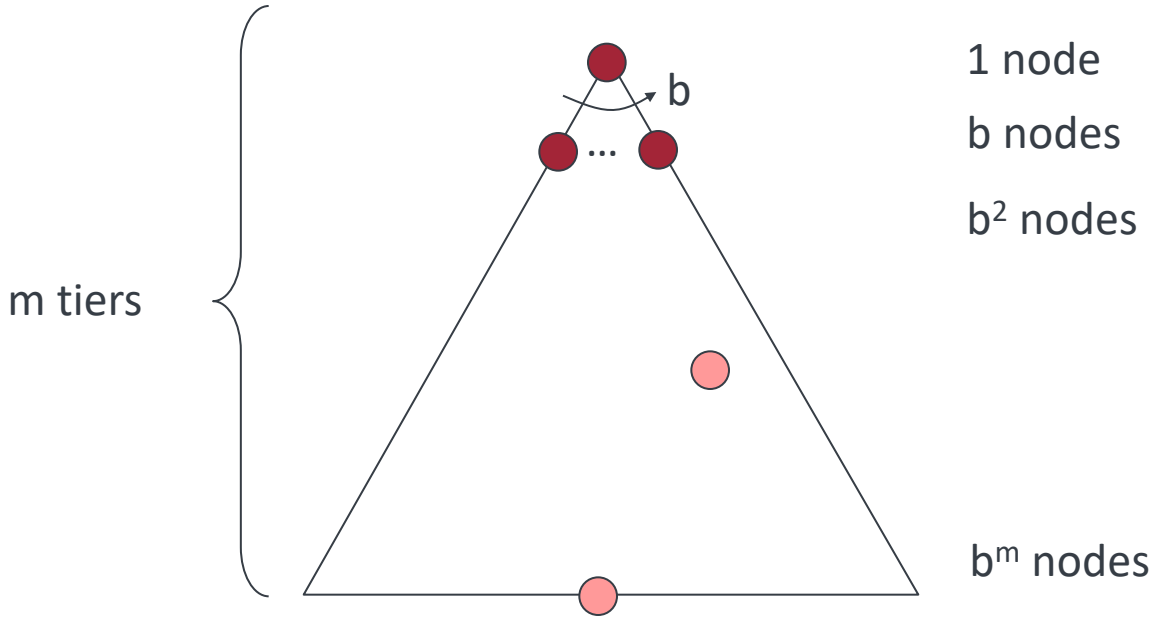
Number of expanded nodes: 13

Search algorithm properties

- Complete: Guaranteed to find a solution if one exists?
- Optimal: Guaranteed to find the least cost path?
- Time complexity?
- Space complexity?

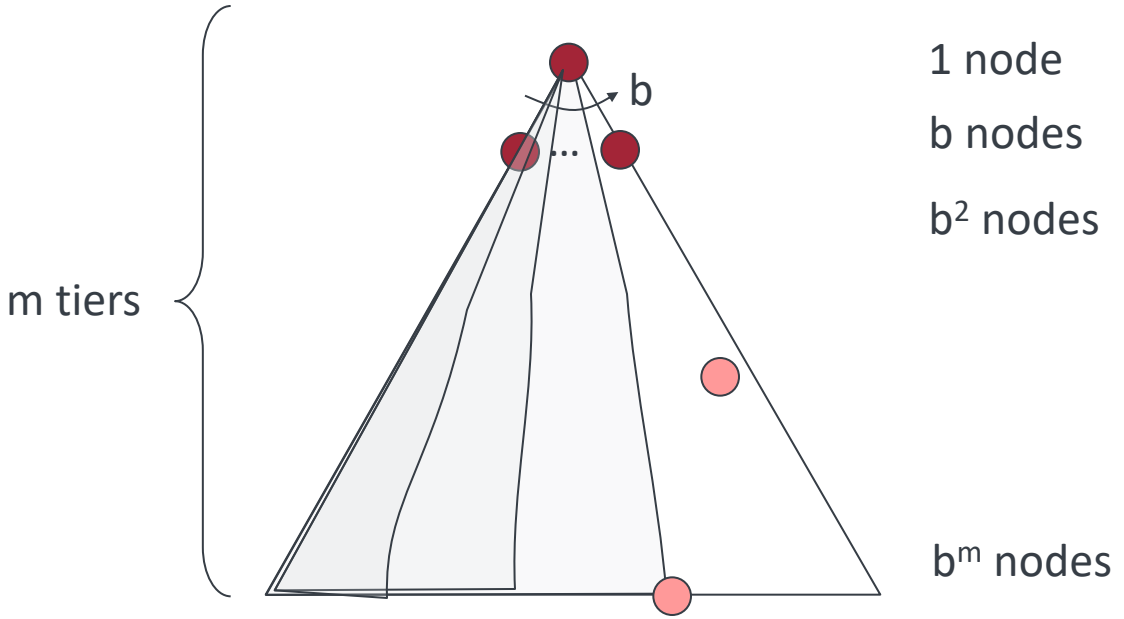
- Cartoon of search tree:
  - b is the branching factor
  - m is the maximum depth
  - solutions at various depths

- Number of nodes in entire tree?
  - $1 + b + b^2 + ... b^m = O(b^m)$

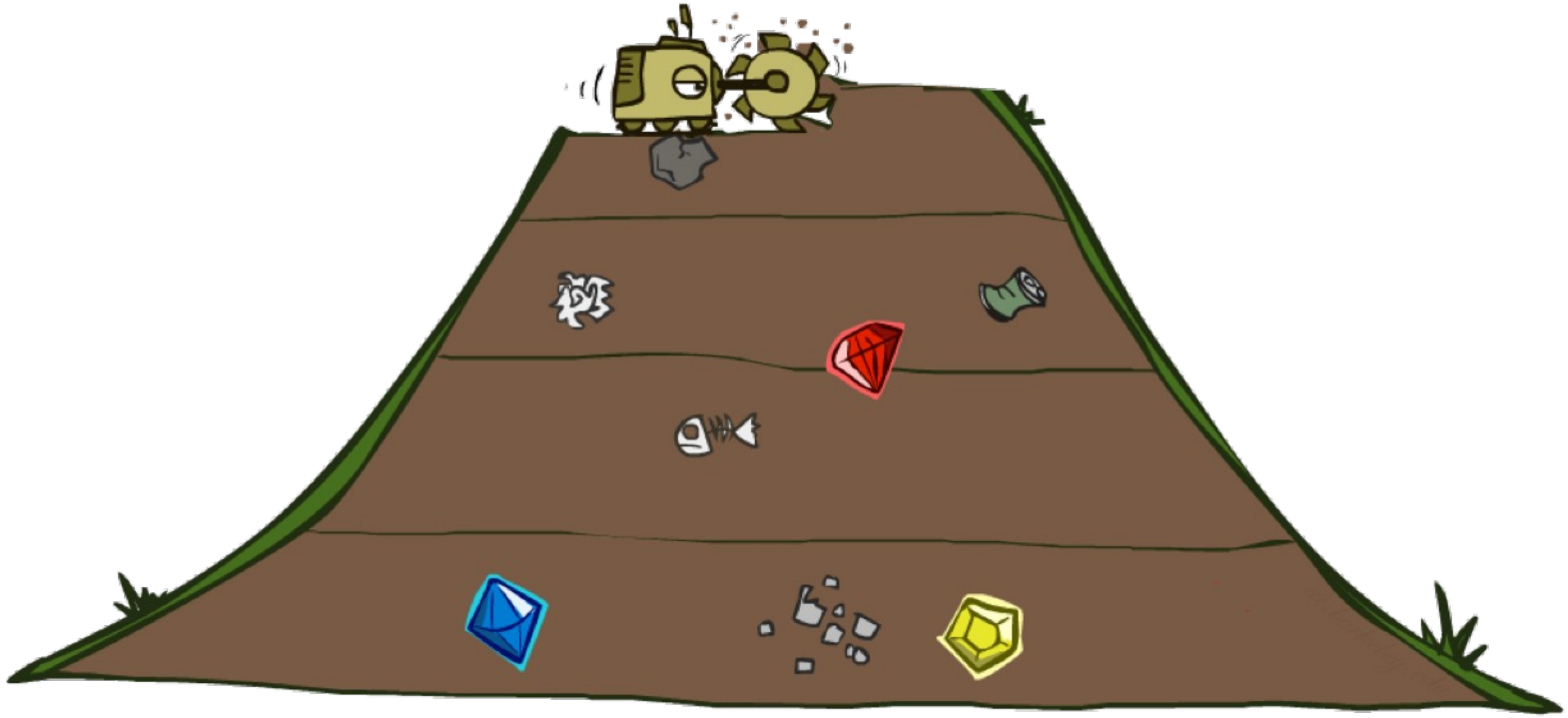


DFS properties

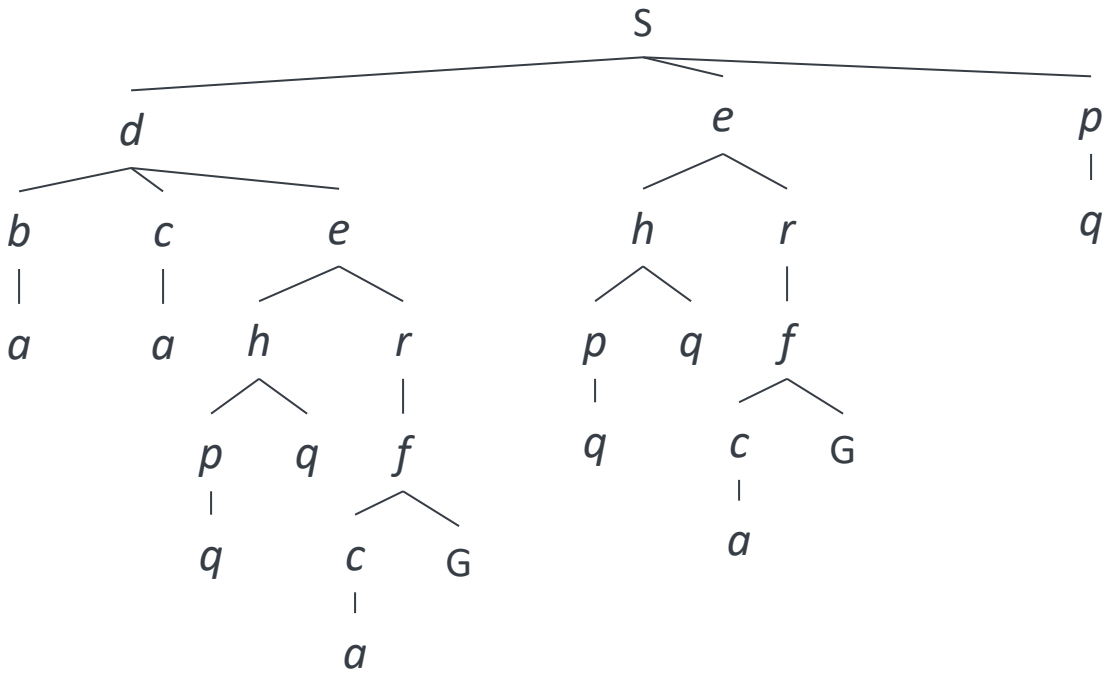
- What nodes DFS expand?
  - Some left prefix of the tree.
  - Could process the whole tree!
  - If m is finite, takes time $O(b^m)$
- How much space does the fringe take?
  - Only has siblings on path to root, so O(bm)
- Is it complete?
  - m could be infinite, so only if we prevent cycles (more later)
- Is it optimal?
  - No, it finds the “leftmost” solution, regardless of depth or cost

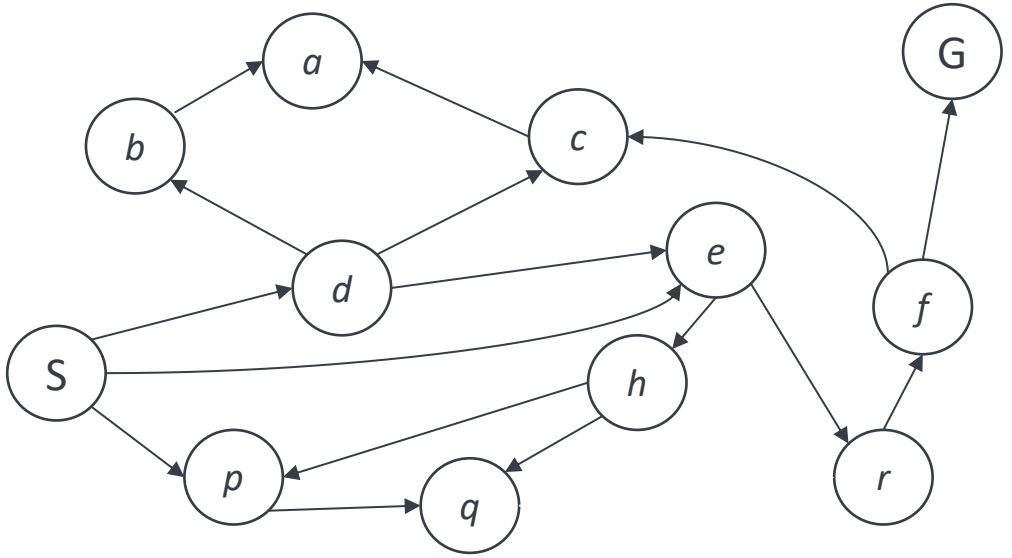


Breadth-First Search (BFS)

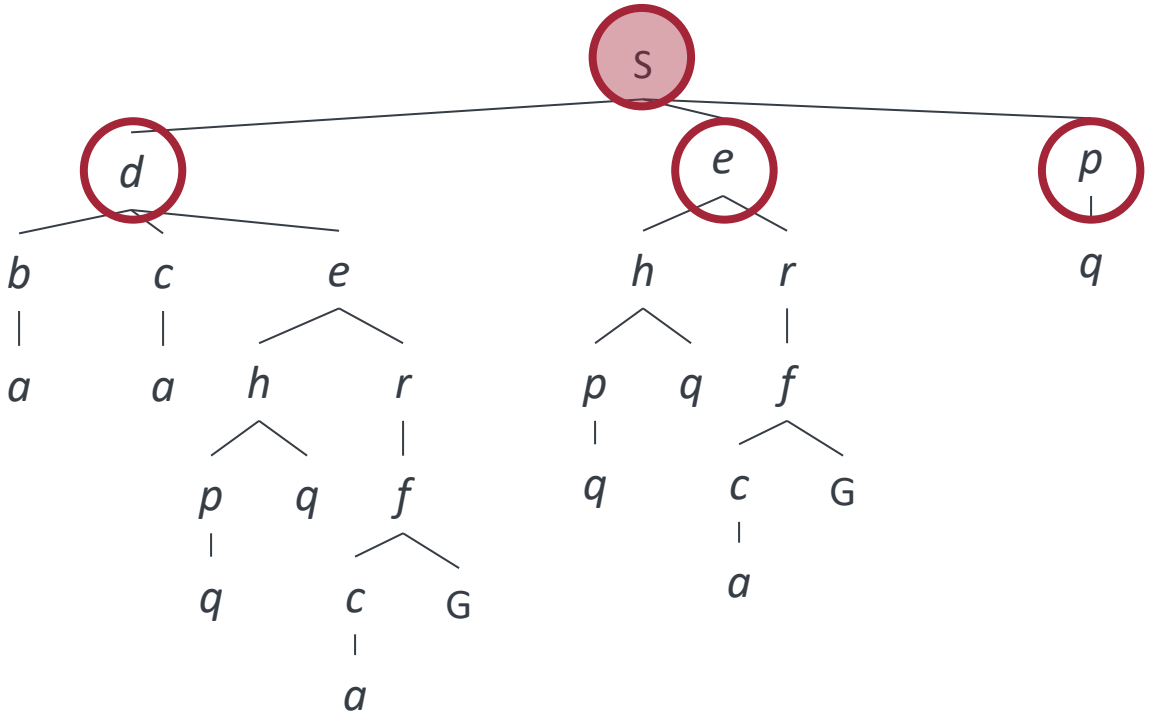


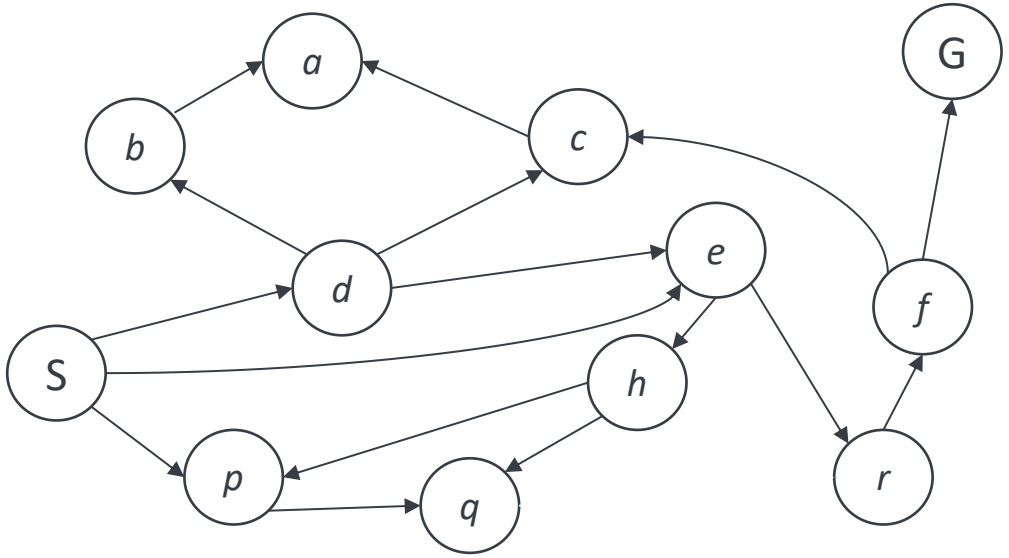
Breadth-First Search (BFS)



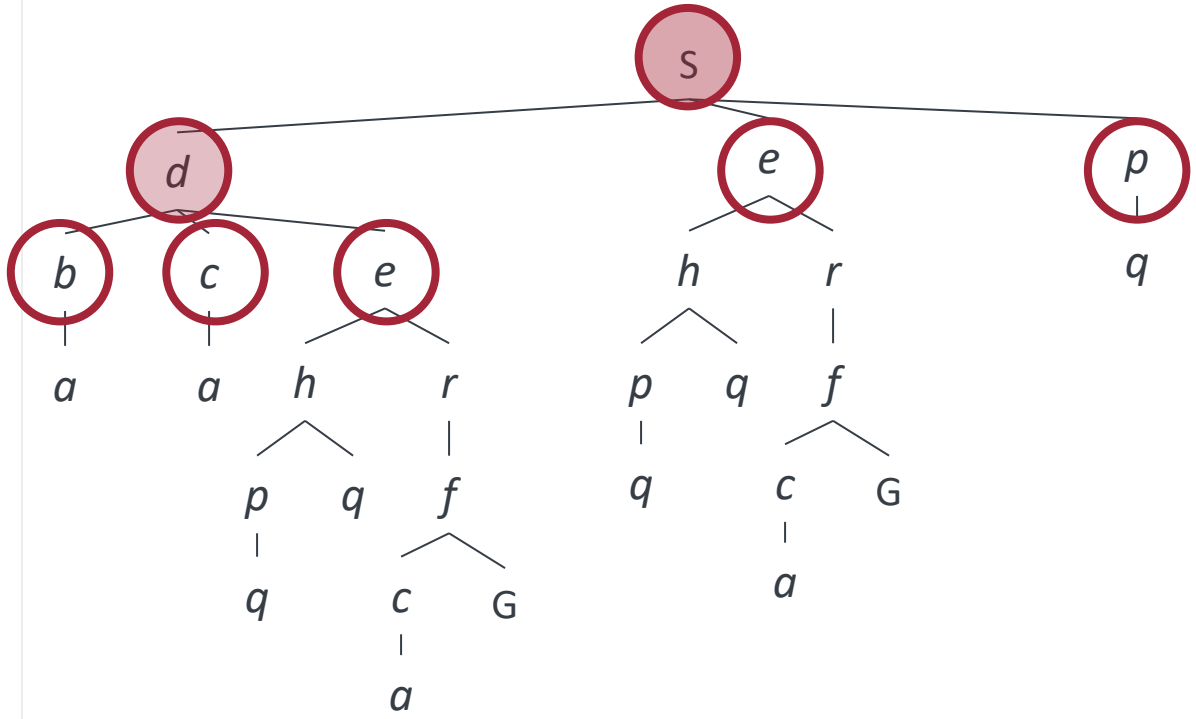


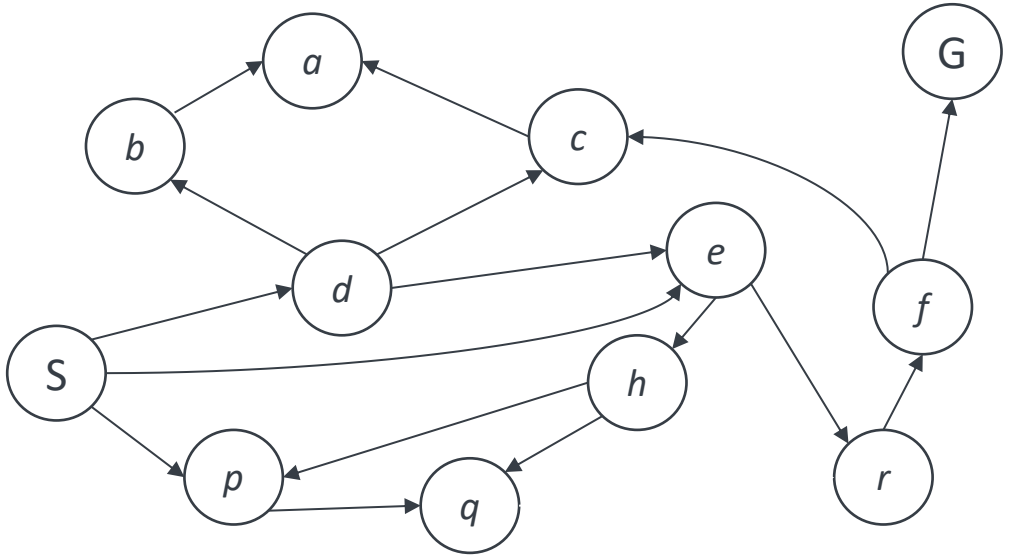
Breadth-First Search (BFS)



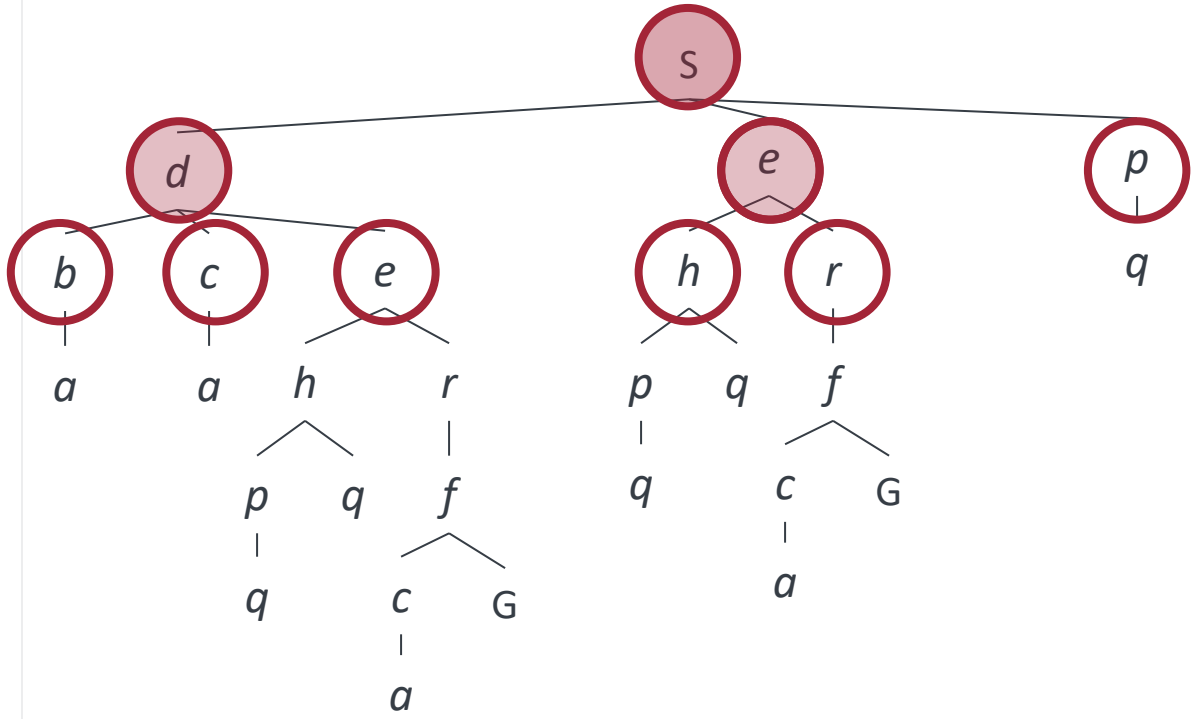


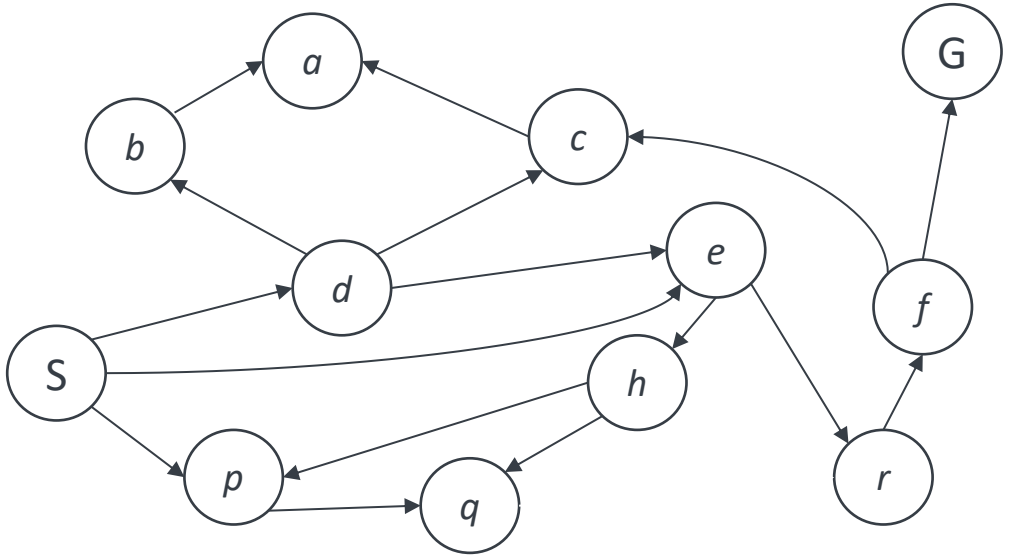
Breadth-First Search (BFS)



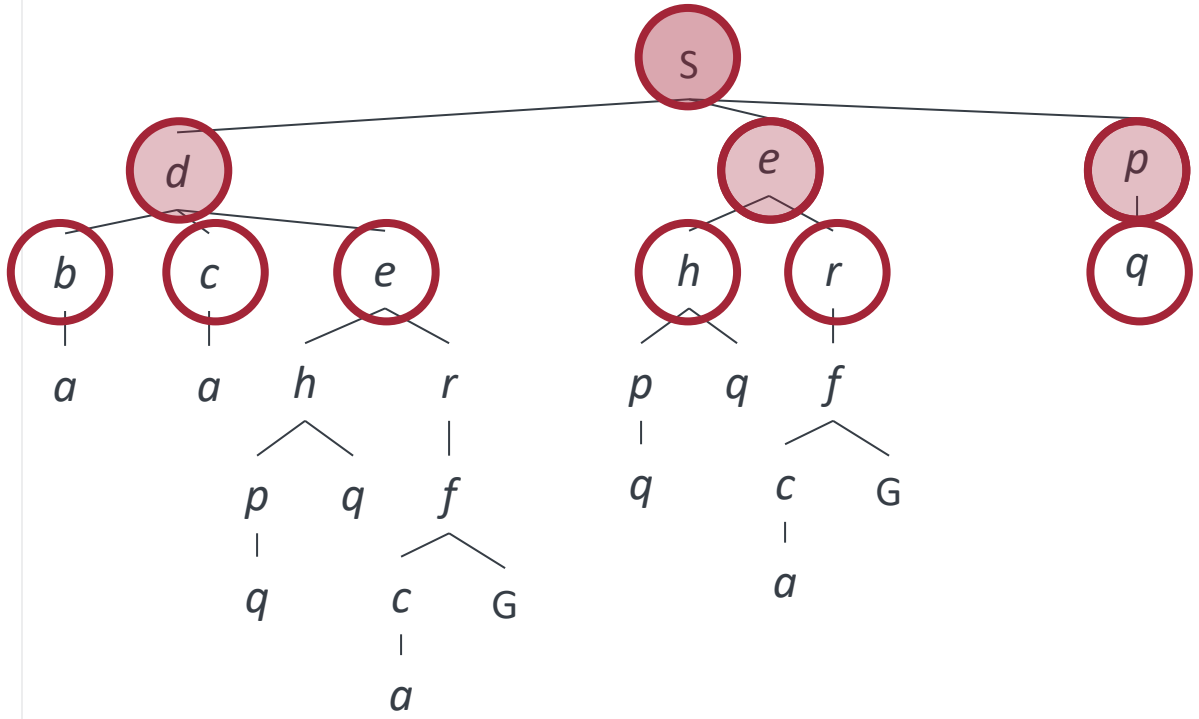


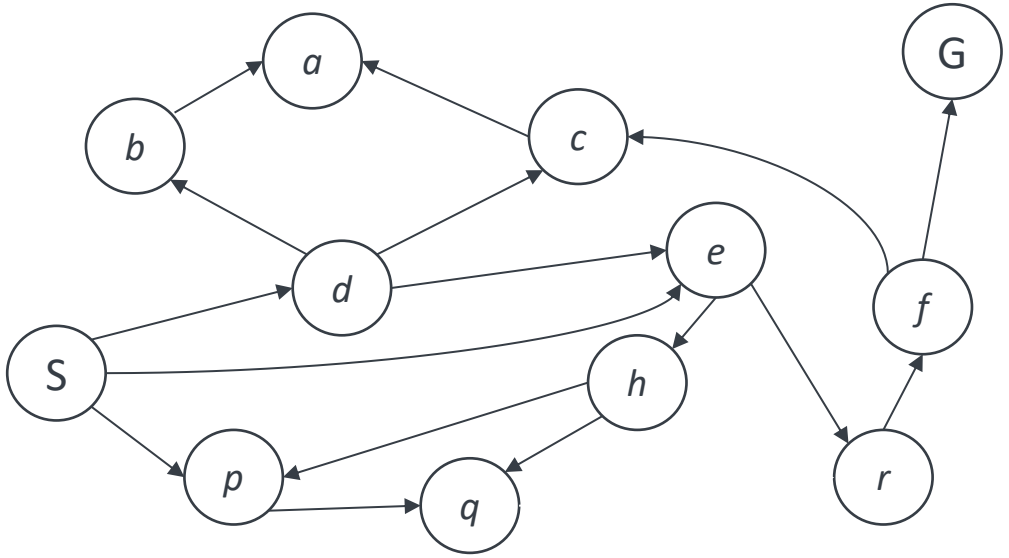
Breadth-First Search (BFS)



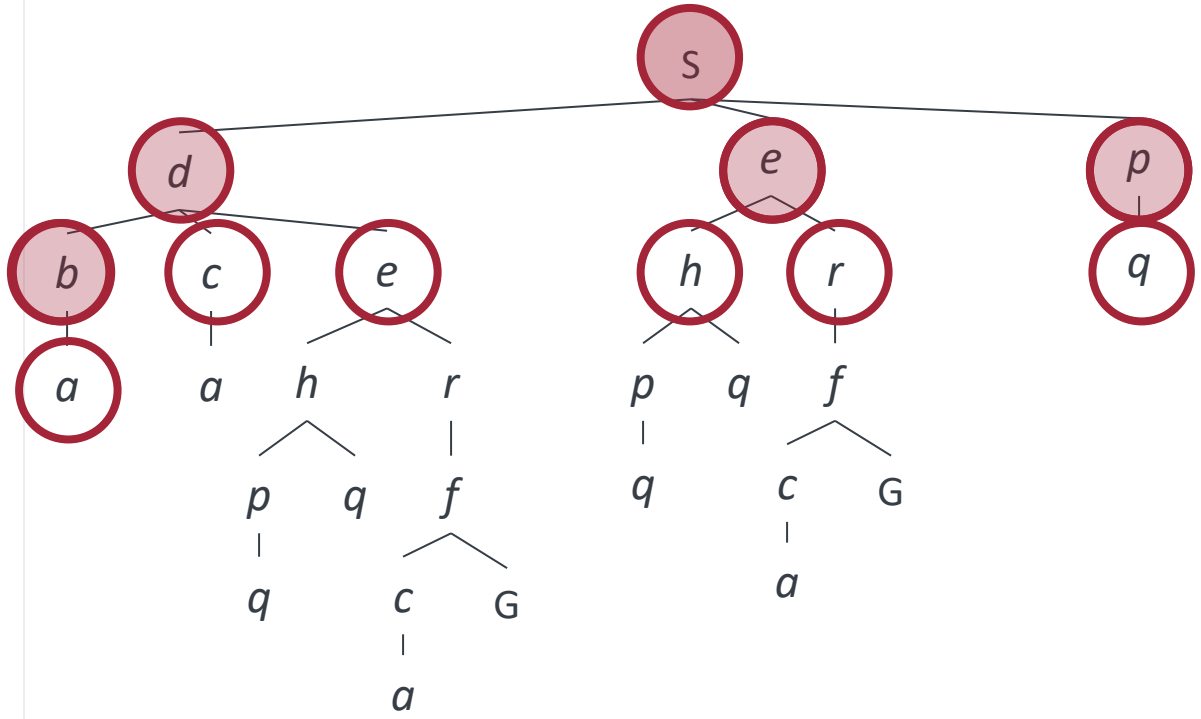


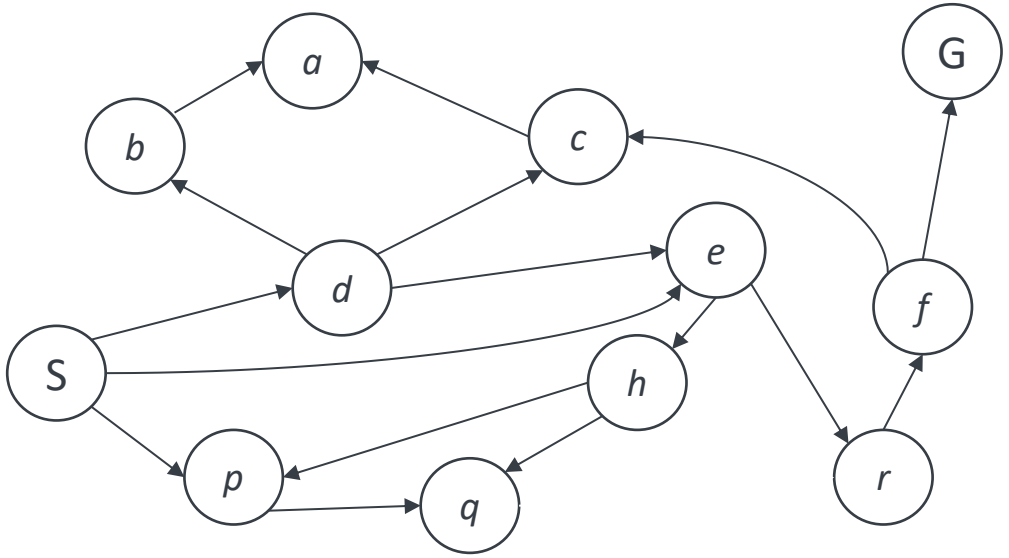
Breadth-First Search (BFS)



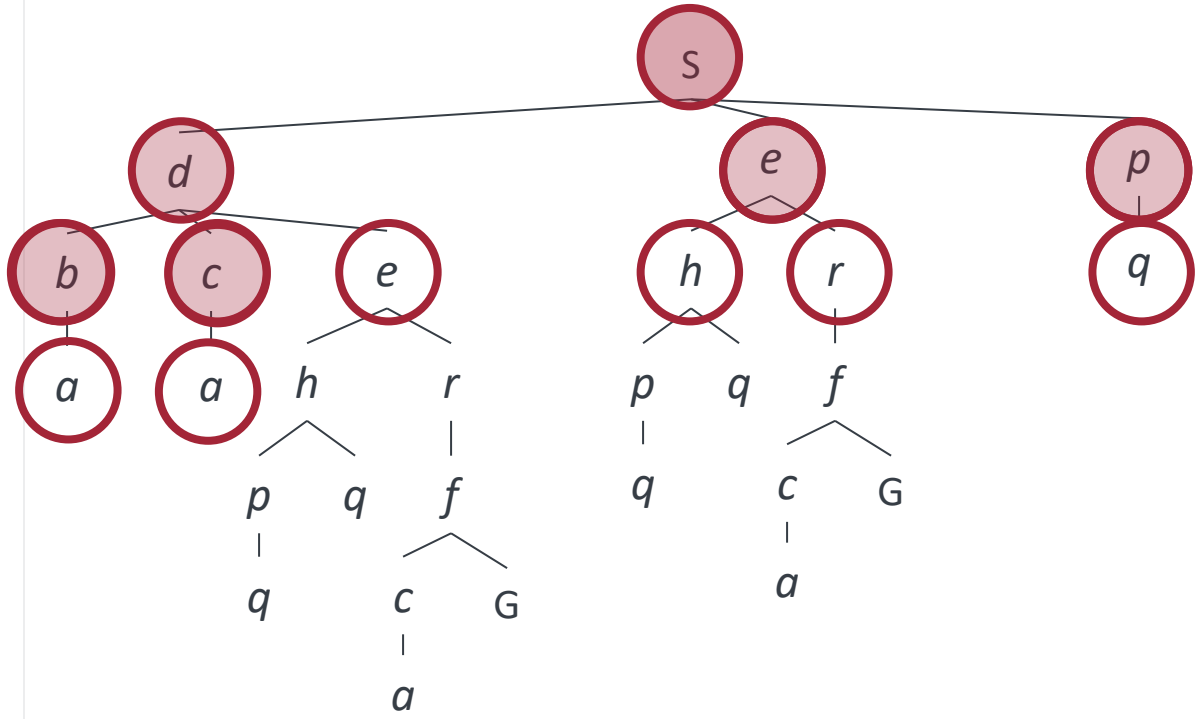


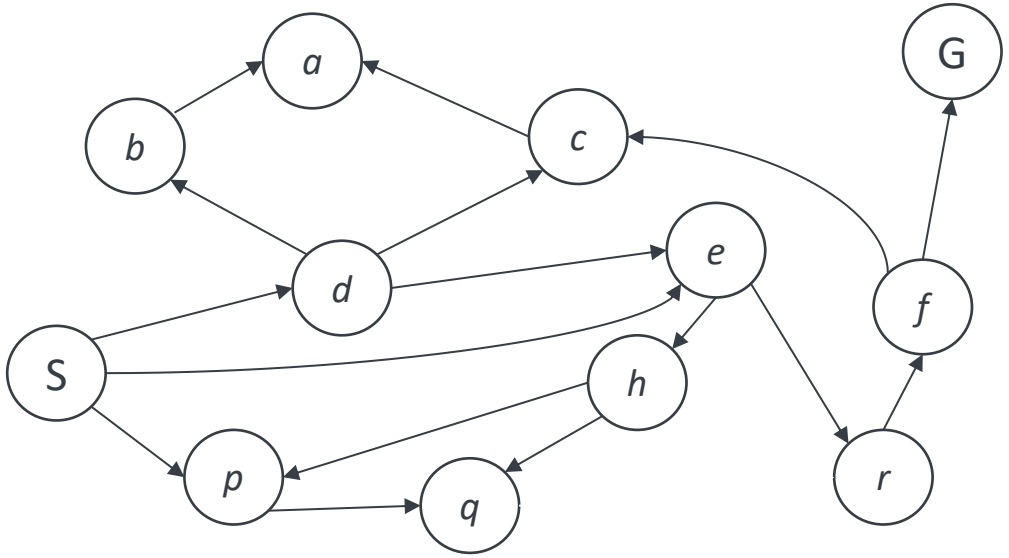
Breadth-First Search (BFS)



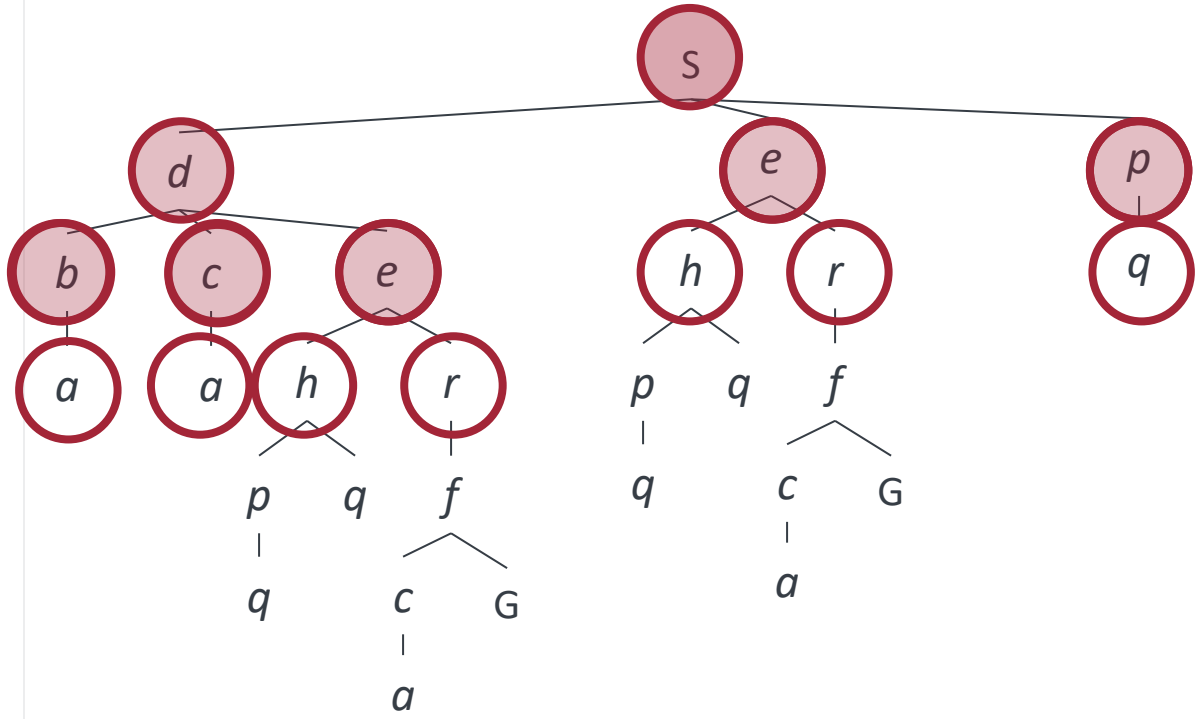


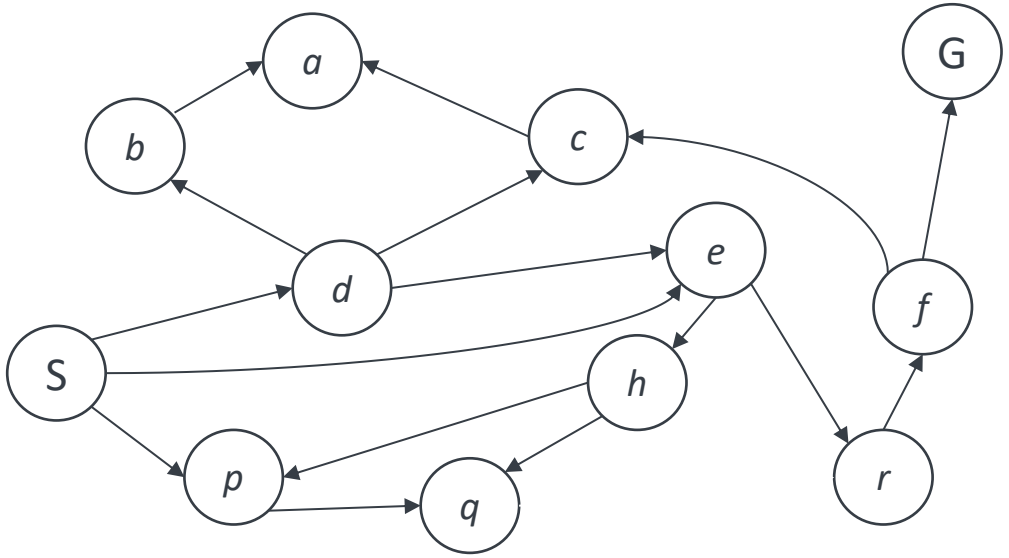
Breadth-First Search (BFS)



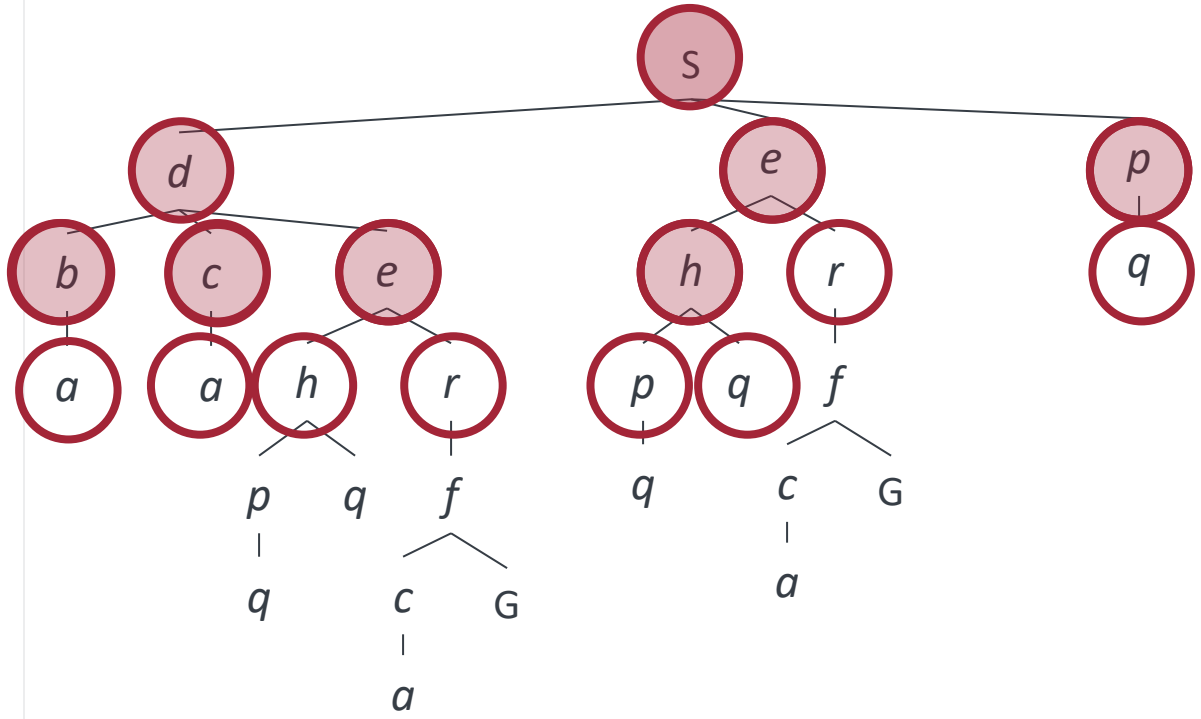


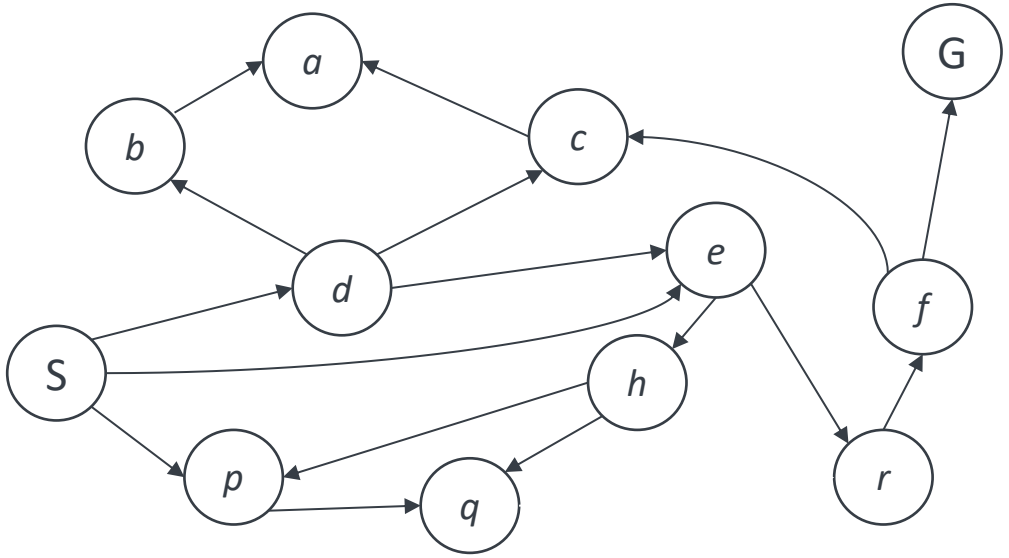
Breadth-First Search (BFS)



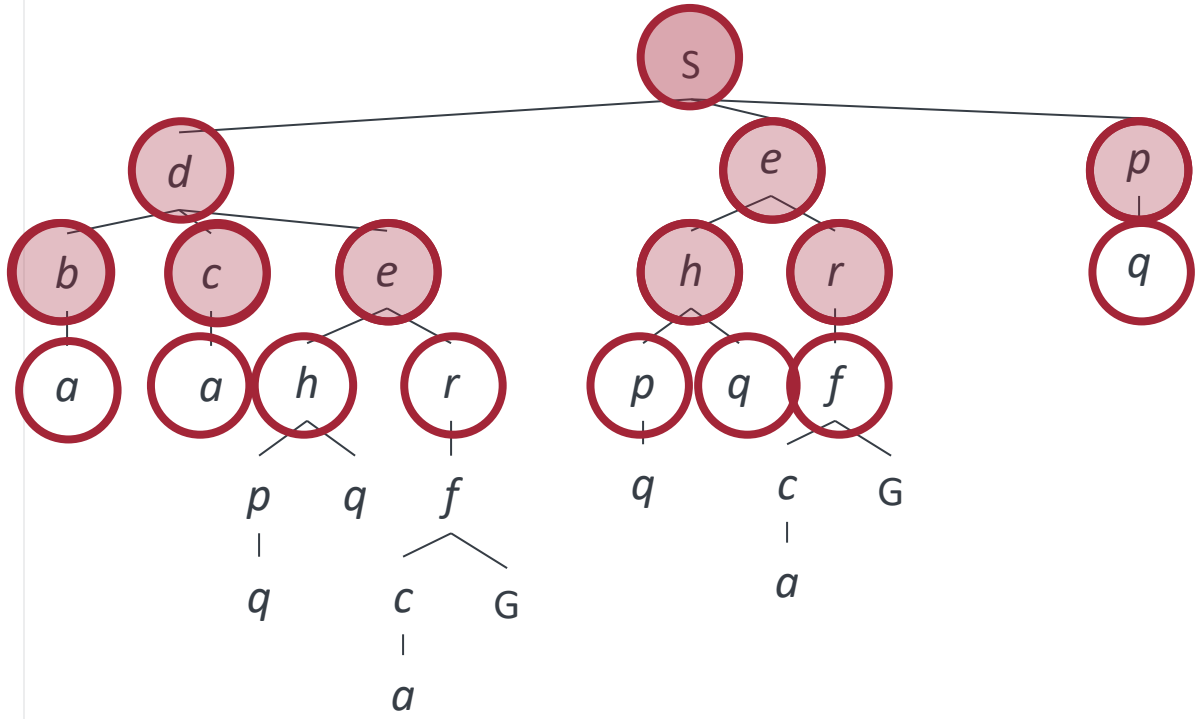


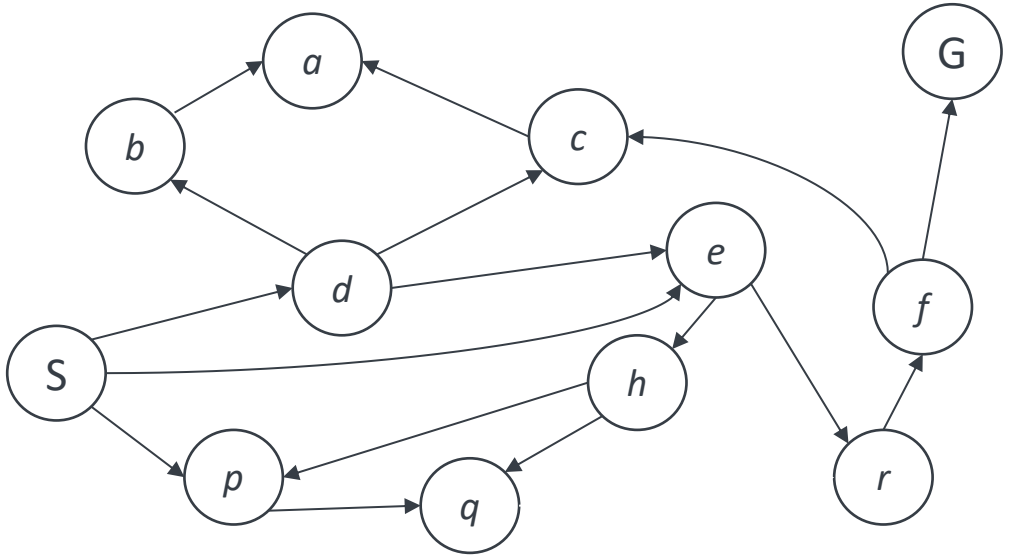
Breadth-First Search (BFS)



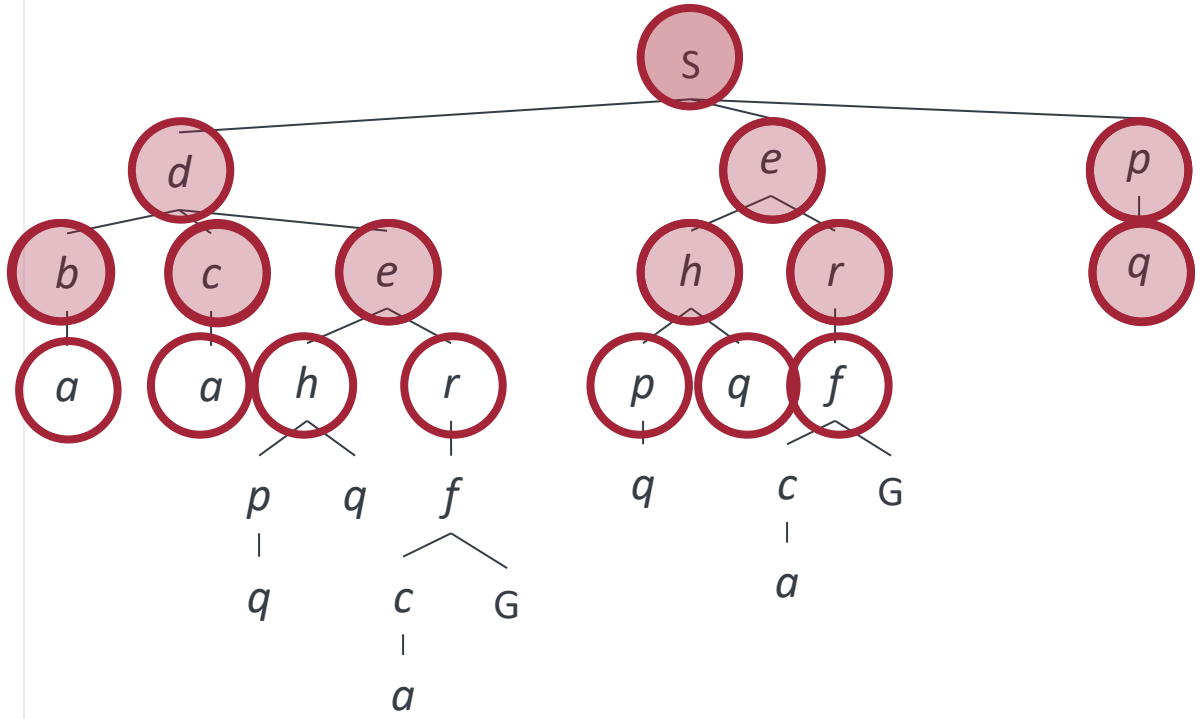


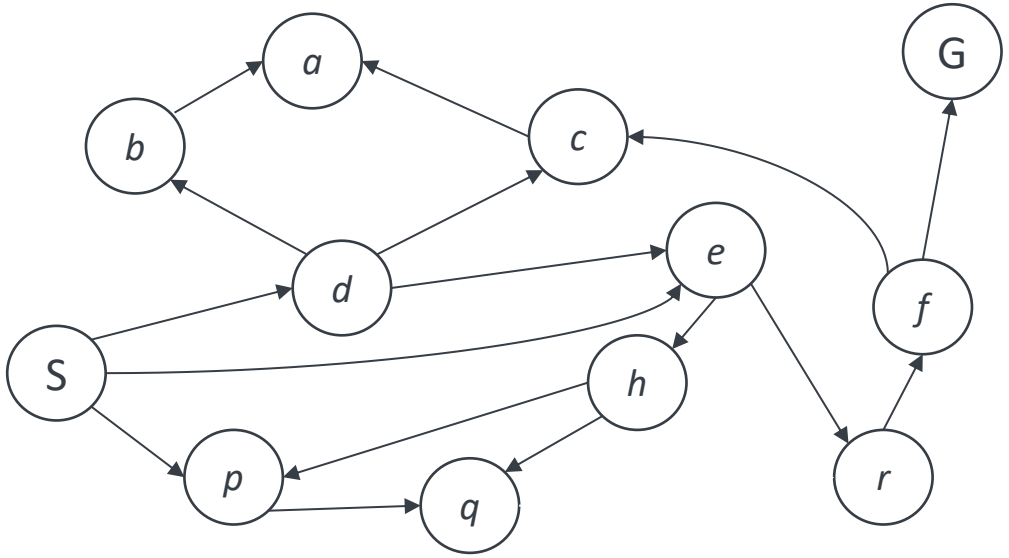
Breadth-First Search (BFS)



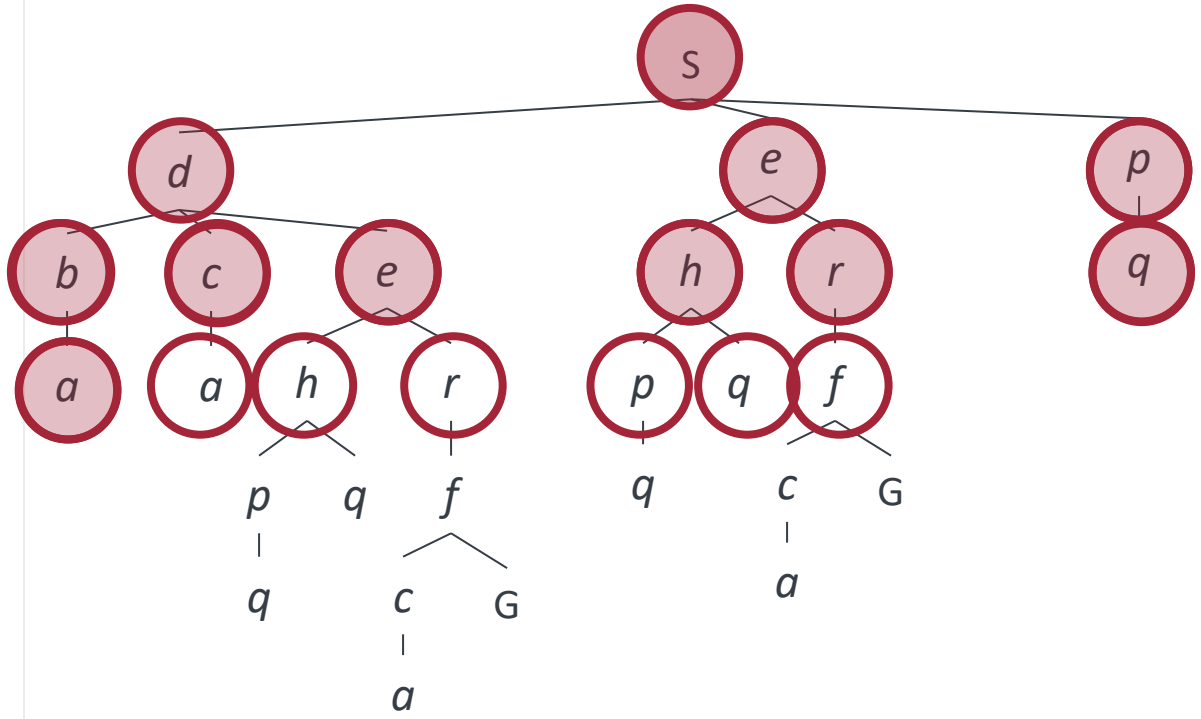


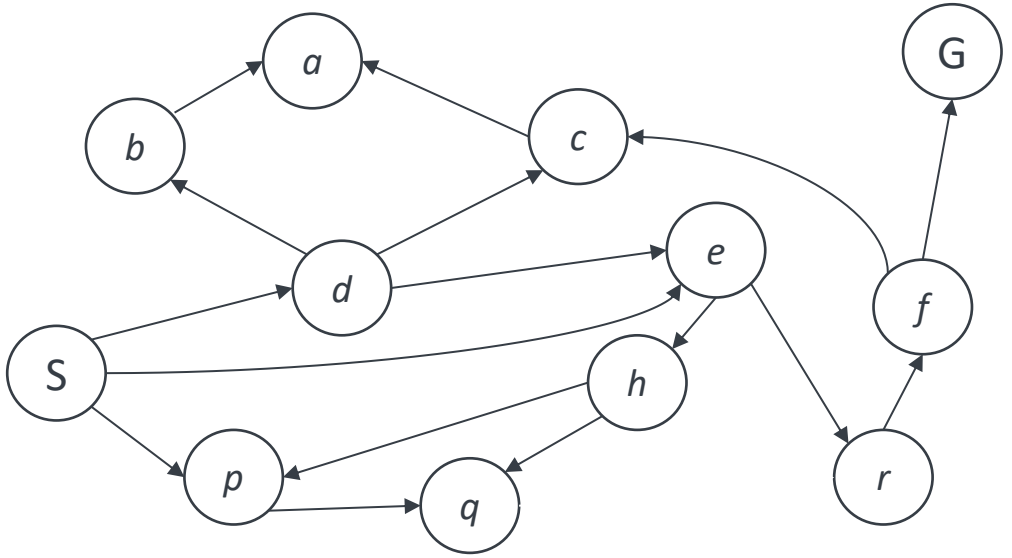
Breadth-First Search (BFS)



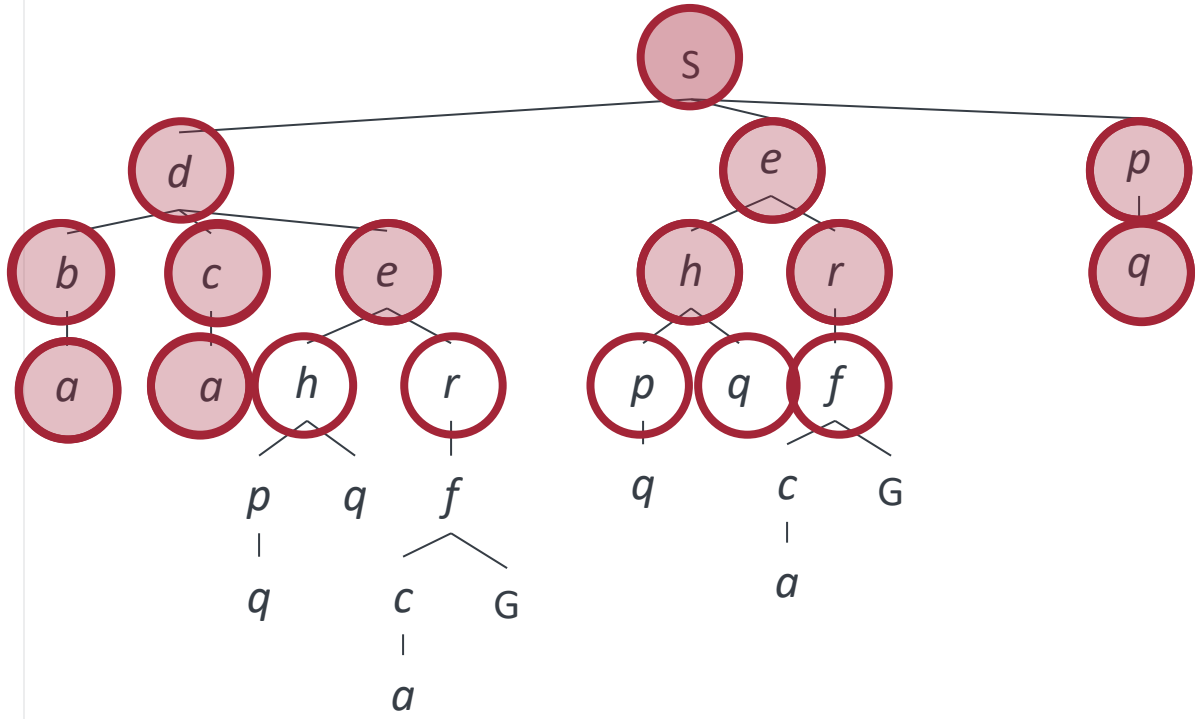


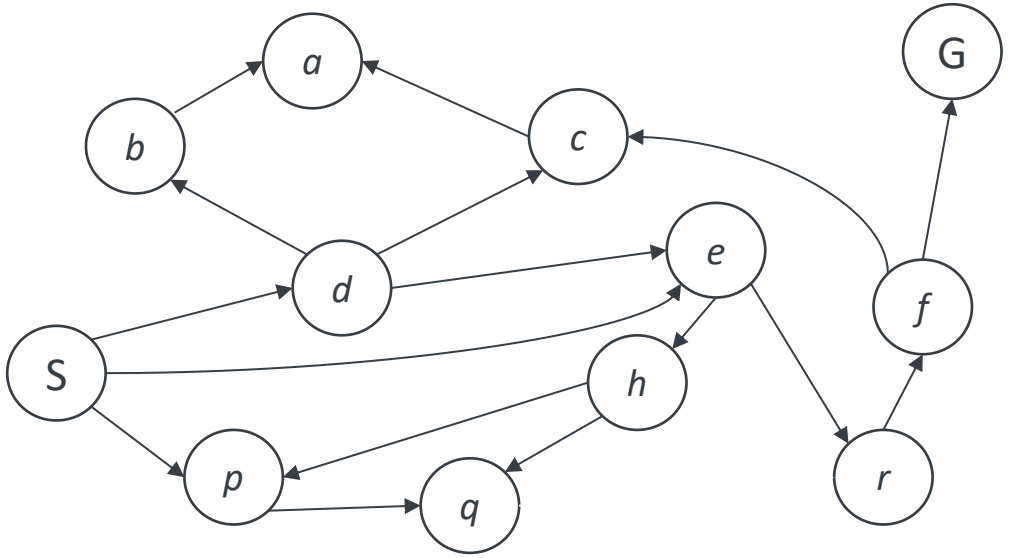
Breadth-First Search (BFS)



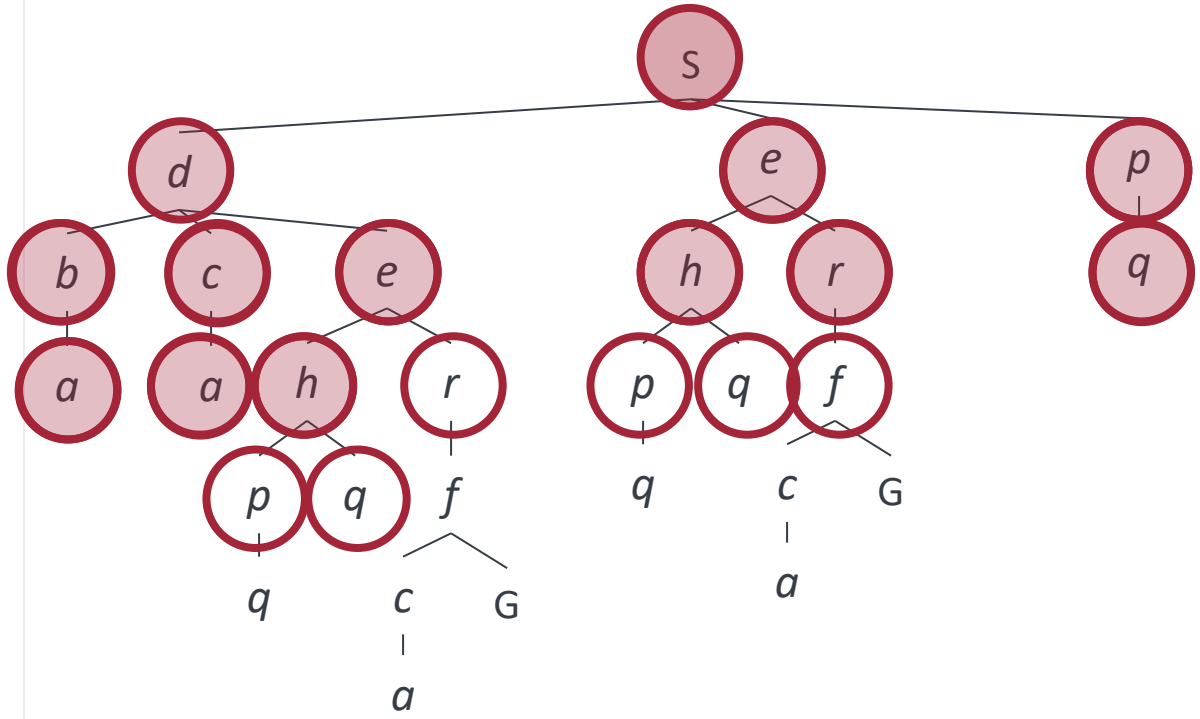


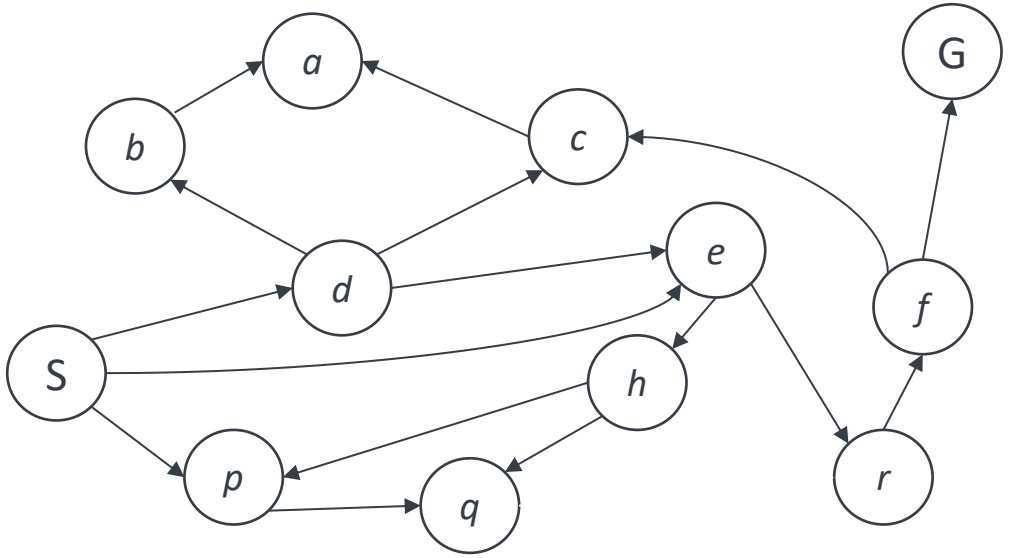
Breadth-First Search (BFS)



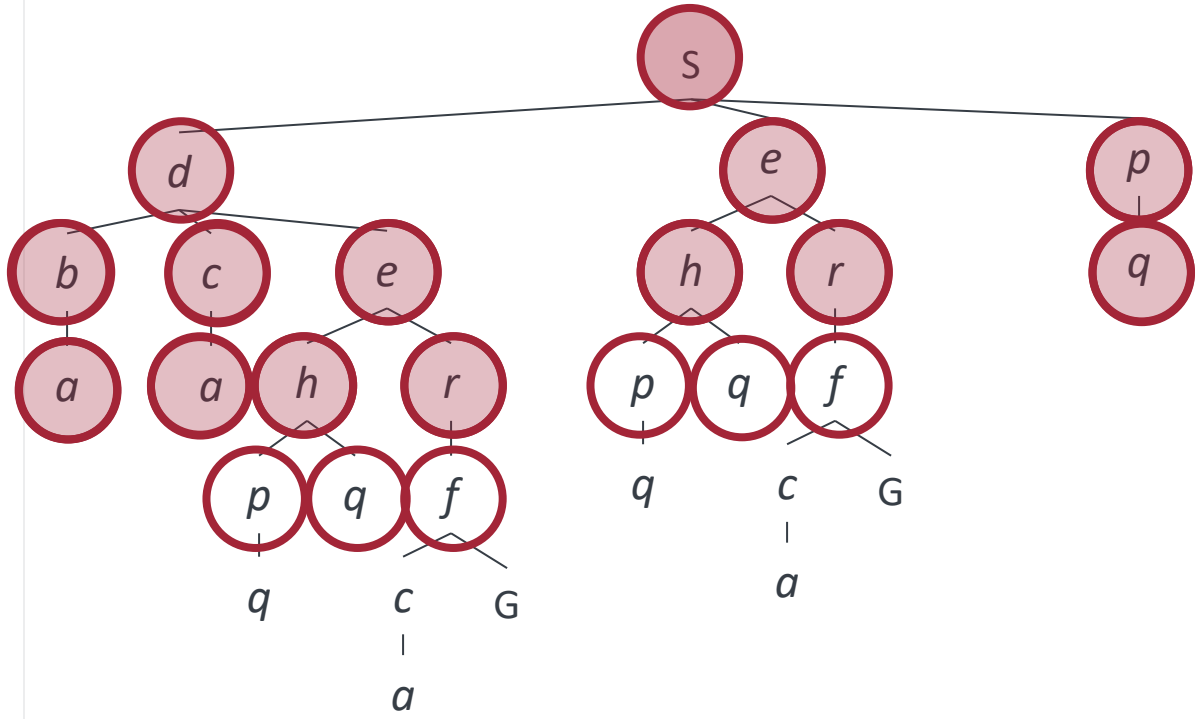


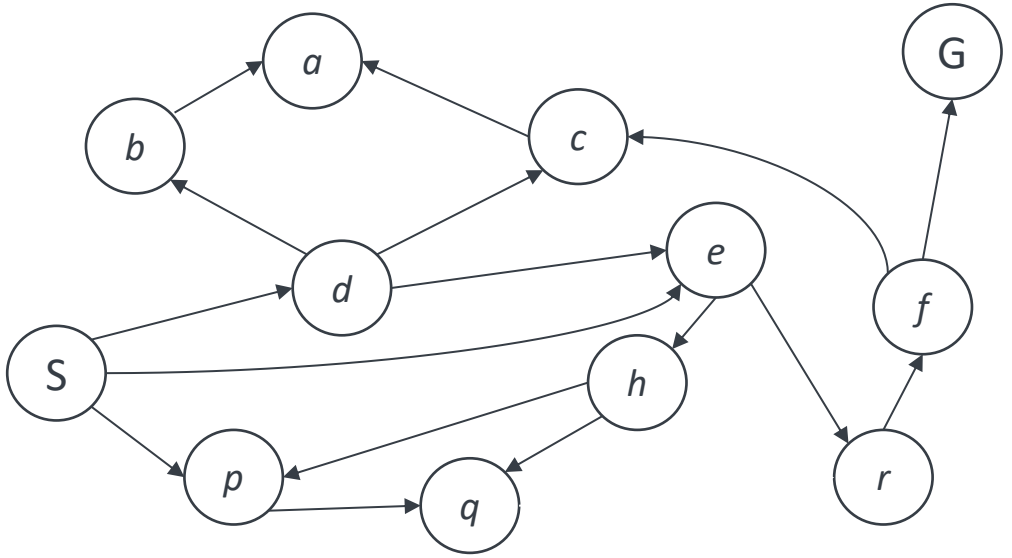
Breadth-First Search (BFS)



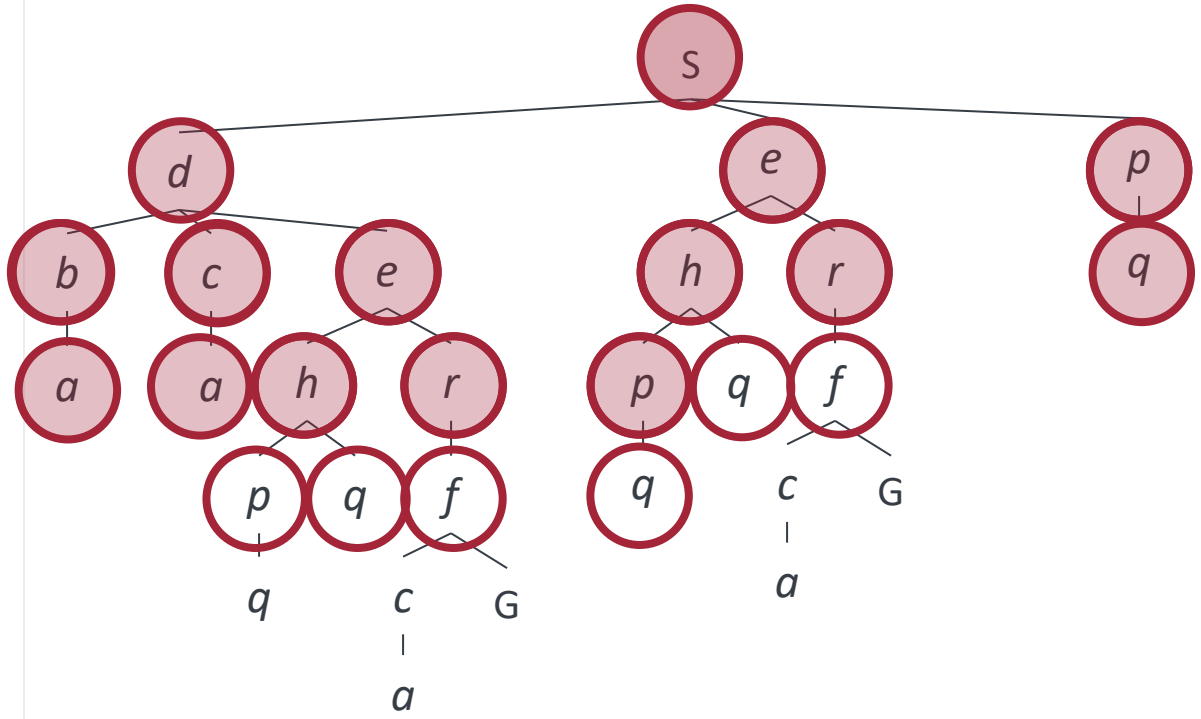


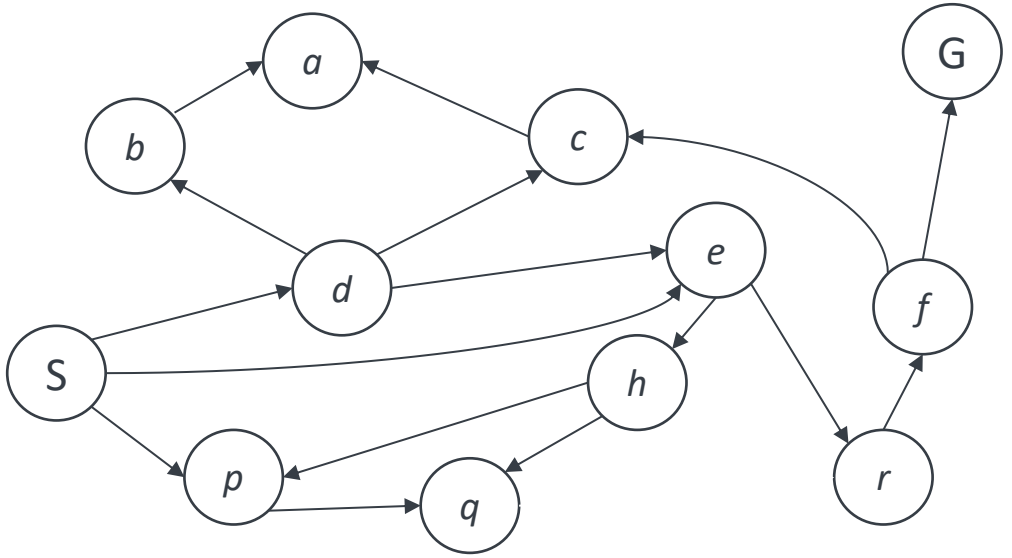
Breadth-First Search (BFS)



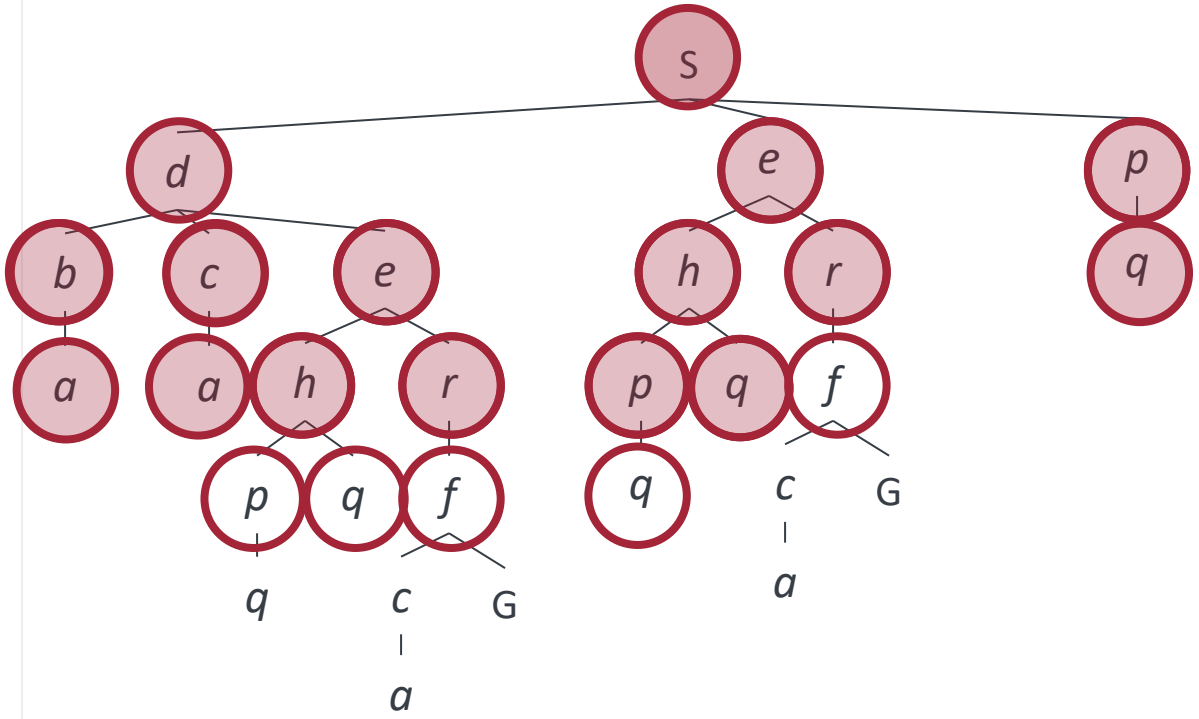


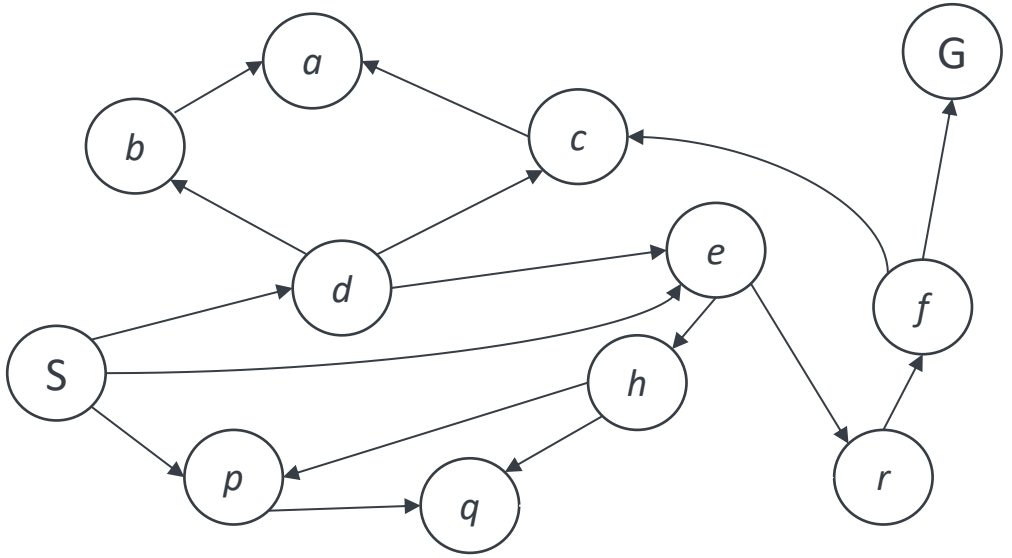
Breadth-First Search (BFS)



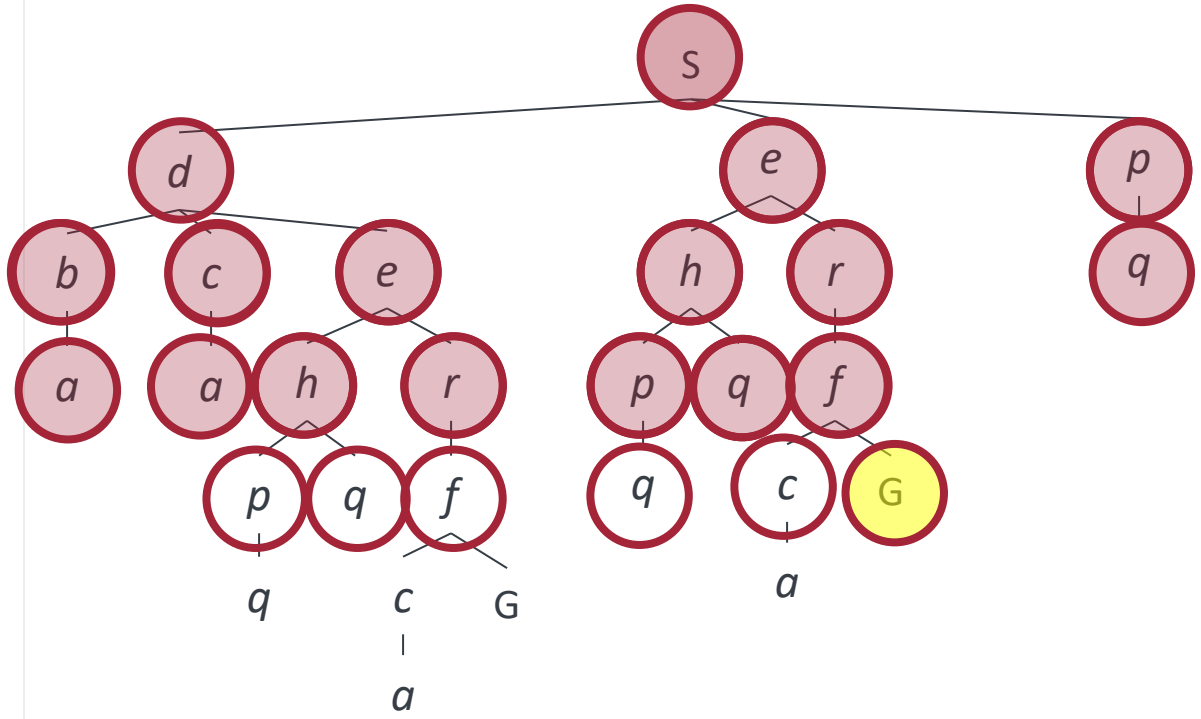


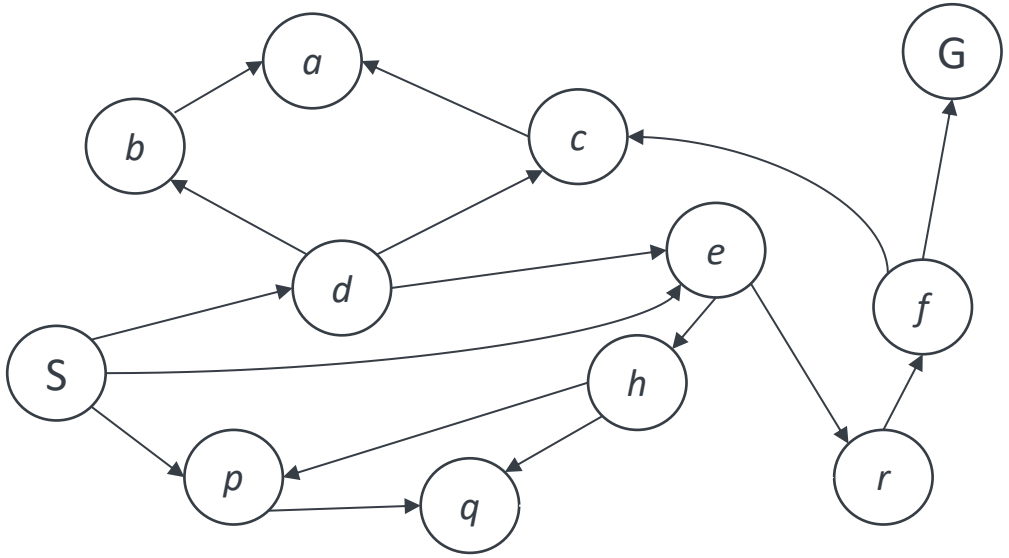
Breadth-First Search (BFS)





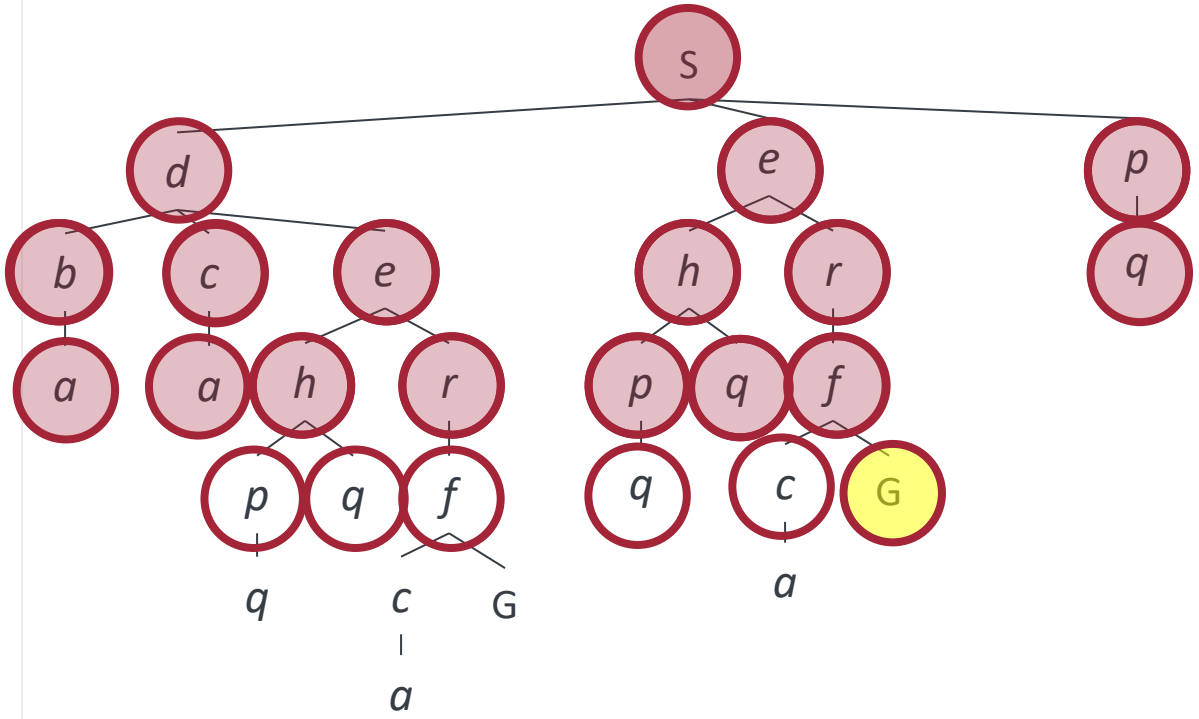
Breadth-First Search (BFS)

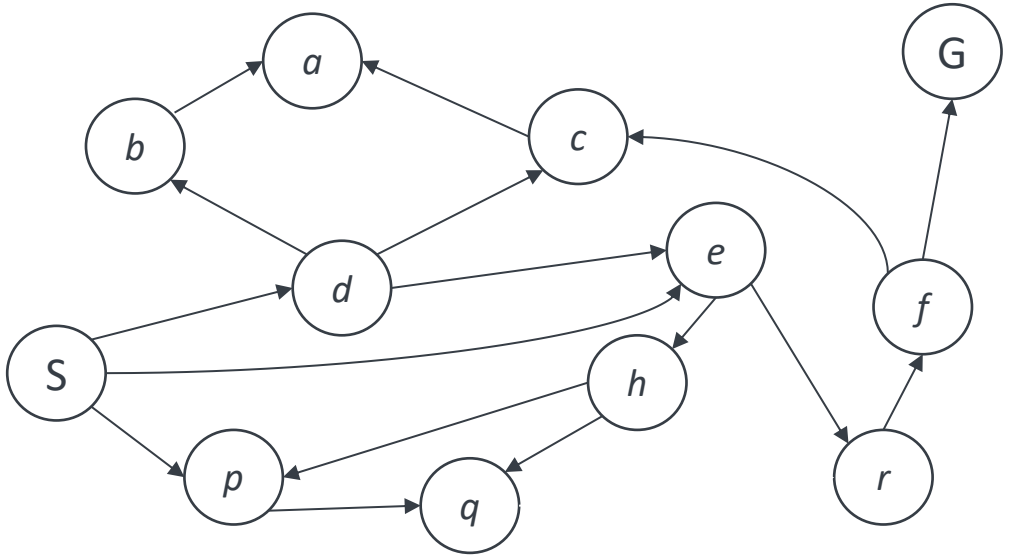




Note that the search ends when the goal node is **generated** (not expanded).

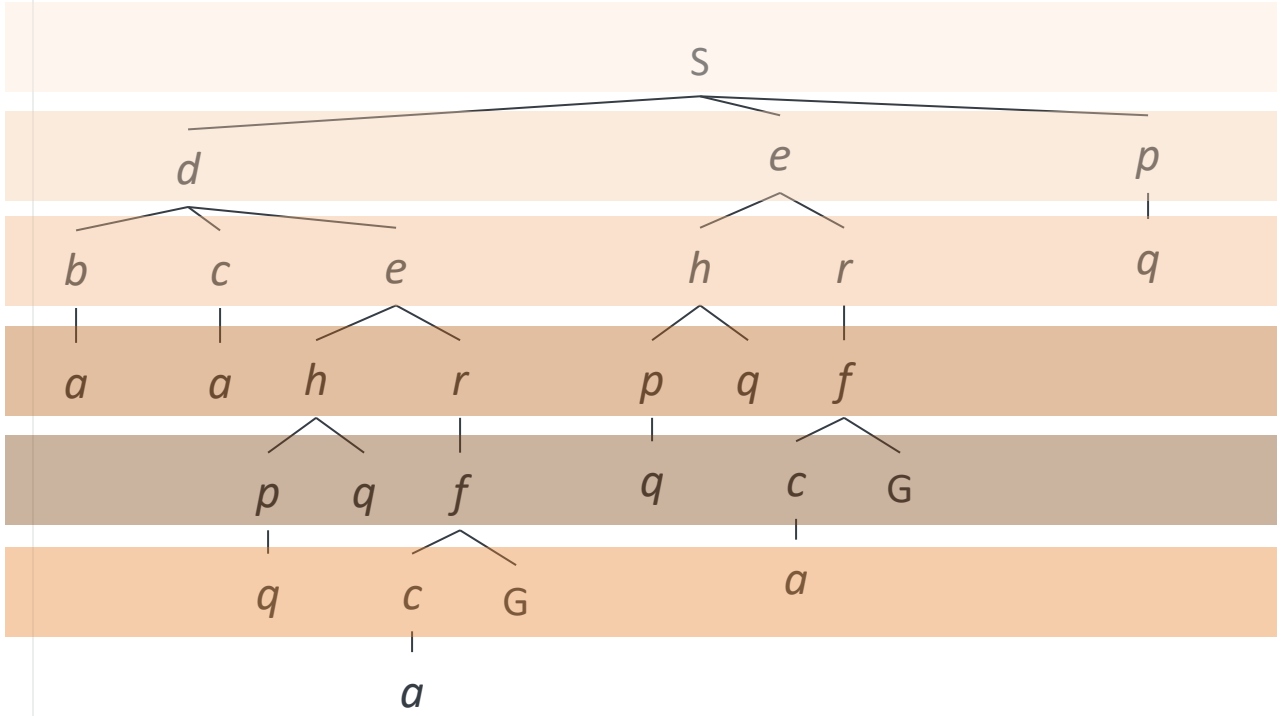
Breadth-First Search (BFS)

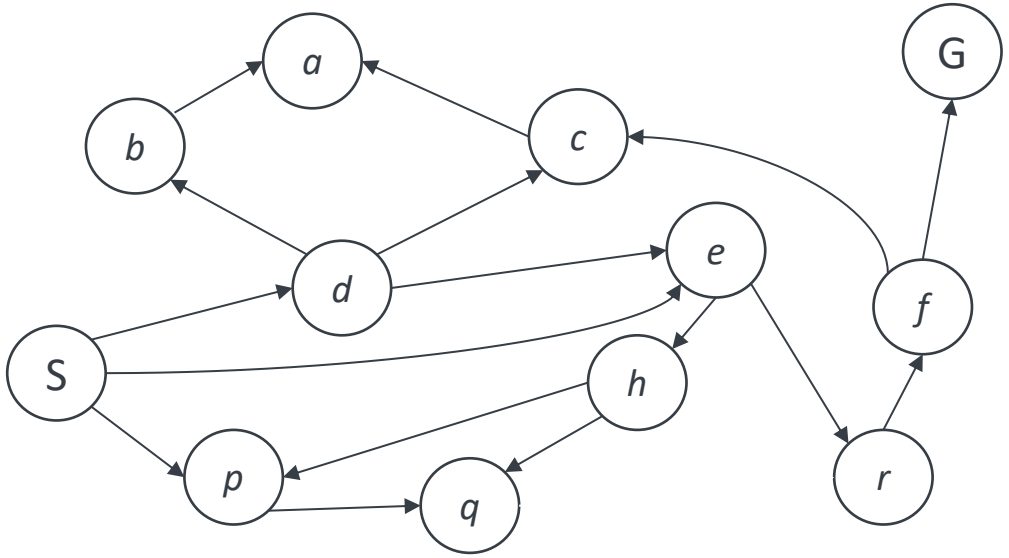




Number of expanded nodes: 17

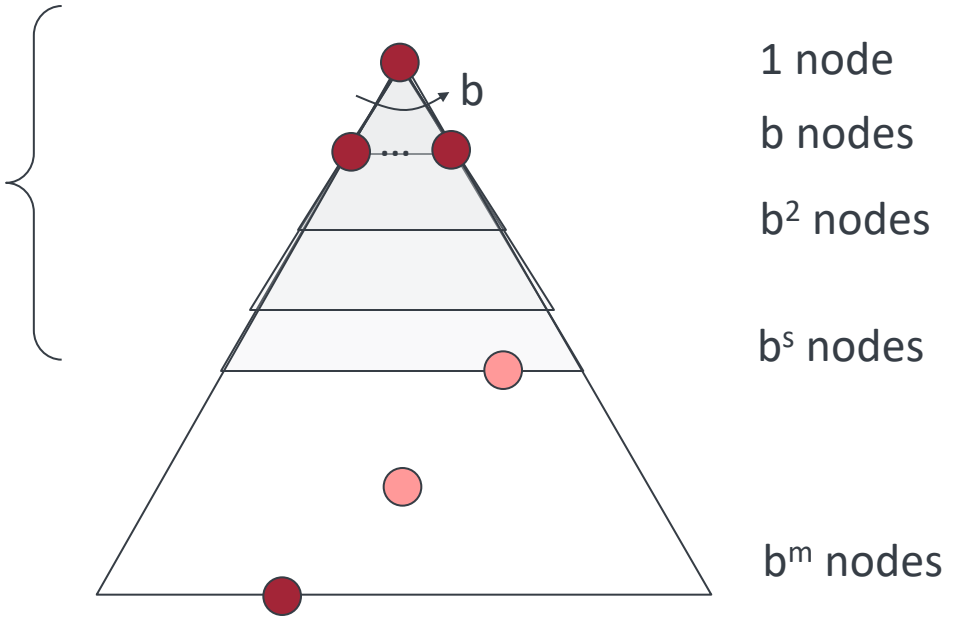
Breadth-First Search (BFS)





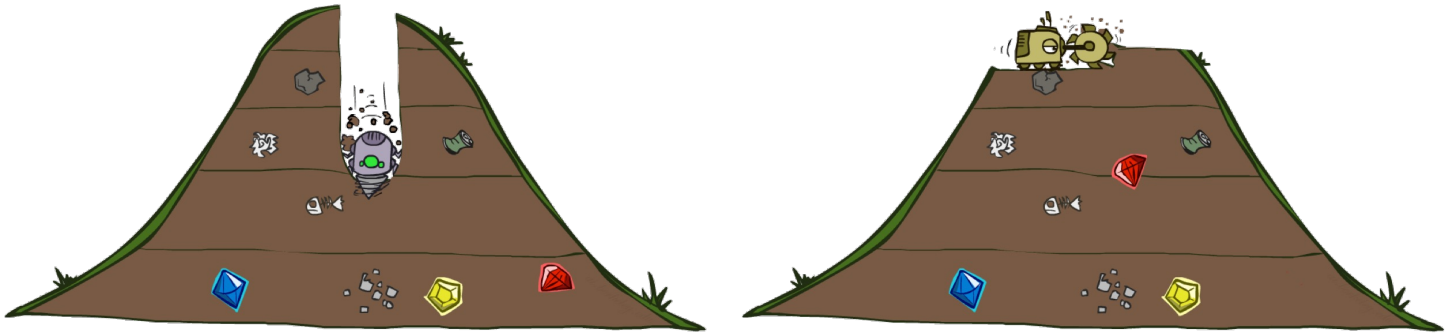
BFS Properties

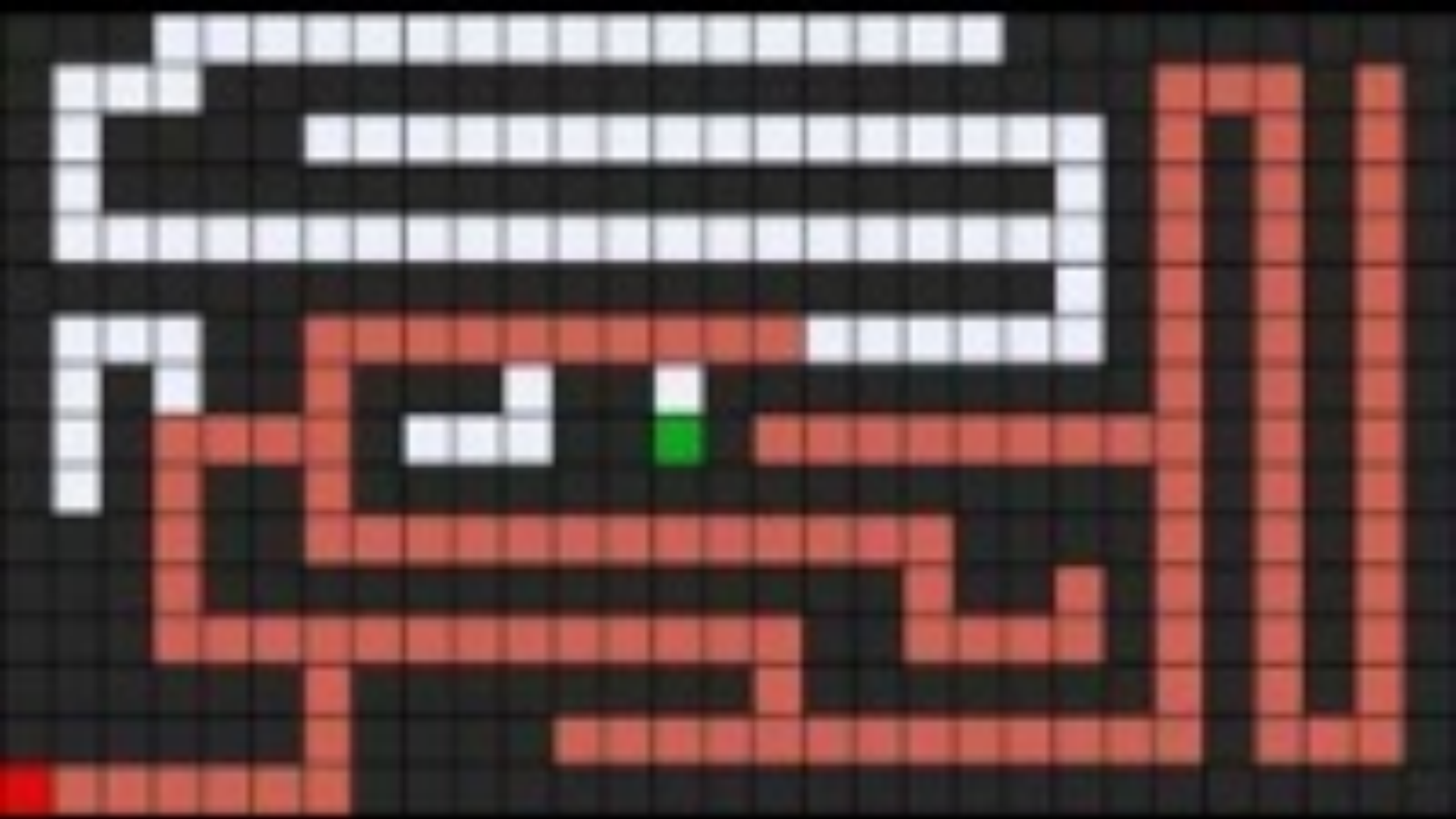
- What nodes does BFS expand?
  - Processes all nodes above shallowest solution
  - Let depth of shallowest solution be s
  - Search takes time $O(b^s)$
- How much space does the fringe take?
  - Has roughly the last tier, so $O(b^s)$
  - Space complexity is a real problem.
- Is it complete?
  - s must be finite if a solution exists, so yes!
- Is it optimal?
  - Only if costs are all 1 (more on costs later)



Quiz: DFS vs. BFS

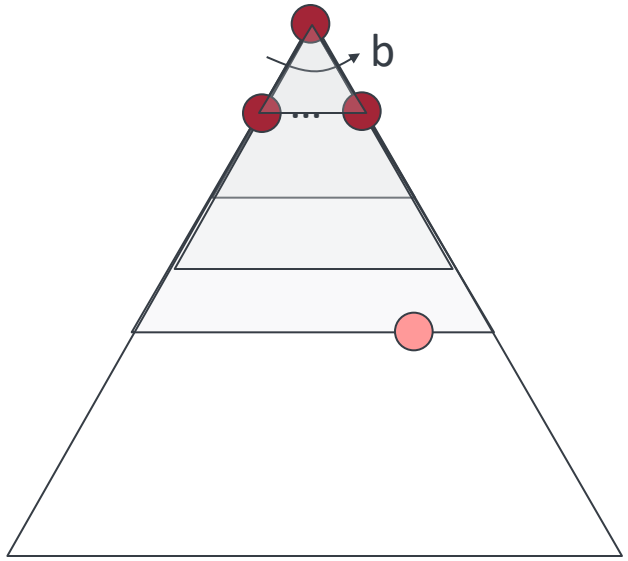
- When will BFS outperform DFS?
- When will DFS outperform BFS?





Iterative Deepening Search (IDS)

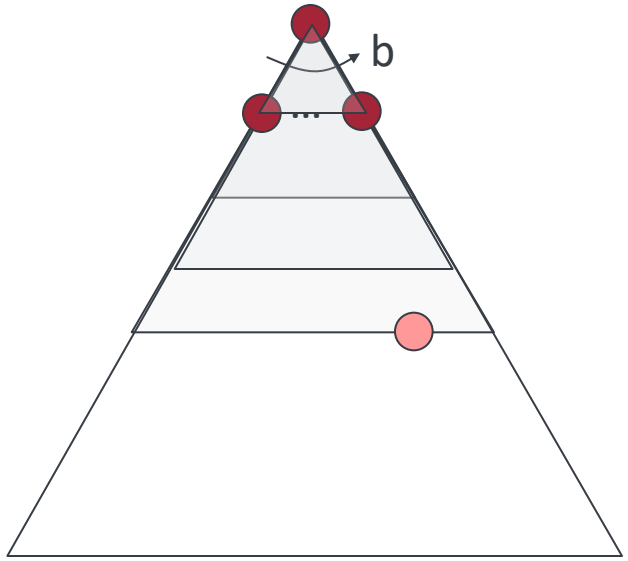
- Idea: get DFS's space advantage with BFS's time / shallow-solution advantages
  - Run a DFS with depth limit 1. If no solution...
  - Run a DFS with depth limit 2. If no solution...
  - Run a DFS with depth limit 3. .....



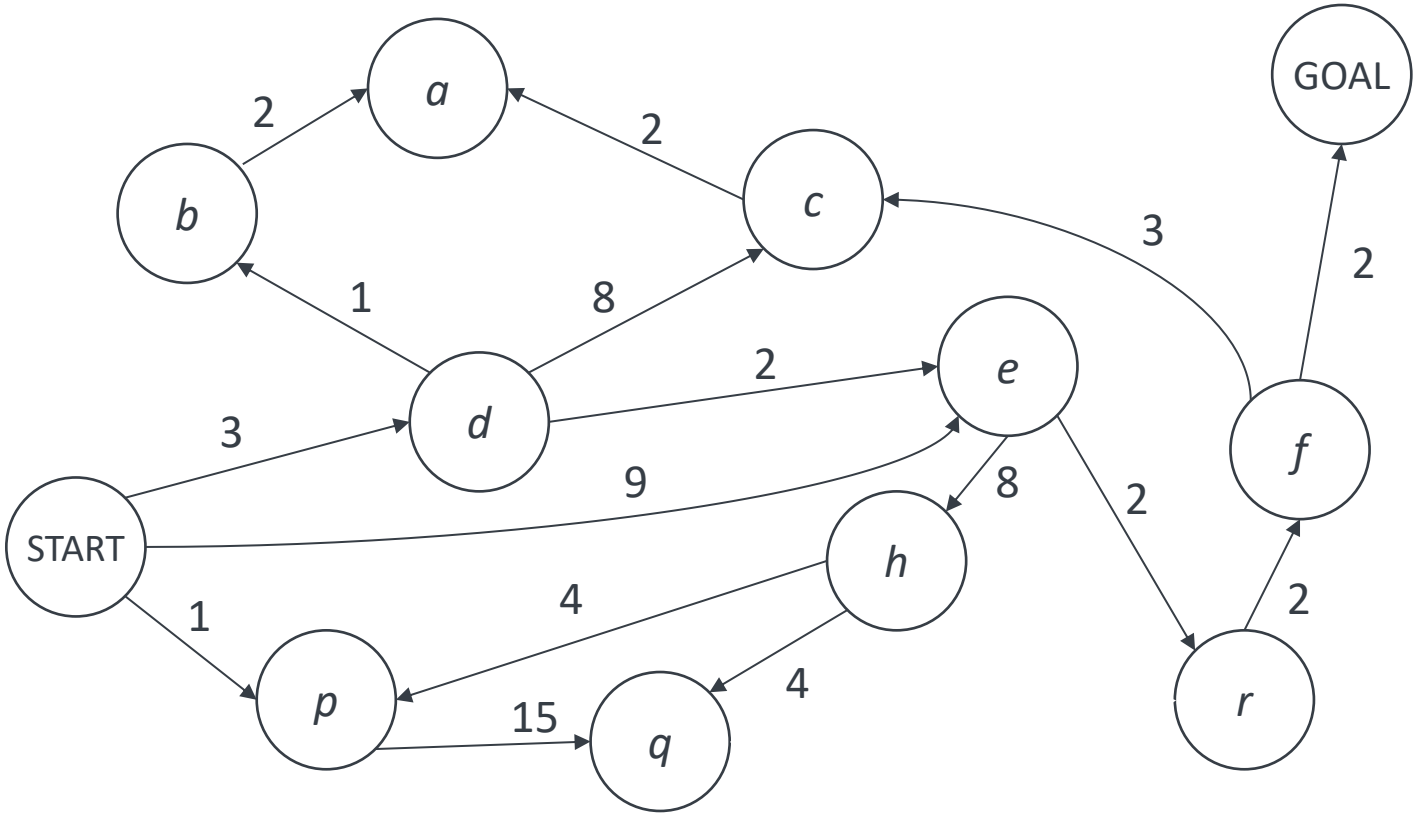
- Isn't that wastefully redundant?
  - Generally most work happens in the lowest level searched, so not so bad!

IDS properties

- What nodes IDS expand?
  - Some left prefix of the tree.
  - $mb^0 + (m-1)b^1 + (m-2)b^2 + \ldots + b^{m-1} = O(b^m)$
- How much space does the leaf nodes take?
  - Only has siblings on path to root, so O(bm)
- Is it complete?
  - m could be infinite, so only if we prevent cycles (more later)
- Is it optimal?
  - Only if costs are all 1 (more on costs later)

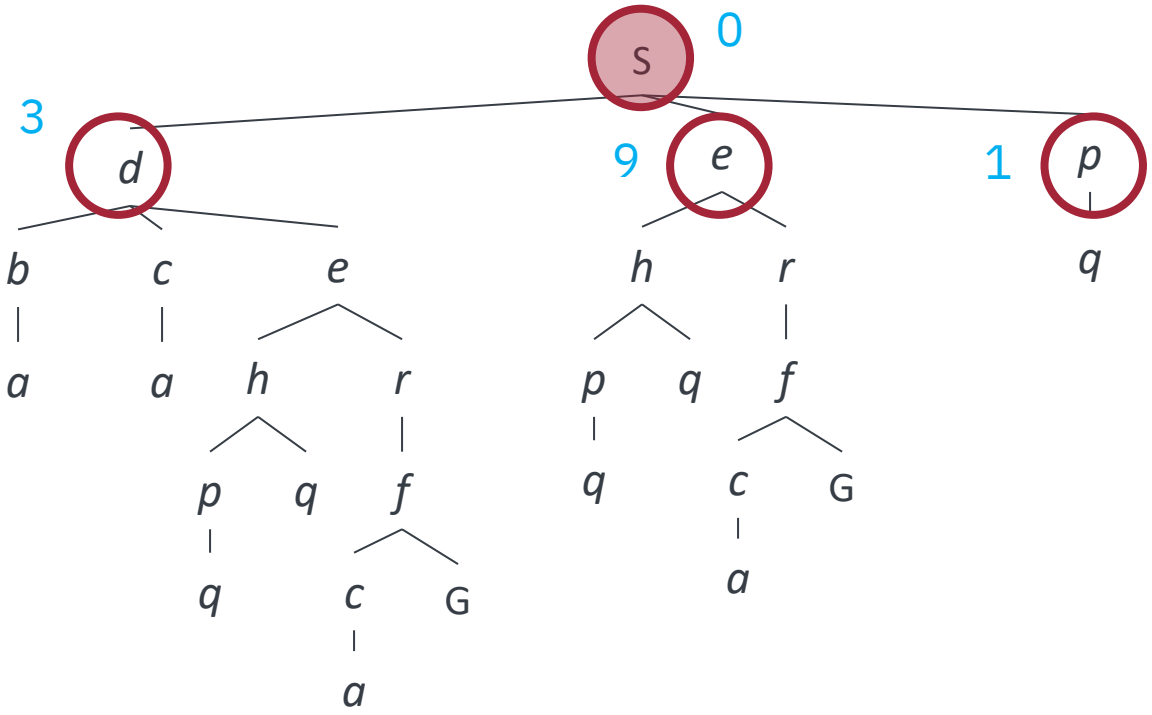


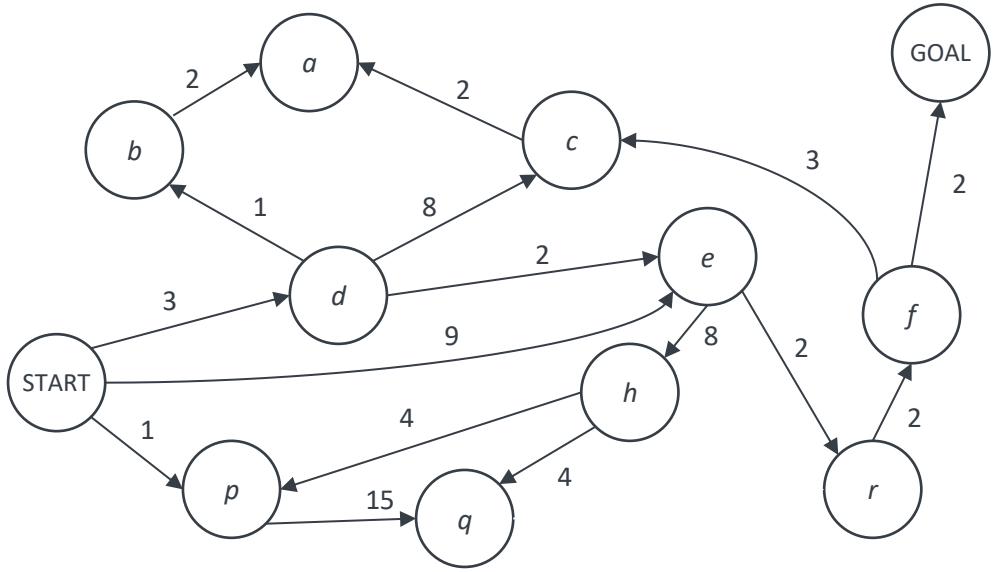
Cost-sensitive search



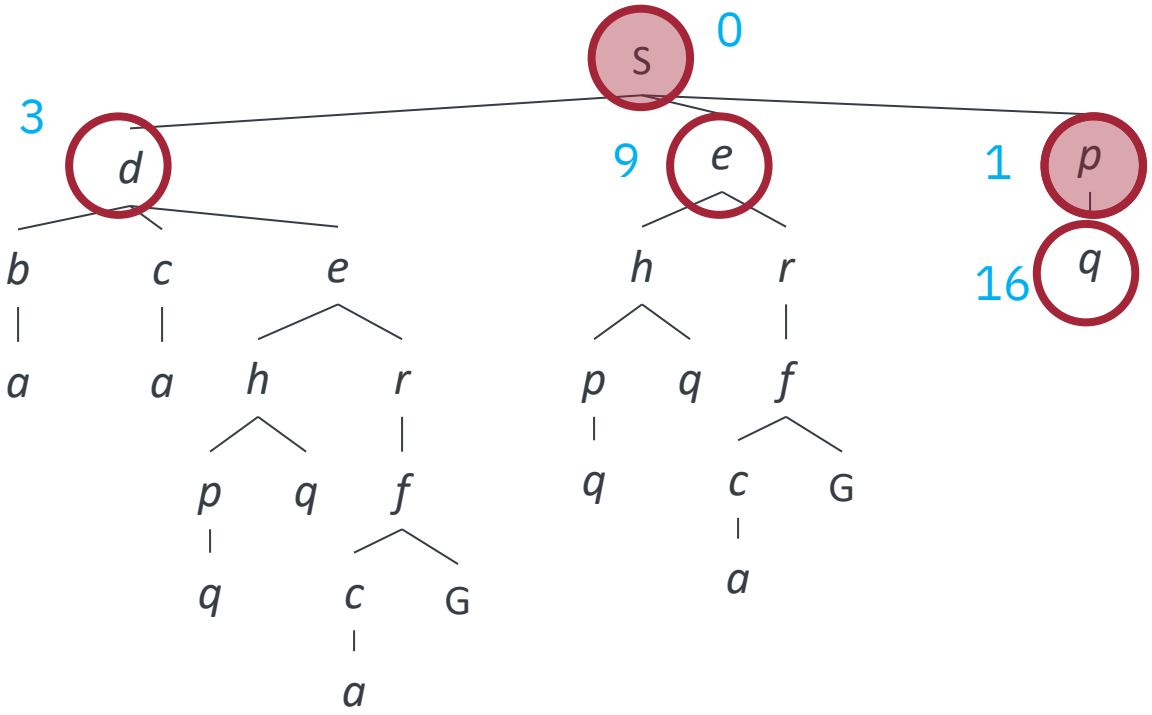
BFS finds the shortest path in terms of number of actions.
It does not find the least-cost path. We will now cover
a similar algorithm which does find the least-cost path.

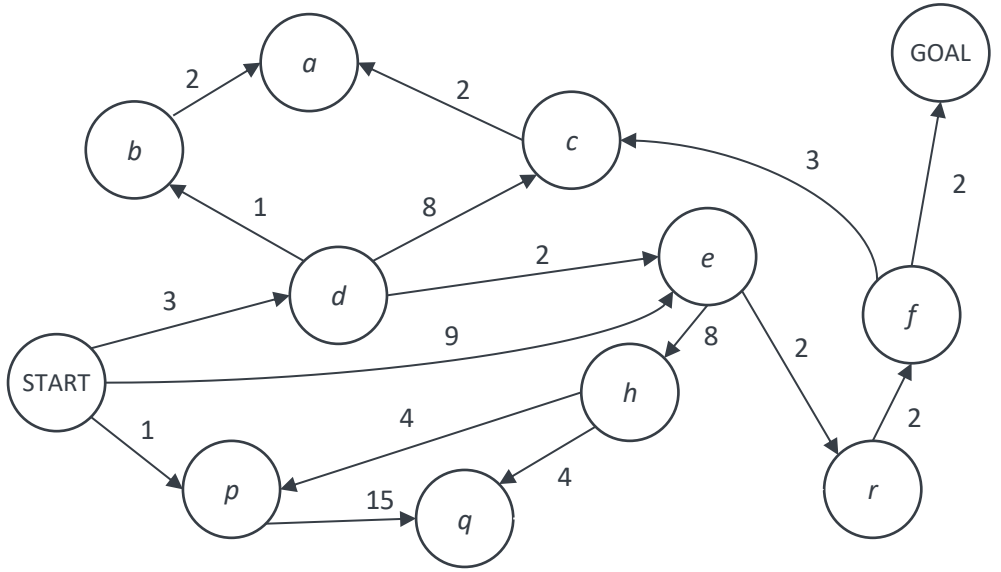
Uniform Cost Search (UCS)



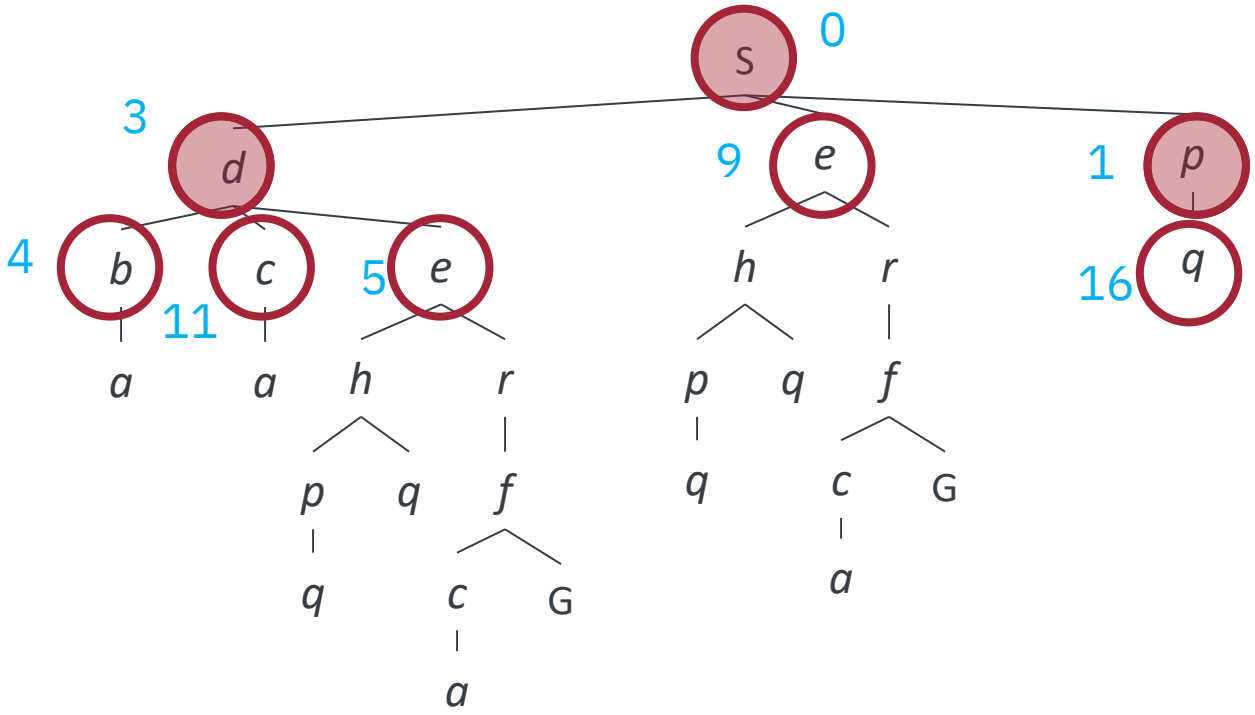


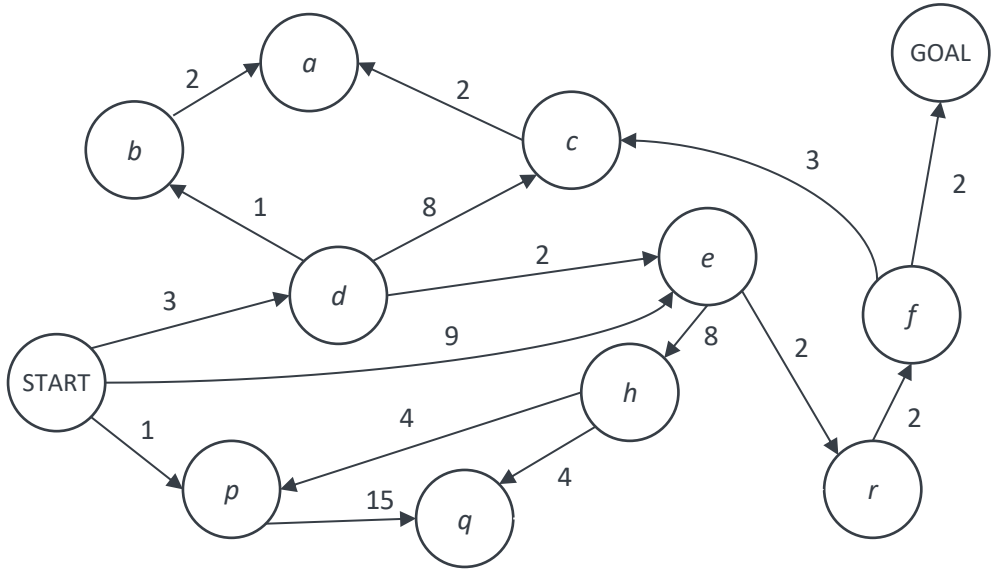
Uniform Cost Search (UCS)



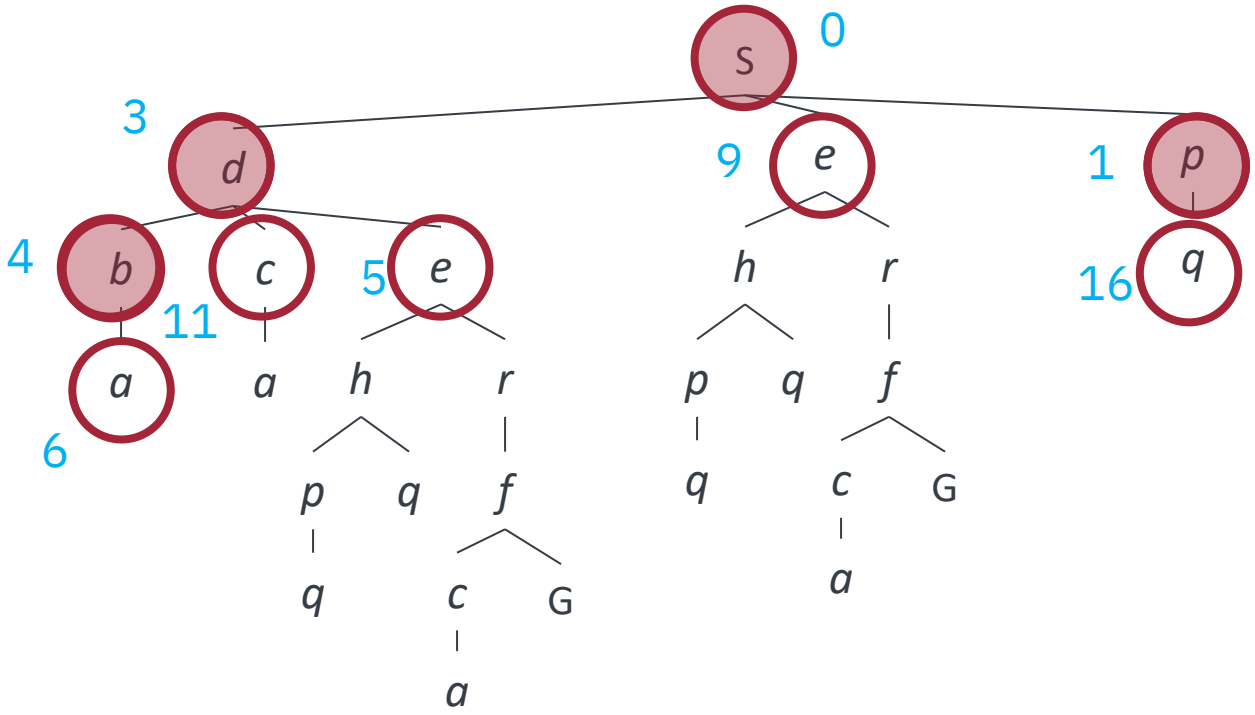


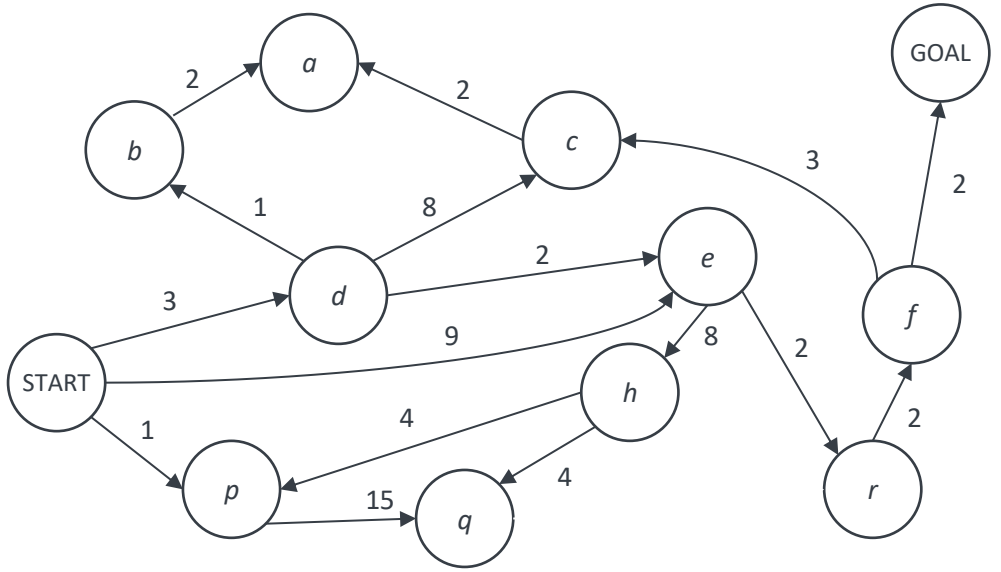
Uniform Cost Search (UCS)



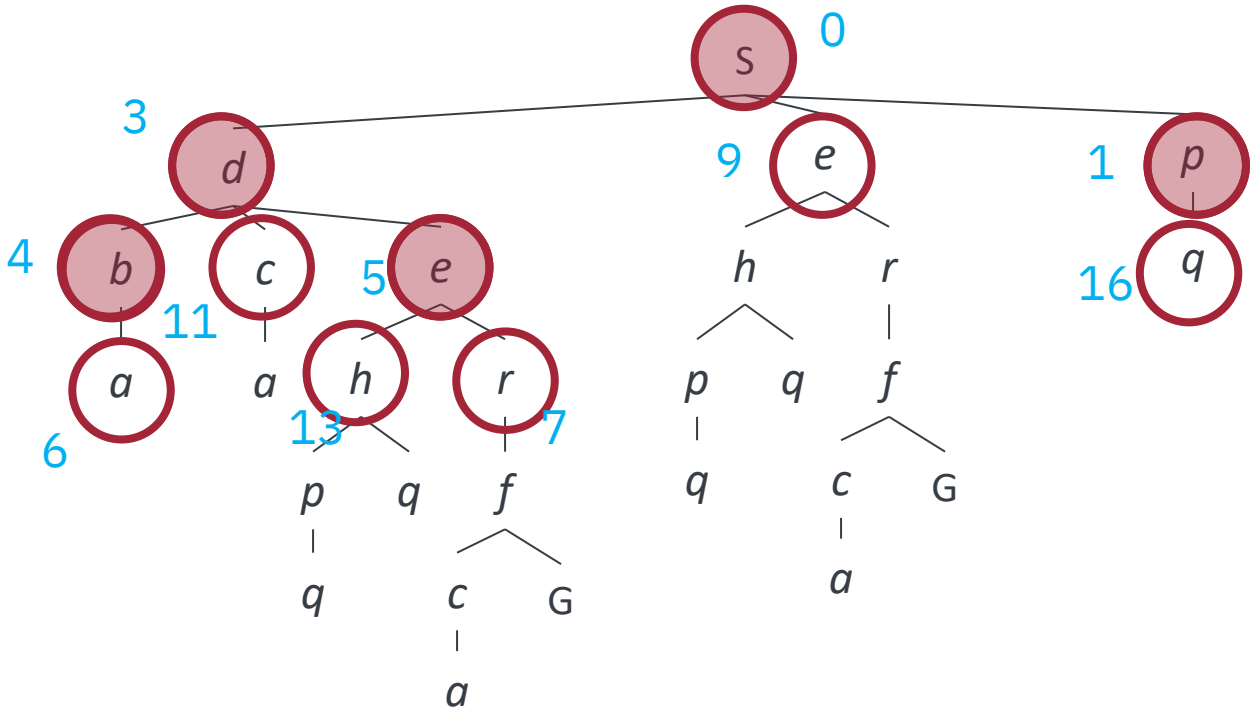


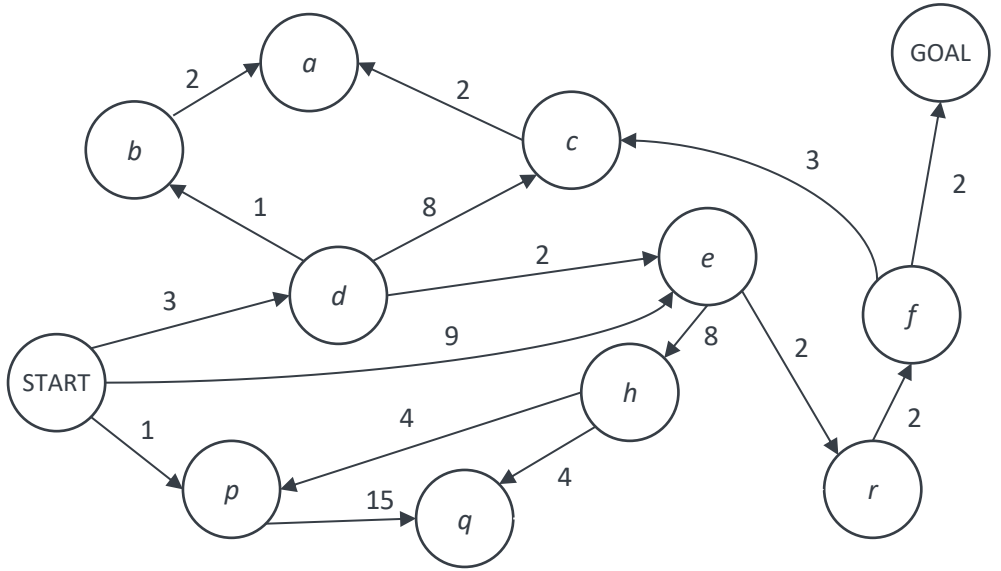
Uniform Cost Search (UCS)



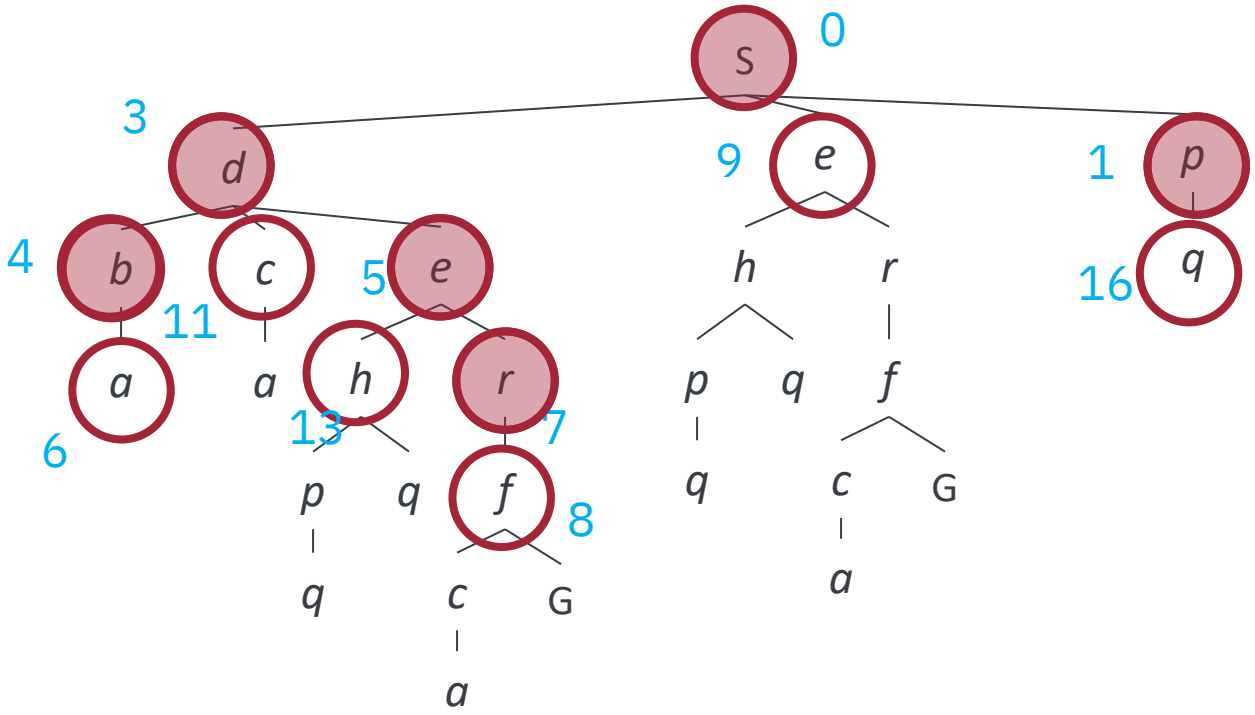


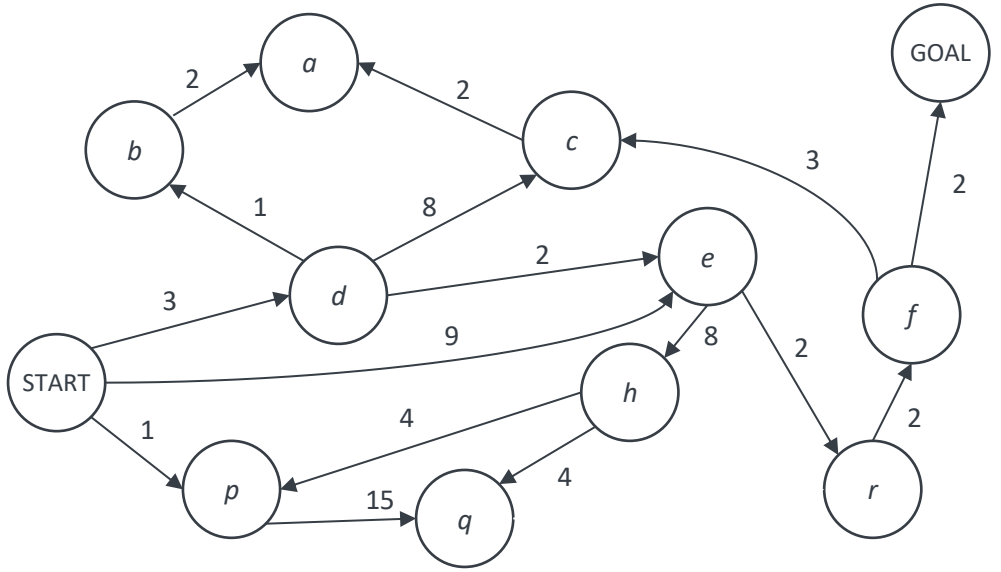
Uniform Cost Search (UCS)



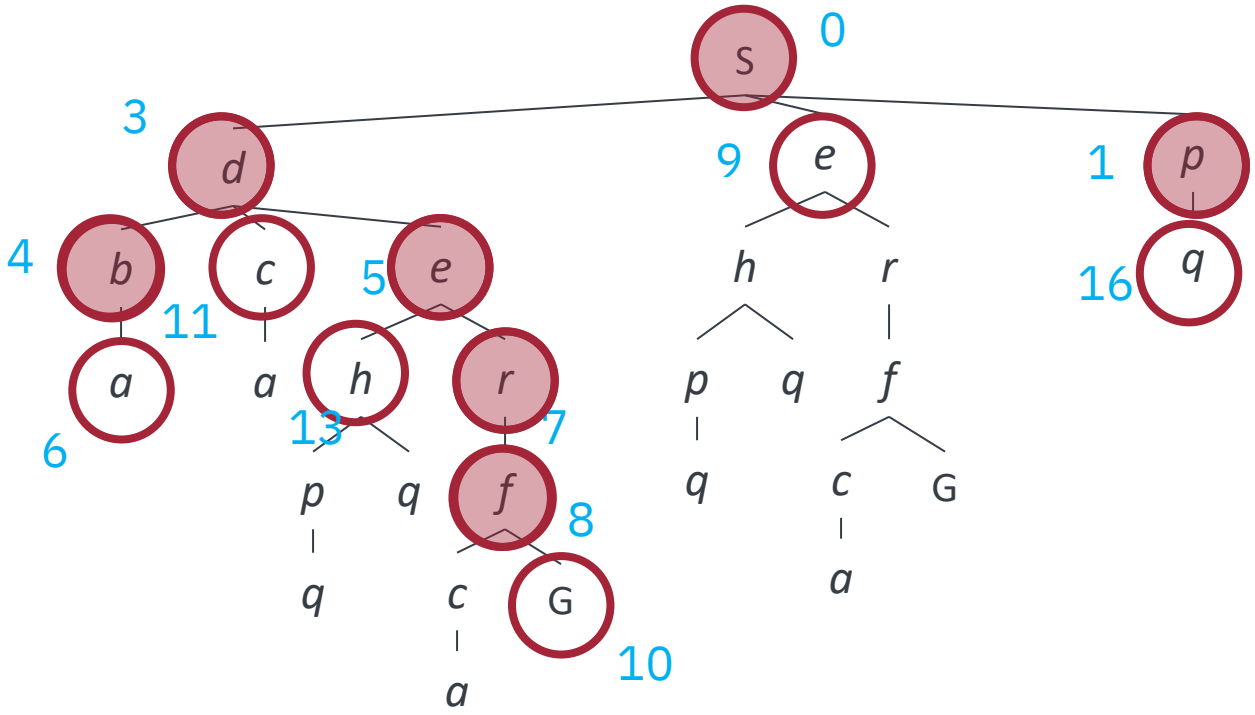


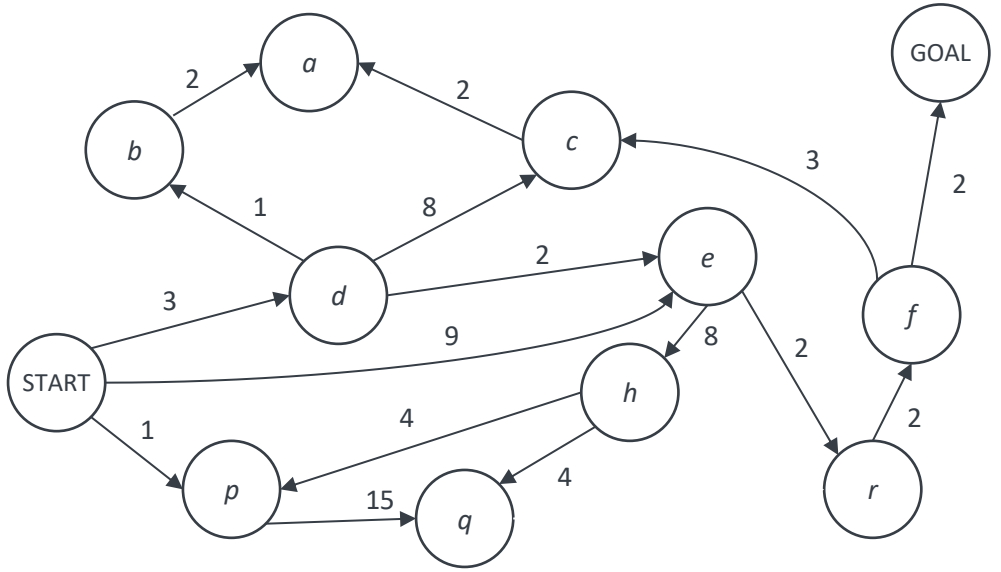
Uniform Cost Search (UCS)



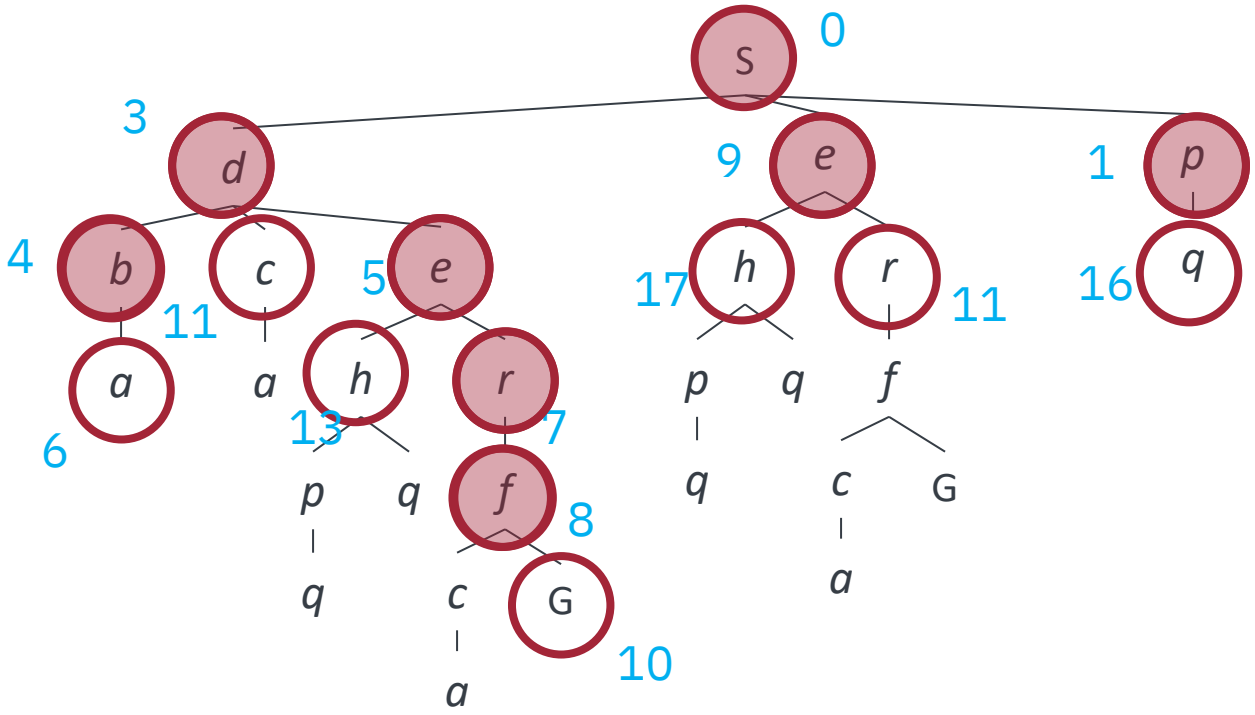


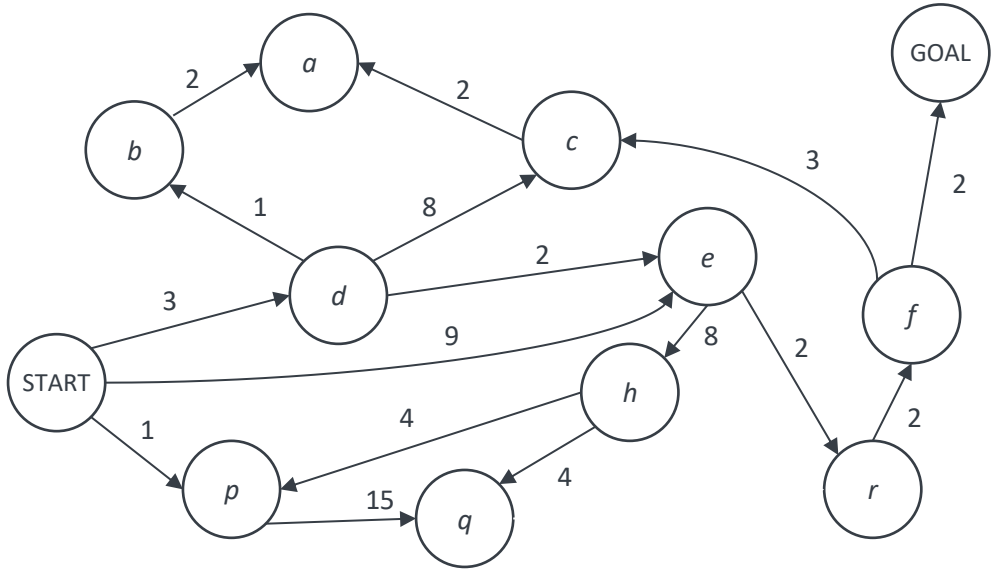
Uniform Cost Search (UCS)



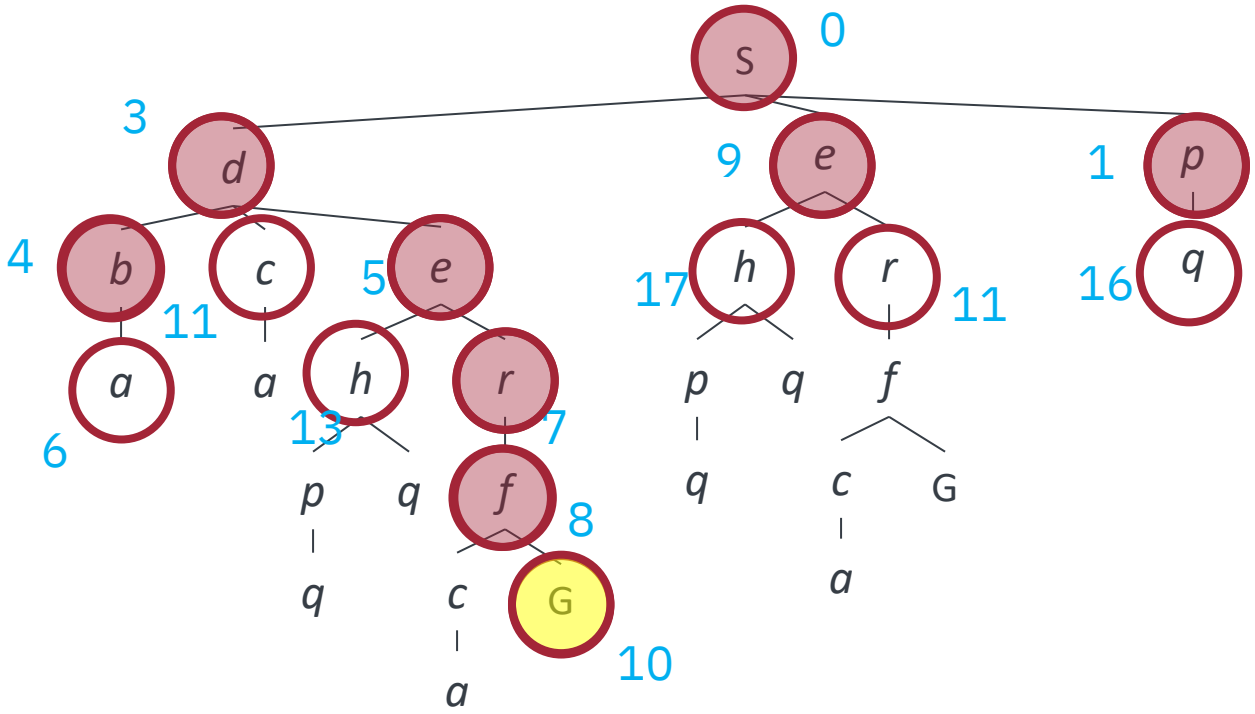


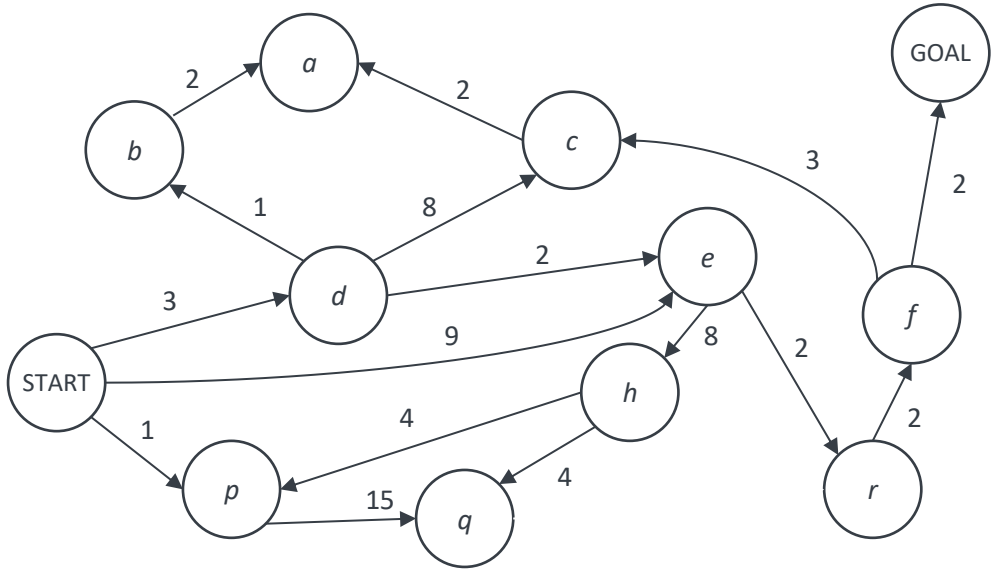
Uniform Cost Search (UCS)



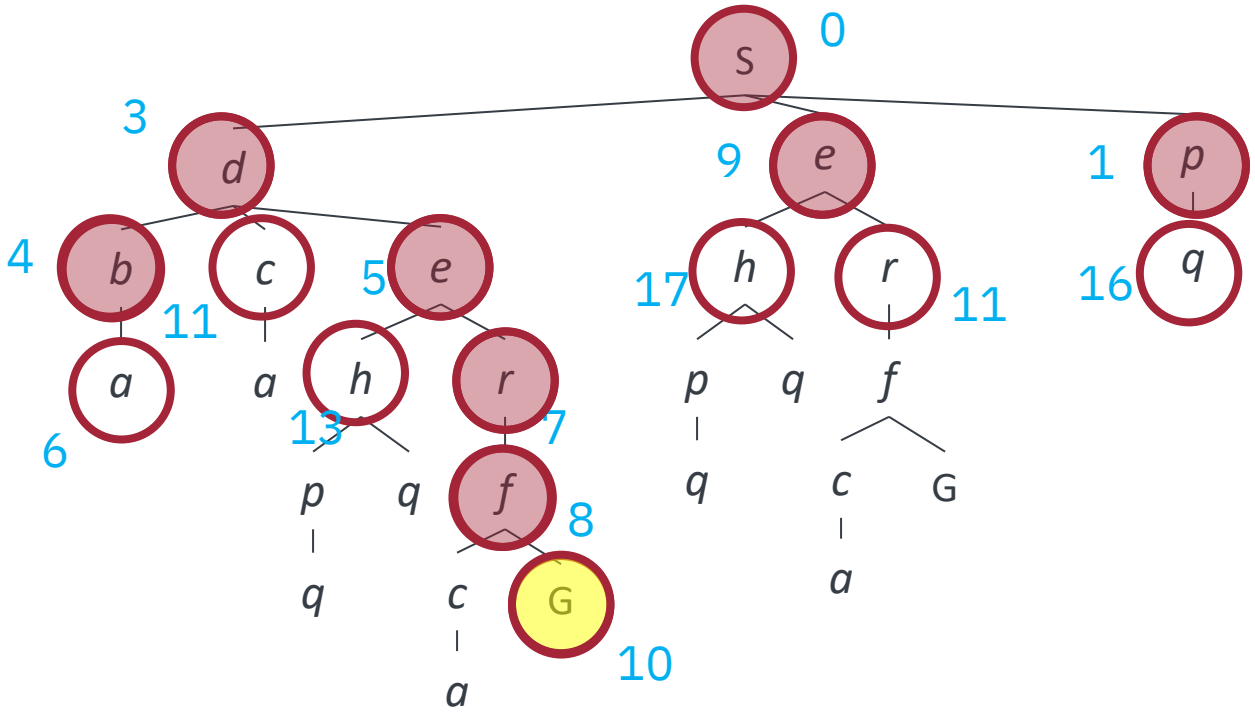


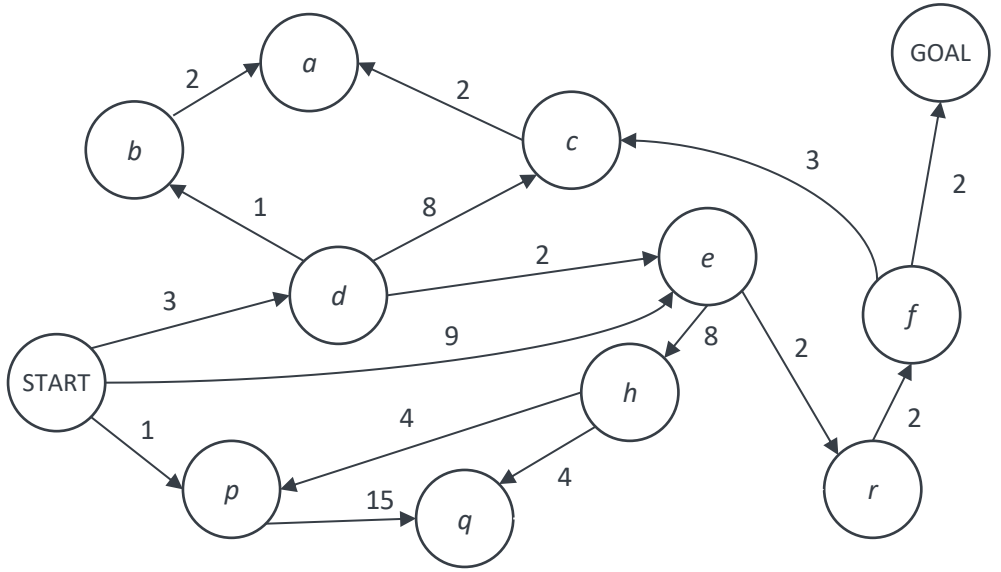
Uniform Cost Search (UCS)





Uniform Cost Search (UCS)

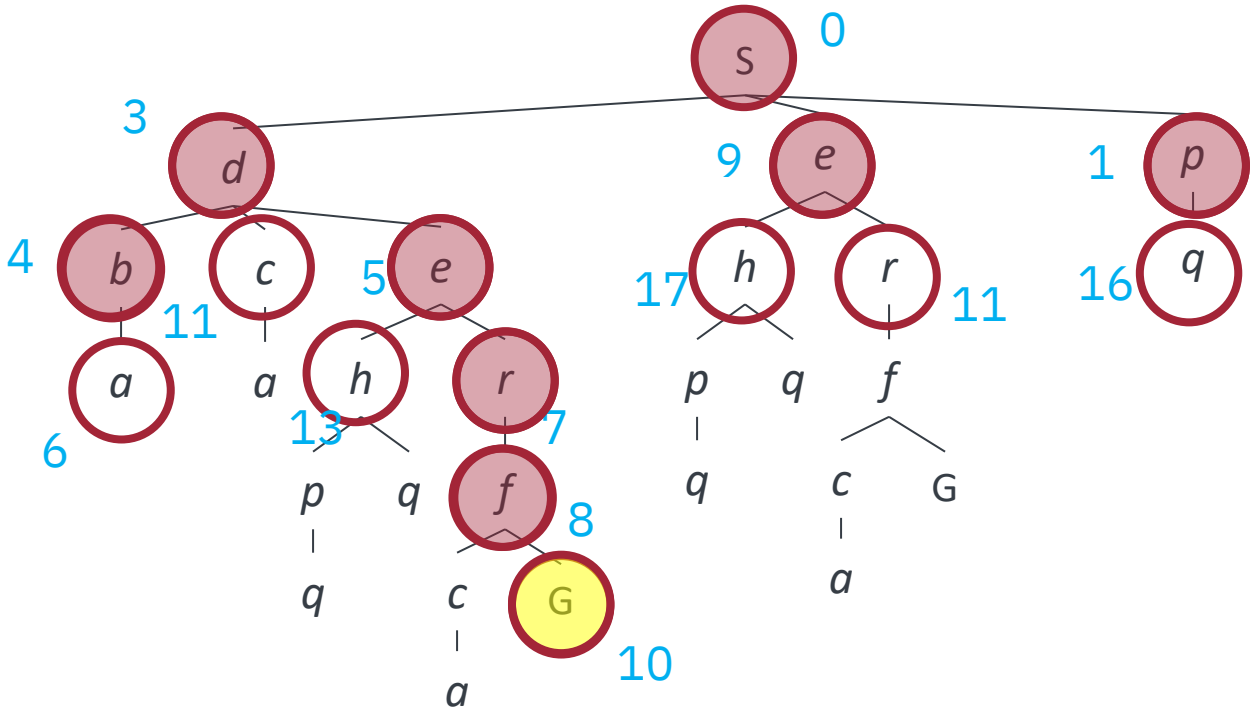


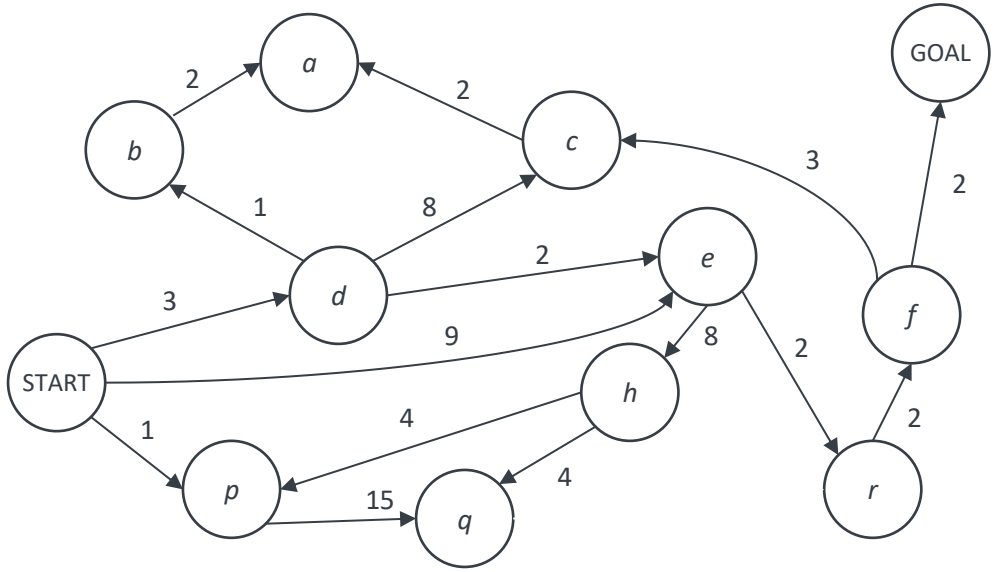


Note that the search ends when the goal node is **expanded** (not generated).

Why?

Uniform Cost Search (UCS)

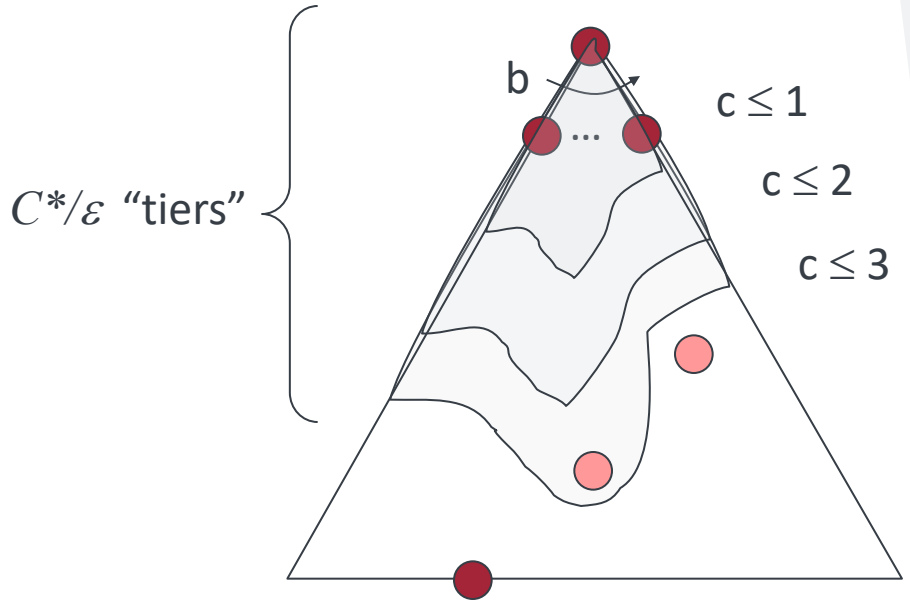




Number of expanded nodes: 9

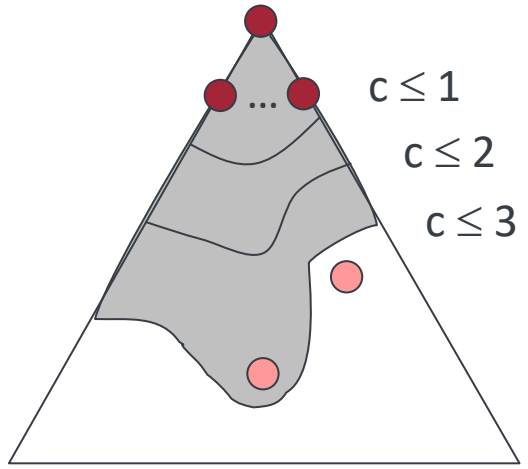
UCS properties

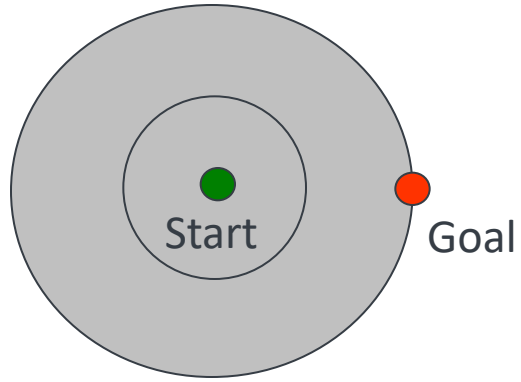
- What nodes does UCS expand?
  - Processes all nodes with cost less than cheapest solution!
  - If that solution costs C^* and arcs cost at least ε, then the "effective depth" is roughly C^*/ε
  - Takes time $O(b^{C^*/\varepsilon})$ (exponential in effective depth)
- How much space does the fringe take?
  - Has roughly the last tier, so $O(b^{C^*/\varepsilon})$
- Is it complete?
  - Assuming best solution has a finite cost and minimum arc cost is positive, yes!
- Is it optimal?
  - Yes! (Proof next lecture via A*)



UCS Issues

- Remember: UCS explores increasing cost contours
- The good: UCS is complete and optimal!
- The bad:
  - Explores options in every “direction”
  - No information about goal location
- We’ll fix that soon!





Summary

- All these search algorithms are the same except for the order in which they expand nodes.
  - Conceptually, all fringes are priority queues (*i.e.* collections of nodes with attached priorities)
  - Practically, for DFS and BFS, you can avoid the log(n) overhead from an actual priority queue, by using stacks and queues
  - Can even code one implementation that takes a variable queuing object

Attendance

- We use QR codes to save time.
  - Submit a google form through the QR code.

- How to scan a QR code?
  - Android
    - Built-in camera (Google lens)
    - https://play.google.com/store/apps/details?id=com.camvision.qrcode.barcode.reader&hl=en_US&gl=US&pli=1
  - iOS
    - Built-in camera
    - https://apps.apple.com/us/app/qr-code-reader/id1200318119

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Questions?

Chapter 1, 2, 3

Acknowledgement

Fahiem Bacchus, University of Toronto
Dan Klein, UC Berkeley
Kate Larson, University of Waterloo