Last time: Summary

- **Definition of AT?**
- **Turing Test?**
- **Intelligent Agents:**
 - Anything that can be sign mentaling its environment through sensors and acting upon that environment through its **effectors** to maximize progress towards its **goals**.
 - PAGE (Percepts, Actions, Goals, Environment)
 Described as a Perception (sequence) to Action Mapping: T: P* → A

 - Using look-up-table, closed form, etc.

WeChat: cstutorcs

- **Agent Types:** Reflex, state-based, goal-based, utility-based
- **Rational Action:** The action that maximizes the expected value of the performance measure given the percept sequence to date

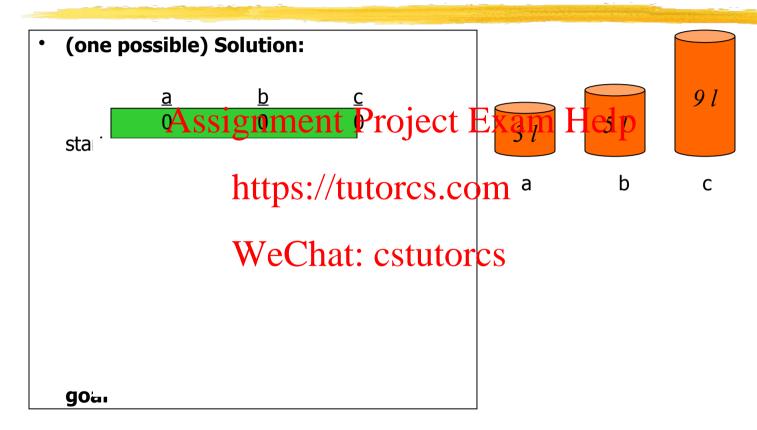
Outline: Problem solving and search

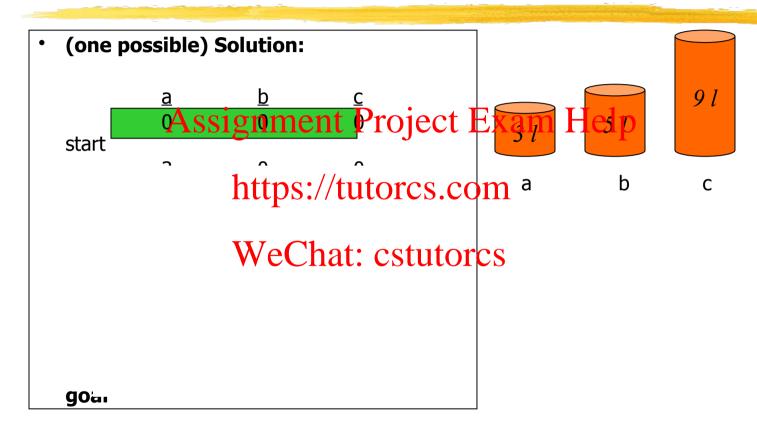
- Introduction to Problem Solving
- Complexity Assignment Project Exam Help
- Uninformed searchhttps://tutorcs.com
 - Problem formulation
 - Search strategies: depth-first, breadth-first CSTUTOTCS
- Informed search
 - Search strategies: best-first, A*
 - Heuristic functions

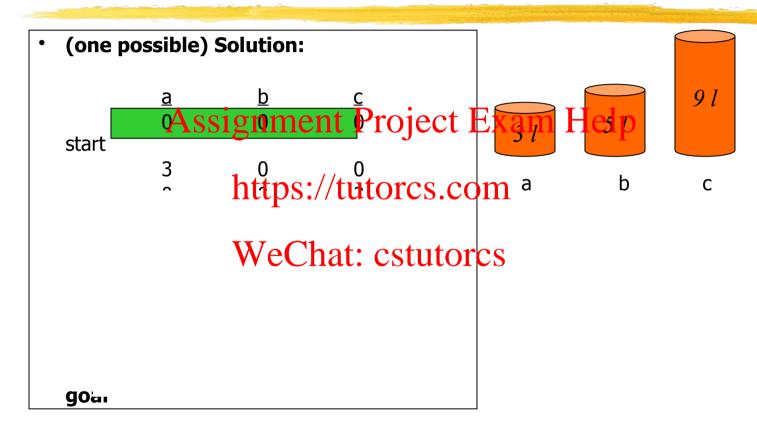


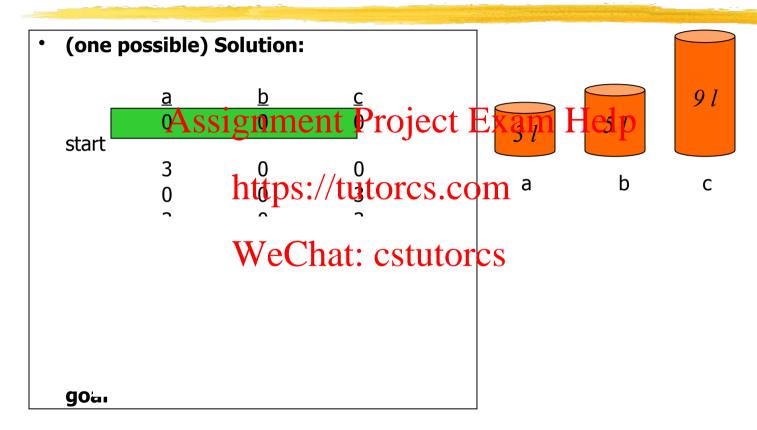
WeChat: cstutorcs

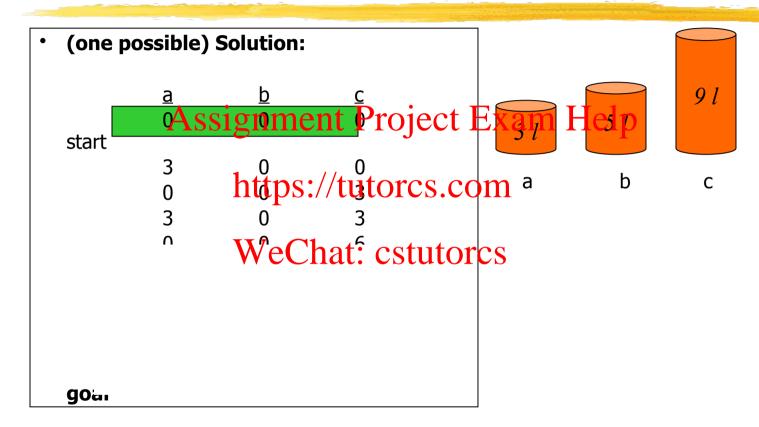
Problem: Using these three buckets, measure 7 liters of water.

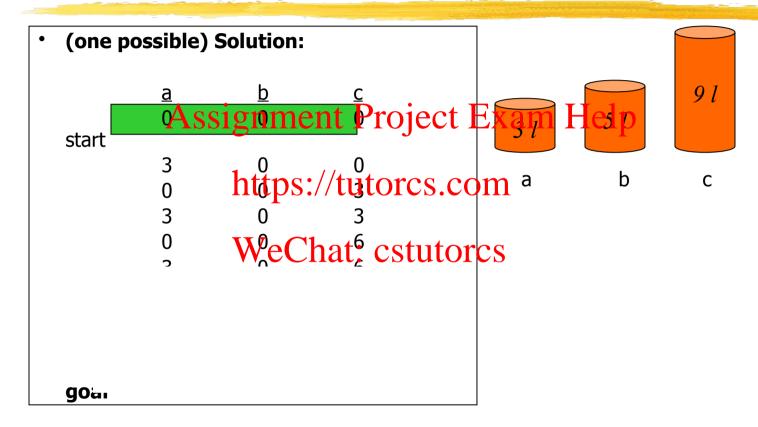


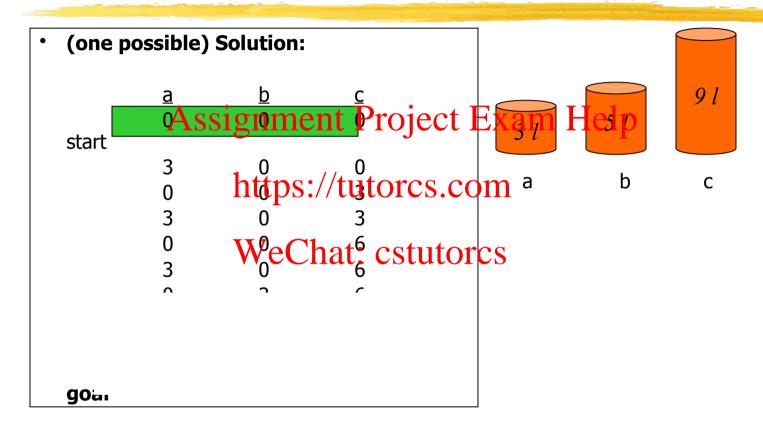


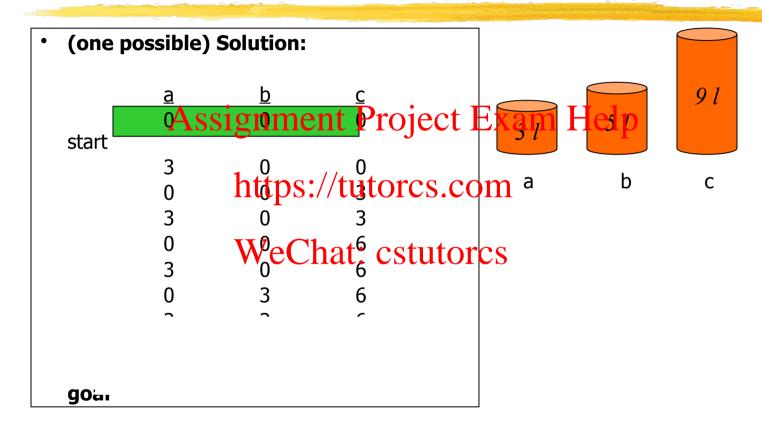


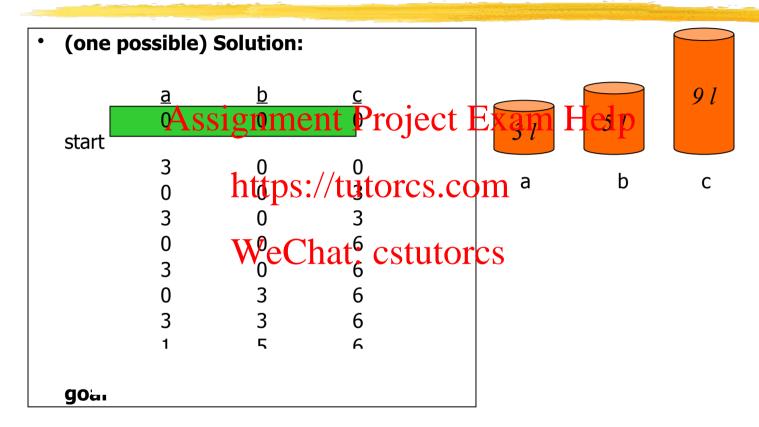


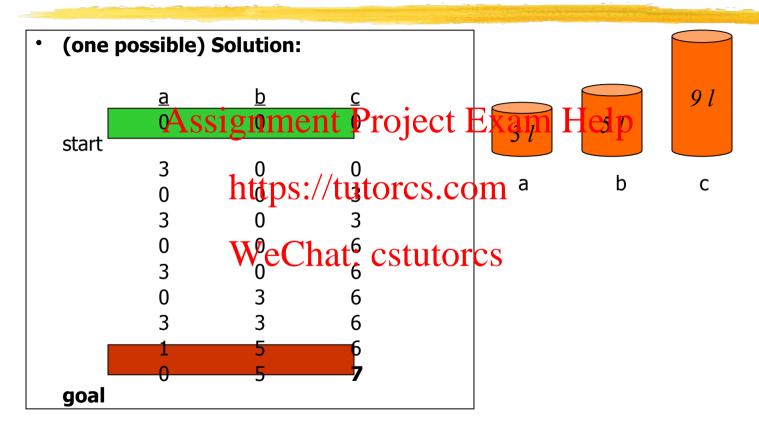


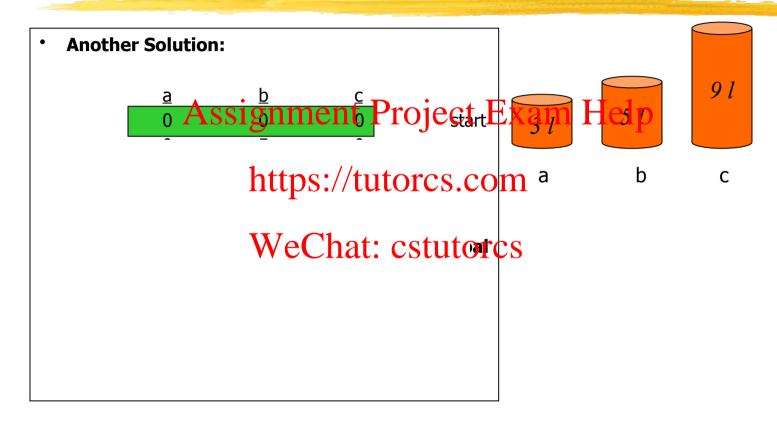


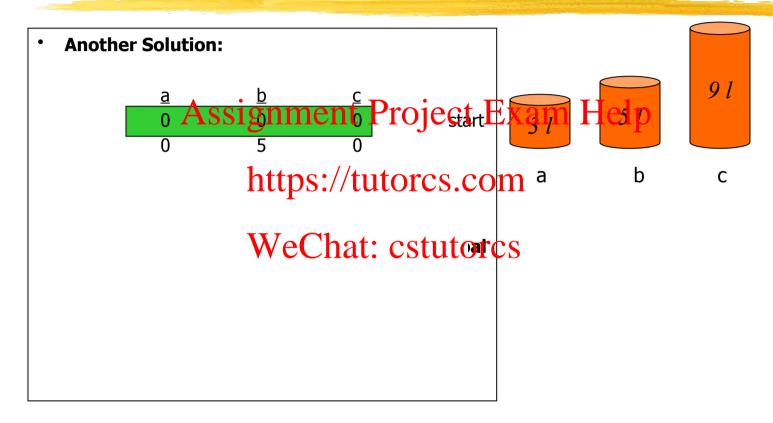


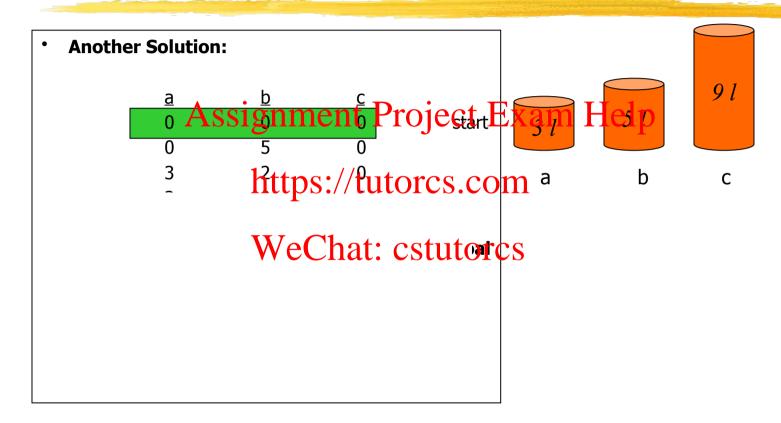


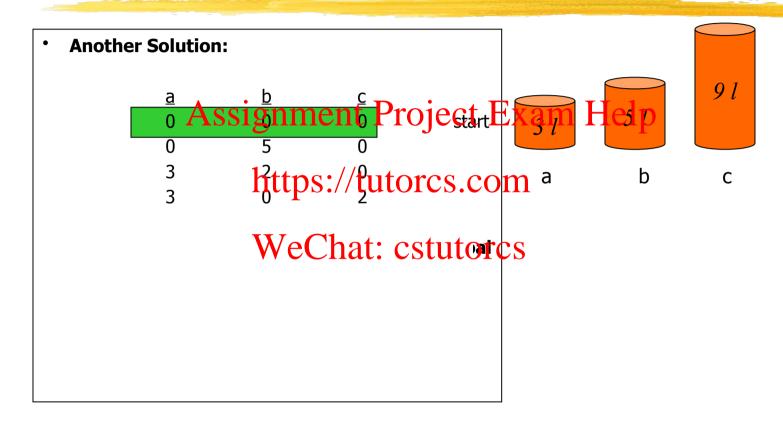


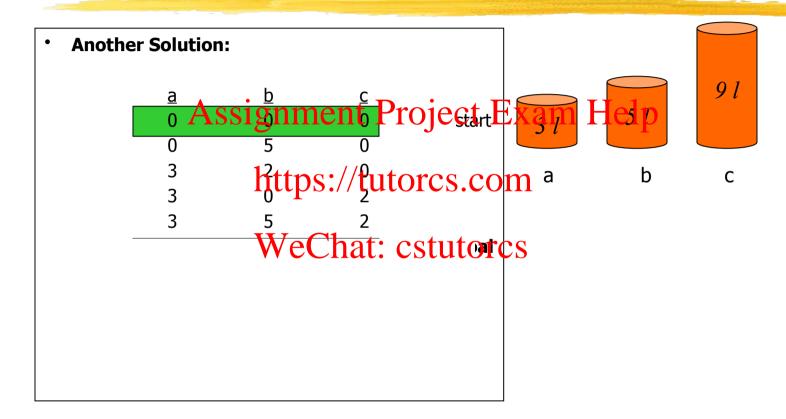


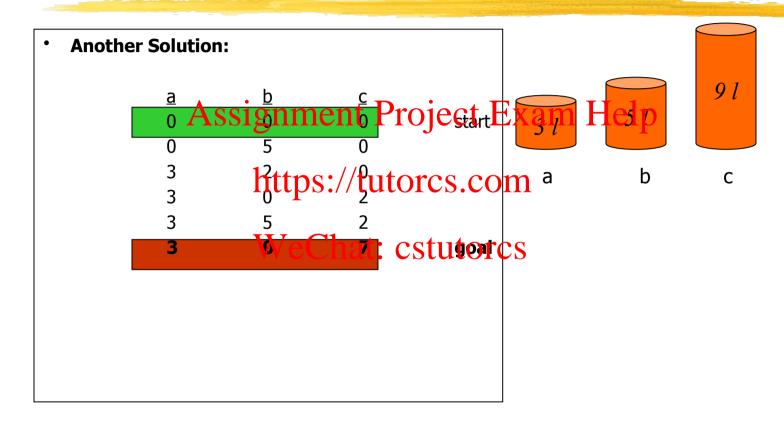




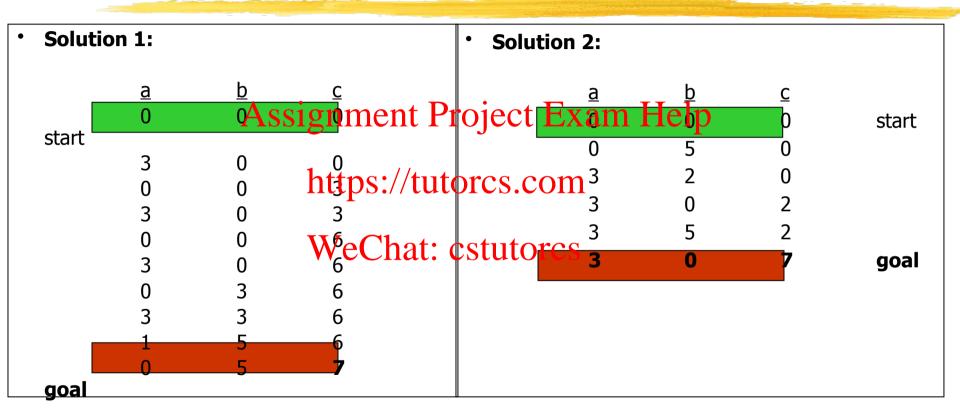








Which solution do we prefer?



Problem-Solving Agent

```
function SIMPLE-PROBLEM-SOLVING-AGENT(p) returns an action
   inputs: p, a percept
   static: s, an action sequence, initially empty
ASSI SOME TIPLE EXAMINATION FOR
              q, a goal, initially null
              problem formulation
   state \leftarrow \text{UPDATE-STATE}(state, p) // What is the current state?
   if s is empty then Chat: CSTUTOTCS
g \leftarrow \text{FORMULATE} \text{-} \text{COAL}(\text{state}) \text{ // From LA to San Diego (given curr. state)}
         problem \leftarrow FORMULATE-PROBLEM(state, g) // e.g., Gas usage
          s \leftarrow \text{Search}(problem)
    action \leftarrow \text{RECOMMENDATION}(s, state)
    s \leftarrow \text{Remainder}(s, state) // If fails to reach goal, update
   return action
```

Note: This is *offline* problem-solving. *Online* problem-solving involves acting w/o complete knowledge of the problem and environment

Example: Buckets

Measure 7 liters of water using a 3-liter, a 5-liter, and a 9-liter buckets.

· Formulate goalstiggenthiters to Pwatist it Biten by oktelp

• Formulate problemhttps://tutorcs.com

States: amount of water in the buckets

• Operators: WeChait! buskettfrom source, empty bucket

• **Find solution:** sequence of operators that bring you from current state to the goal state

Remember: Environment types

| Environment | Accessible | Deterministic | Episodic | Static | Discrete |
|-----------------------|------------|-----------------|----------|--------|------------------|
| Operating As | signme | iff Projec | d°Exar | n'Hel | p ^{Yes} |
| Virtual Reality | Yes | Yes Ytutorcs | Yes/No | No | Yes/No |
| Office Environment | No No | No | No | No | No |
| Mars | NoW eC | sati: cstu | orcs | Semi | No |

The environment types largely determine the agent design.

- * Single-state problem: deterministic, accessible

 Agent knows everything about world, thus can

 calculate optimal action sequence to reach goal state.

 ASSIGNMENT Project Exam Help
- Multiple-state problem: deterministic, inaccessible Agent must react position for states assumed while working towards goal state.
- Contingency problem: CStuttores inaccessible
 - Must use sensors during execution
 - Solution is a tree or policy
 - Often interleave search and execution
- **Exploration problem:** unknown state space *Discover and learn about environment while taking actions.*

Single-state problem:

deterministic, accessible

· Agent knows everything about world (the exact state),

https://tutorcs.com

• Can calculate optimal action sequence to reach goal state.

WeChat: cstutorcs

• E.g., playing chess. Any action will result in an exact state

- Multiple-state problem: deterministic, inaccessible
 - Agent does As Risw the exact spice (to be an argin the possible states)
 - May not have sensors at all https://tutorcs.com
 - Assume states while working towards goal state. WeChat: cstutorcs

- E.g., walking in a dark room
 - If you are at the door, going straight will lead you to the kitchen
 - If you are at the kitchen, turning left leads you to the bedroom

• ...

- Contingency problem: nondeterministic, inaccessible
 - · Must use sensosignment-Project Exam Help

 - Solution is a tree or policy
 Often interleave states of the contraction of the contraction

WeChat: cstutores

- E.g., a new skater in an arena
 - Sliding problem.
 - Many skaters around

• **Exploration problem:** unknown state space

Assignment Project Exam Help Discover and learn about environment while taking actions.

https://tutorcs.com

WeChat: cstutorcs

• E.g., Maze

Example: Vacuum world

Simplified world: 2 locations, each may or not contain dirt,

each may or not contain vacuuming agent.

Goal of agent: Alean un the dirt Project Exam Help

Single-state, start in #5. Solution??

Multiple-state, start in {1, 2, 3, 4, 5, 6, 7, 8}
e.g., Right goes W2Chat: Solution??
Contingency, start in #5
Murphy's Law: Suck can dirty a clean car- 5 pet
Local sensing: dirt, location only.
Solution??
8
Solution??







CS 561, Sessions 2-3



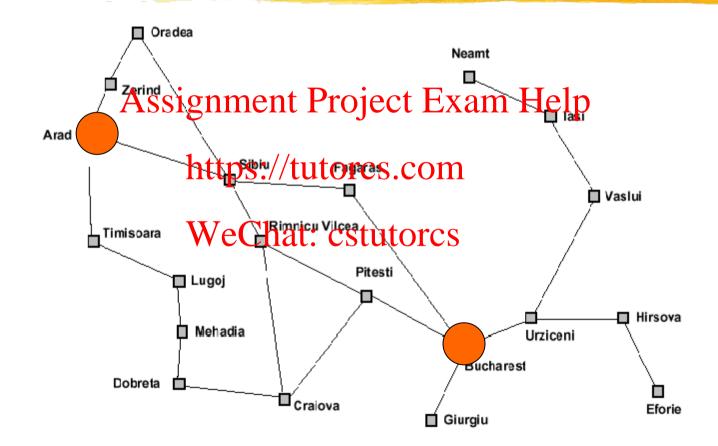
Example: Romania

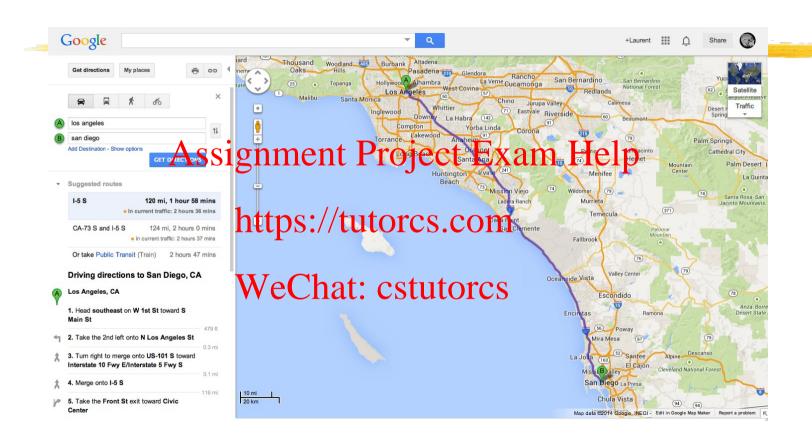
- In Romania, on vacation. Currently in Arad.
- Flight leaves tomorrow from Bucharest.
- Formulate goal ssignment Project Exam Help
 - ➤ be in Bucharest

https://tutorcs.com

- Formulate problem:
 - > states: various cities eChat: cstutorcs
 - > operators: drive between cities
- Find solution:
 - > sequence of cities, such that total driving distance is minimized.

Example: Traveling from Arad To Bucharest





Problem formulation

A *problem* is defined by four items: <u>initial state</u> e.g., "at Arad" Assignment Project Exam Help e.g., Arad \rightarrow Zerind Arad \rightarrow Sibiu https://tutorcs.com etc. goal test, can be explicit e.g., NoDirt(x) path cost (additive)

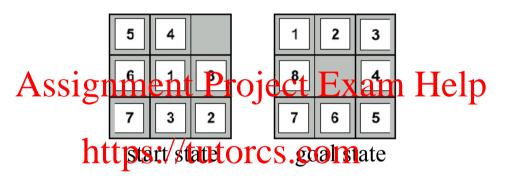
e.g., sum of distances, number of operators executed, etc.

A *solution* is a sequence of operators leading from the initial state to a goal state

Selecting a state space

- Real world is absurdly complex; some abstraction is necessary to allow us to reason on it...
- Selecting the correct abstraction and resulting state space is a difficult problem!
- Abstract states https://tutorcs.com
- Abstract operators \Leftrightarrow We Chatter Cest Lip Gord actions (e.g., going from city i to city j costs Lij \Leftrightarrow actually drive from city i to j)
- Abstract solution
 set of real actions to take in the real world such as to solve problem

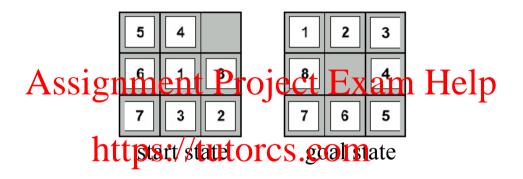
Example: 8-puzzle



- State: WeChat: cstutorcs
- Operators:
- Goal test:
- Path cost:



Example: 8-puzzle



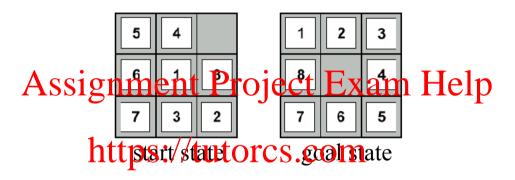
• State: integer location of tiles (ignore intermediate locations)

• Operators: moving blank left, right, up, down (ignore jamming)

Goal test: does state match goal state?

• Path cost: 1 per move

Example: 8-puzzle

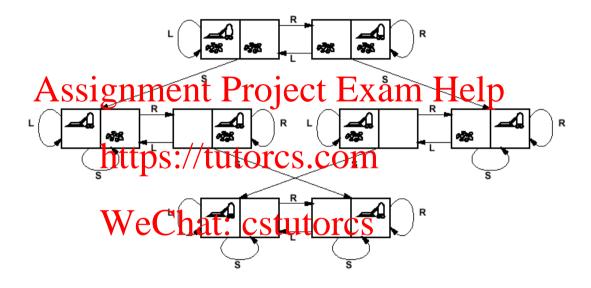


- Why search algorithms?

 8-puzzle has 362,880 states tutores
 - 15-puzzle has 10^12 states
 - 24-puzzle has 10²⁵ states

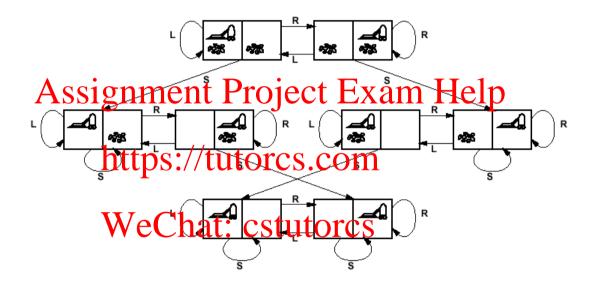
So, we need a principled way to look for a solution in these huge search spaces...

Back to Vacuum World



states??
operators??
goal test??
path cost??

Back to Vacuum World



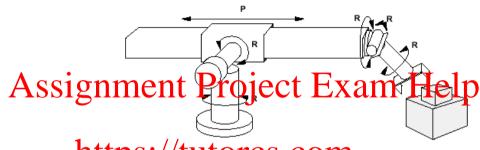
<u>states</u>??: integer dirt and robot locations (ignore dirt *amounts*)

operators??: Left, Right, Suck

goal test??: no dirt

path cost??: 1 per operator

Example: Robotic Assembly



https://tutorcs.com

states??: real-valued coordinates of

parts of the object to be assembled

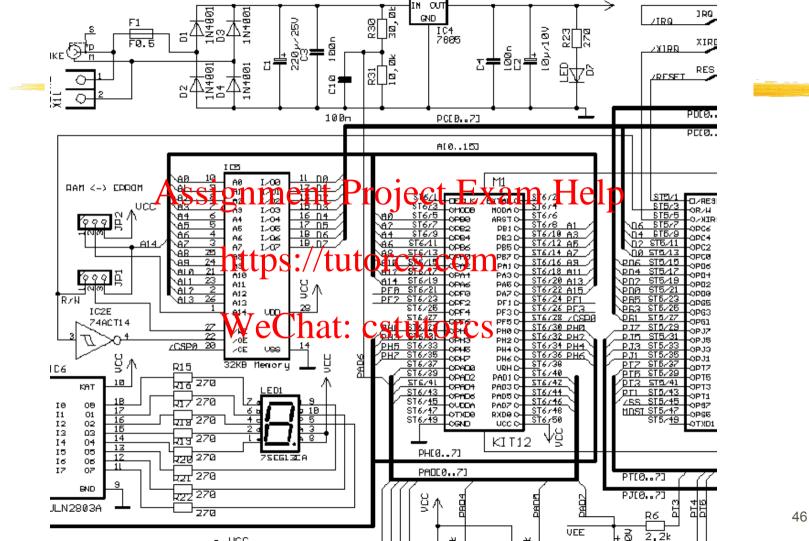
operators??: continuous motions of robot joints

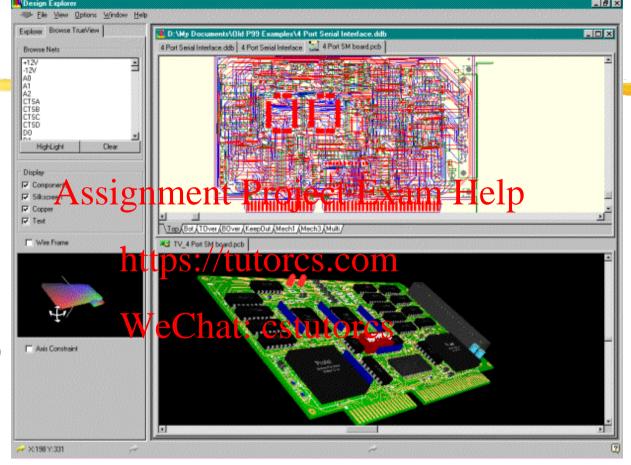
goal test??: complete assembly with no robot included!

path cost??: time to execute

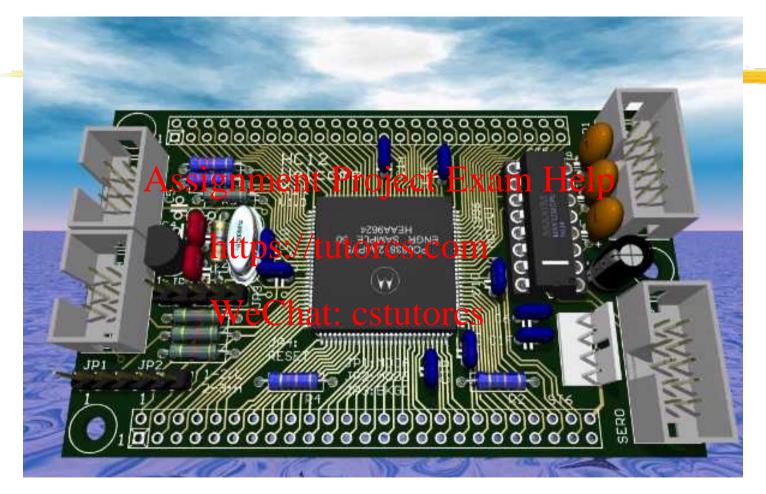
Real-life example: Circuit Board Layout

- Given schematic diagram comprising components (chips, resistors, capacitors, etc) and interconnections (wires), find optimal way to place components on a printed schematic under the lateral points as small number of wire layers are available (and wires on a given layer cannot cross!)
- "optimal way"?? WeChat: cstutorcs
- minimize surface area
- minimize number of signal layers
- > minimize number of vias (connections from one layer to another)
- minimize length of some signal lines (e.g., clock line)
- distribute heat throughout board
- > etc.





Protel 99 SE's unique 3D visualization feature lets you see your finished board before it leaves your desktop. Sophisticated 3D modeling and extrusion techniques render your board in stunning 3D without the need for additional height information. Rotate and zoom to examine every aspect of your board.



Search algorithms

Basic idea:

offline, systematic exploration of simulated state-space by generating signatures of Exported states (explanding)

Function General-Search (problem, strategy) returns a solution, or failure initialize the search tree using the initial state problem WeChat: cstutorcs

loop do

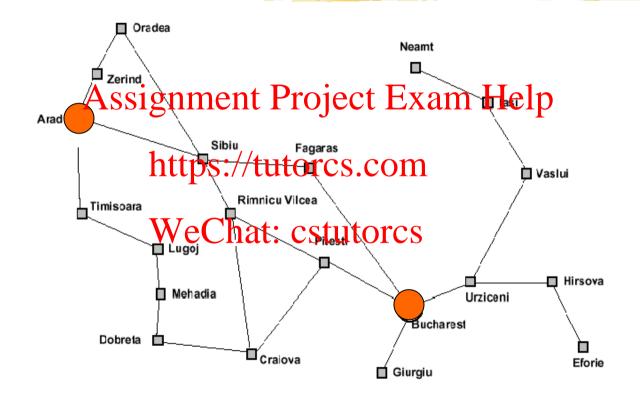
if there are no candidates for expansion then return failure choose a leaf node for expansion according to strategy if the node contains a goal state then **return** the corresponding solution **else** expand the node and add resulting nodes to the search tree

end

Example: micromouse in a maze



Example: Traveling from Arad To Bucharest

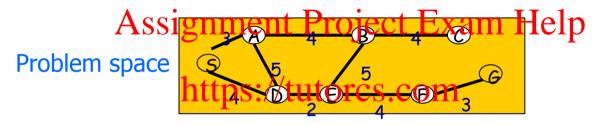


From problem space to search tree

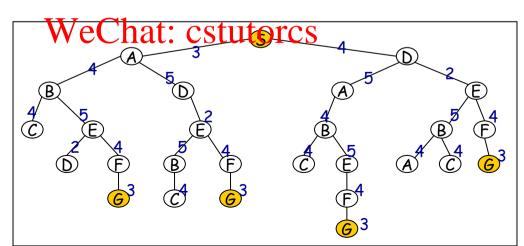
Some material in this and following slides is from

http://www.cs.kuleuven.ac.be/~dannyd/FAI/

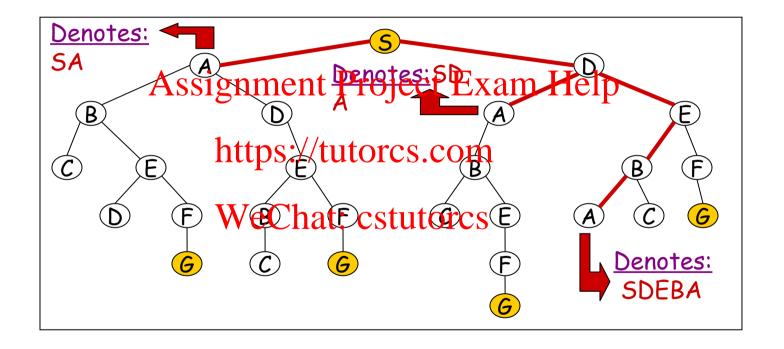
check it out!



Associated loop-free search tree



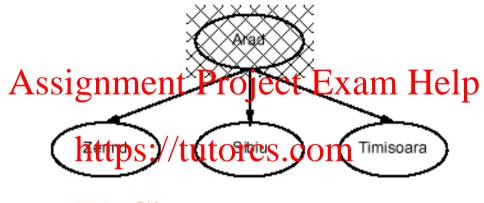
Paths in search trees



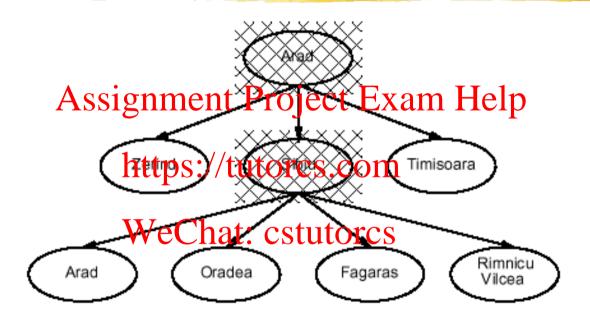
Assignment Project Exam Help

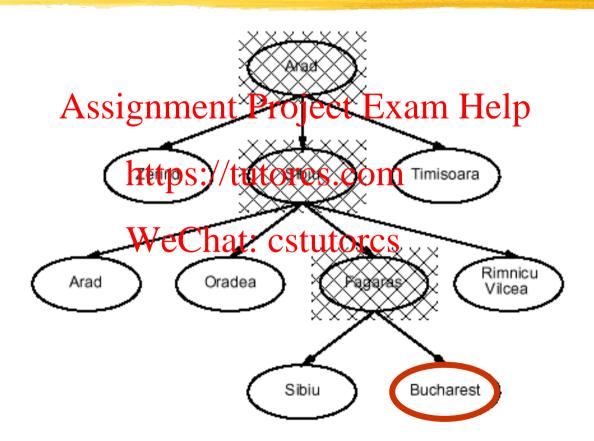
https://tutorcs.com

WeChat: cstutorcs



WeChat: cstutorcs





Implementation of search algorithms

```
Function General-Search(problem, Queuing-Fn) returns a solution, or failure

nodes ← make-queue(make-node(initial-state[problem]))

loop doAssignment Project Exam Help

if nodes is empty then return failure

node ← Remove-Front(nodes)

if Goattestproblem1000estcostmode) succeeds then return node

nodes ← Queuing-Fn(nodes, Expand(node, Operators[problem]))

end

WeChat: cstutorcs
```

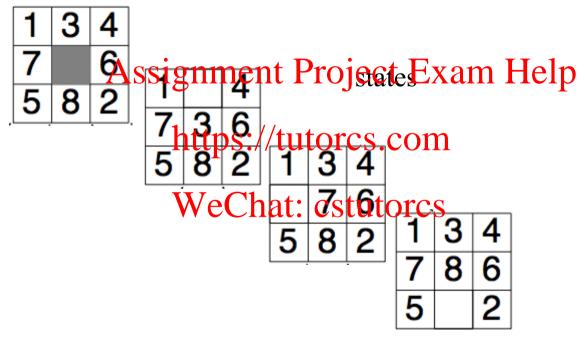
Queuing-Fn(queue, elements) is a queuing function that inserts a set of elements into the queue and <u>determines the order of node expansion</u>. Varieties of the queuing function produce varieties of the search algorithm.

Encapsulating state information in nodes

A state is a (representation of) a physical configuration A node is a data structure constituting part of a search tree includes parent, children, depth, path cost g(r)
ASSIGNMENT Project Example
States do not have parents, children, depth, or path co parent https://tutorcs.com depth = 6State Node cstutores g = 6state children

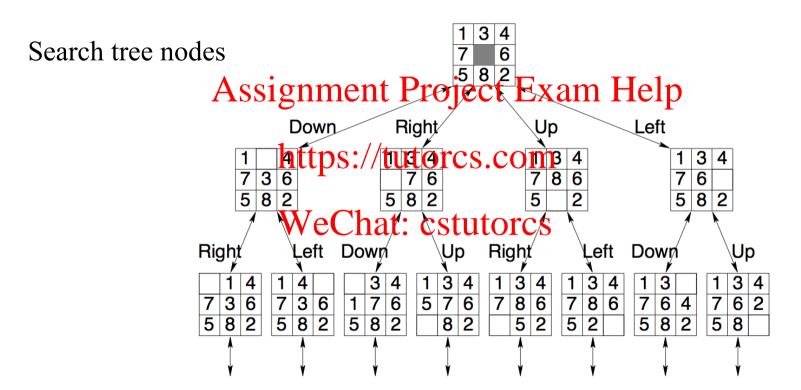
The EXPAND function creates new nodes, filling in the various fields and using the OPERATORS (or SUCCESSORFN) of the problem to create the corresponding states.

Paths in search trees



. . .

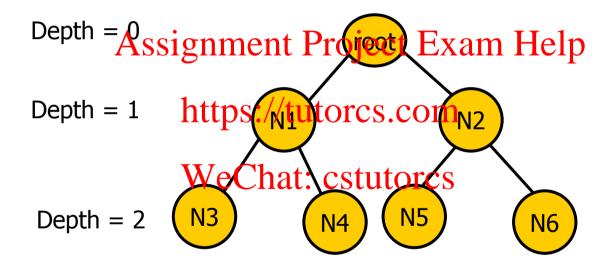
Paths in search trees



Evaluation of search strategies

- A search strategy is defined by picking the order of node expansion.
- Search algorithms are commonly evaluated according to the following four criteria:
 - Completeness to sit always find a count if one exists?
 - Time complexity: how long does it take as function of num. of nodes?
 - Space complexity: How much mention does it require?
 - Optimality: does it guarantee the least-cost solution?
- Time and space complexity are measured in terms of:
 - *b* − max branching factor of the search tree
 - *d* − depth of the least-cost solution
 - $m \max$ depth of the search tree (may be infinity)

Binary Tree Example



Number of nodes at max depth: $n = 2^{max depth}$ Number of levels (given n at max depth) = log2(n)

Complexity

- Why worry about complexity of algorithms?
- > because a problem in practice

https://tutorcs.com

WeChat: cstutorcs

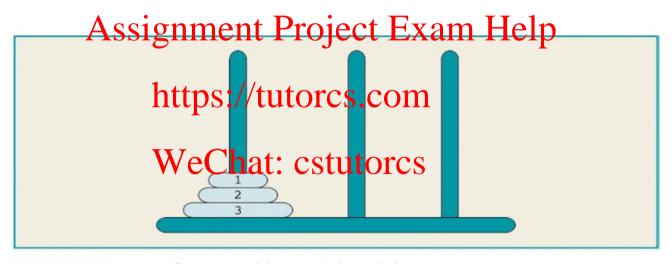


Figure 11-6 Tower of Hanoi problem with three disks

Complexi Tower of I

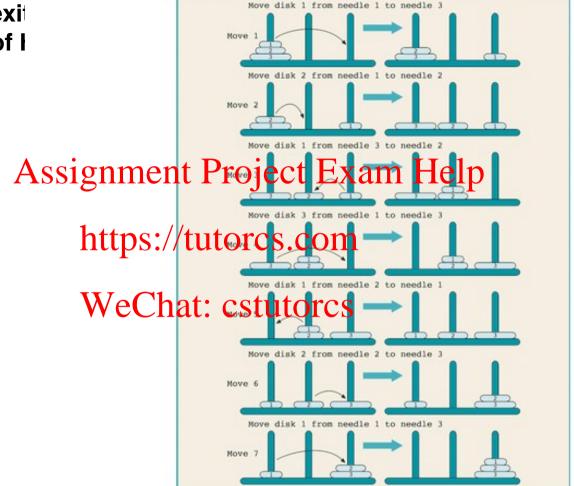


Figure 11-7 Solution of Tower of Hanoi problem with three disks

- 3-disk problem: $2^3 1 = 7$ moves Assignment Project Exam Help
- 64-disk problem: $2^{64} 1$. //tutorcs.com $2^{10} = 1024 \approx 1000 = 10^{3}$,

 - $2^{64} = 2^4 * 2^{60} \approx Wether = cstutores$
- One year $\approx 3.2 \times 10^7$ seconds

The wizard's speed = one disk / second
 Assignment Project Exam Help

1.6 *
$$10^{19} = 5$$
 * 3.2 * $10^{18} = \frac{\text{https://tutorcs.com}}{\text{tutorcs.com}}$
5 * $(3.2 * 10^{7})$ * $10^{17} = \frac{1}{2}$

500 billion years

 The time required to move all 64 disks from needle 1 to needle 3 is roughly 5 * 10¹¹ years.

Assignment Project Exam Help

• It is estimated that our universe is about 15 billion = 1.5 * 10¹⁰ years old. https://tutorcs.com

$$5 * 10^{11} = 50 * 10^{10} \approx 33 * (1.5 * 10^{10}).$$

- Assume: a computer with 1 billion = 10° moves/second.

 Assignment Project Exam Help

 Moves/year=(3.2 *10°) * 10° = 3.2 * 10¹6

https://tutorcs.com

To solve the problem for 64 disks:

•
$$2^{64} \approx 1.6$$
 We Chat: c.stutoros * $10^3 = (3.2 * 10^{16}) * 500$

• 500 years for the computer to generate 2⁶⁴ moves at the rate of 1 billion moves per second.

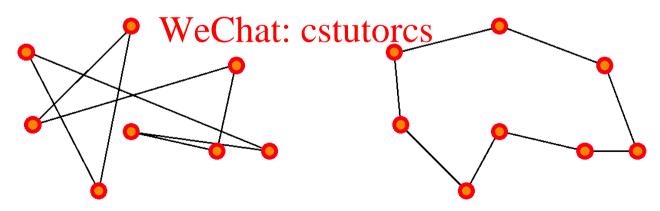
Complexity

- Why worry about complexity of algorithms?
- because a problem may be solvable in principle but may take too long to solve in principle but may take too long to solve in principle but may take too
- How can we evaluatepts://tortoresitoomalgorithms?
- Through asymptotic analysis, i.e., estimate time (or number of operations) necessary to solive at lineating the control of a problem when *n* tends towards infinity
- ➤ See AIMA, Appendix A.

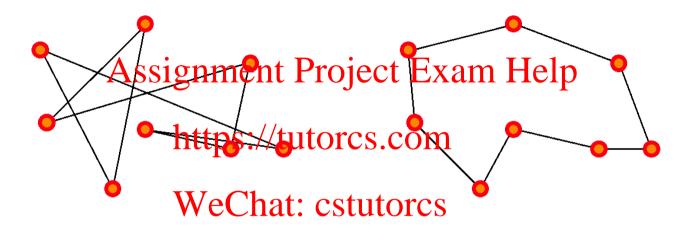
Complexity example: Traveling Salesman Problem

- There are n cities, with a road of length L_{ij} joining city i to city j.
- The salesgian wishest to find e way to a significant is optimal in two ways:

https://each.city is visited only once, and the total route is as short as possible.



Complexity example: Traveling Salesman Problem



This is a *hard* problem: the only known algorithms (so far) to solve it have exponential complexity, that is, the number of operations required to solve it grows as exp(n) for n cities.

Why is exponential complexity "hard"?

It means that the number of operations necessary to compute the exact solution of the problem grows exponentially with the size of the problem (here, the number of cities).

Assignment Project Exam Help

```
• \exp(1) = 2.72 https://tutorcs.com

• \exp(10) = 2.20 10^4 (daily salesman trip)

• \exp(100) • \exp(500) = 1.40 10^{217} (music band worldwide tour)

• \exp(250,000) = 10^{108,573} (fedex, postal services)

• Fastest computer = 10^{12} operations/second
```

So...

In general, exponential complexity problems cannot be solved for any but the smallest instances!

https://tutorcs.com

WeChat: cstutorcs

Complexity

Polynomial-time (P) problems: we can find algorithms that will solve them in a time (=number of operations) that grows polynomially with the size of the input.
 Assignment Project Exam Help

https://tutorcs.com

For example: sort n numbers into increasing order: poor algorithms have n^2 complexity, better ones have $n \log(n)$ complexity.

Complexity

- Since we did not state what the order of the polynomial is, it could be very large! Are there algorithms that require more than polynomial time?
- Yes (until proof of the contrary); for some algorithms, we do not know of any polynomial-time algorithm to solve them. These belong to the class of nondeterministic-polyhotys: titte (OPC algorithms (which includes P problems as well as harder ones).

WeChat: cstutorcs

- for example: traveling salesman problem.
- In particular, exponential-time algorithms are believed to be NP.

Note on NP-hard problems

The formal definition of NP problems is:

A problem is nondetectionistic entyremial effect exists forthe algorithm that can guess a solution and then verify whether or not the guess is correct in polynomial time.

https://tutorcs.com

(one can also state this as these problems being solvable in polynomial time on a nondeterministic Turing machine.)

In practice, until proof of the contrary, this means that known algorithms that run on known computer architectures will take more than polynomial time to solve the problem.

Complexity: O() and o() measures (Landau symbols)

- How can we represent the complexity of an algorithm?
- Problem input (or instance) size: n
 Number of sperations to solve problem: X(a)m Help Given:
- If, for a given function to the state of t

then
$$\exists k \in \Re, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \quad n_0, f(n) \leq kg(n)$$

$$f \in O(g) \qquad \text{f is dominated by g}$$

If, for a given function g(n), we have:

$$\forall k \in \Re, \exists n_0 \in \mathbb{N}, \forall n \in \mathbb{N}, n \quad n_0, f(n) \leq kg(n)$$

$$f \in o(g) \qquad \text{f is negligible compared to g}$$

then

Landau symbols

$$f \in O(g)$$
 sign $f \in O(g)$ sign $f \in O(g)$ $f \in O(g)$

$$f \in o(g) \Leftrightarrow \forall k, f(n) \leq kg(n) \Leftrightarrow \frac{f(n)}{g(n)} \underset{n \to \infty}{\longrightarrow} 0$$

Examples, properties

- f(n)=n, $g(n)=n^2$: n is $o(n^2)$, because $n/n^2 = 1/n \rightarrow 0$ as $n \rightarrow infinity$ similarly, log(n) is o(n). Project Exam Help n^C is o(exp(n)) for any C
- if f is O(g), then for any K, K. is also O(g); idem for o()
- if f is O(h) and g is O(h), then for any K, L: K.f + L.g is O(h) idem for o() WeChat: cstutorcs
- if f is O(g) and g is O(h), then f is O(h)
- if f is O(g) and g is o(h), then f is o(h)
- if f is o(g) and g is O(h), then f is o(h)

Polynomial-time hierarchy

 From Handbook of Brain Theory & Neural Networks

(Arbib, ed.;

MIT Press 1995).

Assignment Project Exam Help

AC⁰ NC¹ NC P complete

https://tutorcs.com

PH

WeChat: estutores

ACo: can be solved using gates of constant depth

NC1: can be solved in logarithmic depth using 2-input gates

NC: can be solved by small, fast parallel computer

P: can be solved in polynomial time

P-complete: hardest problems in P; if one of them can be proven to be

NC, then P = NC

NP: nondeterministic-polynomial algorithms

NP-complete: hardest NP problems; if one of them can be proven to be

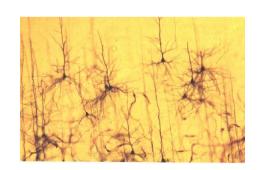
P, then NP = P

PH: polynomial-time hierarchy

Complexity and the human brain

- Are computers close to human brain power?
- Current computer chip (CPU):
 - 10^3 inputs Apigs) ignment Project Exam Ho
 10^7 processing elements (gates)

 - 2 inputs per processing element (fan-in = 2)
 - processing elements quitipuse: podient (by CORC AND) NOT, etc)
- Typical human brain:
 - 10^7 inputs (sensors WeChat: cstutorcs
 - 10^10 processing elements (neurons)
 - $fan-in = 10^3$
 - processing elements compute complicated functions



Still a lot of improvement needed for computers; but computer clusters come close!

Remember: Implementation of search algorithms

```
Function General-Search(problem, Queuing-Fn) returns a solution, or failure

nodes ← make-queue(make-node(initial-state[problem]))

loop doAssignment Project Exam Help

if nodes is empty then return failure

node ← Remove-Front(nodes)

if Goaltes Sprobled Lappes to State(node) succeeds then return node

nodes ← Queuing-Fn(nodes, Expand(node, Operators[problem]))

end

WeChat: cstutorcs
```

Queuing-Fn(*queue, elements***)** is a queuing function that inserts a set of elements into the queue and <u>determines the order of node expansion</u>. Varieties of the queuing function produce varieties of the search algorithm.

Remember: Implementation of search algorithms

```
Function General-Search(problem, Queuing-Fn) returns a solution, or failure

nodes ← make-queue(make-node(initial-state[problem]))

loop doAssignment Project Exam Help

if nodes is empty then return failure

node ← Remove-Front(nodes)

if Goattes Sprobleb 1 (1990 est capacide) succeeds then return node

nodes ← Queuing-Fn(nodes, Expand(node, Operators[problem]))

end

WeChat: cstutorcs
```

Breadth-first search: Enqueue expanded (children) nodes to the back of the queue (FIFO order)

Depth-first search: Enqueue expanded (children) nodes to the front of the queue (LIFO order)

Uniform cost search: Enqueue expanded (children) nodes so that queue is ordered by path cost (priority queue order).

Encapsulating state information in nodes

A state is a (representation of) a physical configuration A node is a data structure constituting part of a search tree includes parent, children, depth, path cost q(x)States of got ingrement. Pitogie of the xount Field ###orcs.com depth = 6State Node g = 6children

The EXPAND function creates new nodes, filling in the various fields and using the OPERATORS (or SUCCESSORFN) of the problem to create the corresponding states.

Evaluation of search strategies

- A search strategy is defined by picking the order of node expansion.
- Search algorithms are commonly evaluated according to the following four criteria:
 - Complete negs: does it always find a solution if one exists?
 - Time complexity: how long does it take as function of num. of nodes?
 - Space complexity: how much memory does it require?
 - · Optimality de it qualantee helestics solution?
- Time and space complexity are measured in terms of:
 - $b \max$ branching factor of the search tree
 - *d* depth of the least-cost solution
 - $m \max$ depth of the search tree (may be infinity)

Note: Approximations

In our complexity analysis, we do not take the built-in <u>loop-detection</u> into account.

Assignment Project Exam Help
The results only 'formally' apply to the variants of our algorithms

 The results only 'formally' apply to the variants of our algorithms WITHOUT loop-checks.

- Studying the left of the loop the king of the complexity is hard:
 - overhead of the checking MAY or MAY NOT be compensated by the reduction of the size of the tree.
- Also: our analysis DOES NOT take the length (space) of representing paths into account !!

Uninformed search strategies

Use only information available in the problem formulation

- Breadth-first Assignment Project Exam Help
- Uniform-cost
- Depth-first https://tutorcs.com
- Depth-limited
- Iterative deepening WeChat: cstutorcs

Expand shallowest unexpanded node

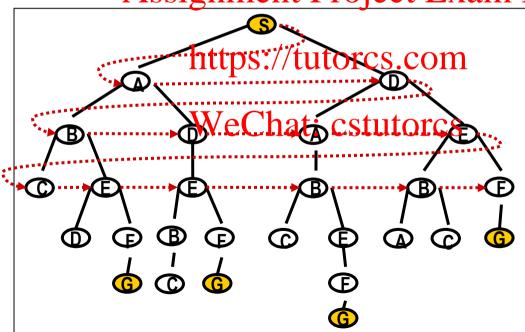
Implementation:

Assignment Projects Example lp

https://tutorcs.com

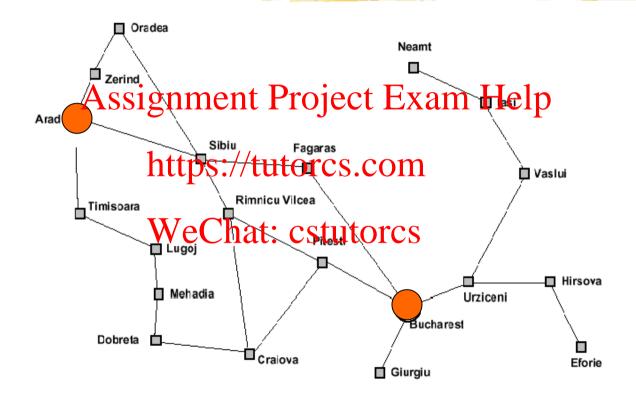
WeChat: cstutorcs





Move downwards, level by level, until goal is reached.

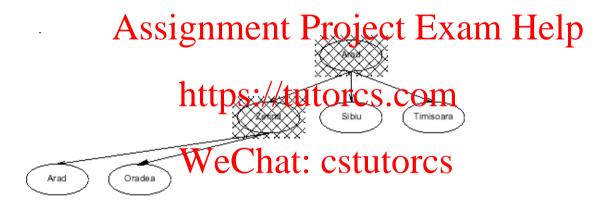
Example: Traveling from Arad To Bucharest

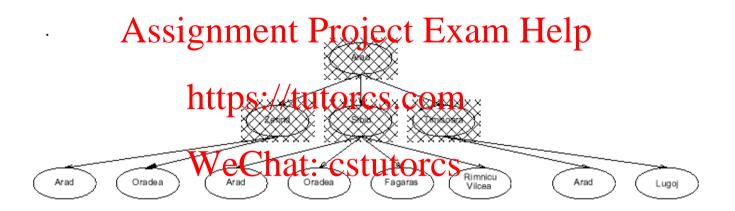


Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs





Properties of breadth-first search

- Completeness:
- Time complexity:
- Space complexity:
 Optimality: Signment Project Exam Help

https://tutorcs.com

- Search algorithms are commonly evaluated according to the following four criteria:
 - **Completeness:** does it always find a solution if one exists?
 - Time complexity: And languages it take as junction of gum. of nodes?
 - **Space complexity:** how much memory does it require?
 - **Optimality:** does it guarantee the least-cost solution?
- Time and space complexity are measured in terms of:
 - b max branching factor of the search tree
 - d depth of the least-cost solution
 - m max depth of the search tree (may be infinity)

Properties of breadth-first search

Completeness: Yes, if b is finite

Time complexity: $1+b+b^2+...+b^d = O(b^d)$, i.e., exponential in d

Space complexity: O(b d) (see following slides)
Optimality: Assignment assuming cost Expense plep Optimality:

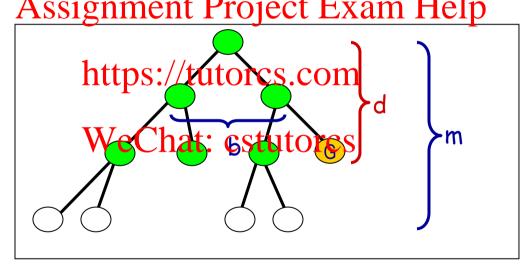
https://tutorcs.com

WeChat: cstutorcs

Time complexity of breadth-first search

• If a goal node is found on depth d of the tree, all nodes up till that depth are created and examined (note: and the children of nodes at depth d are created and enqueued, but not yet examined).

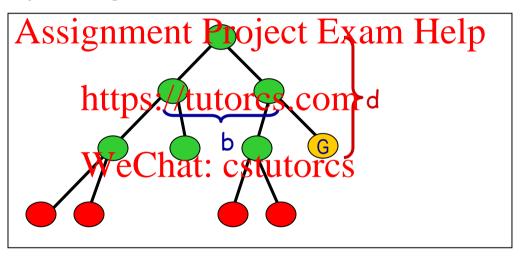
Assignment Project Exam Help



• <u>Thus</u>: O(b^d)

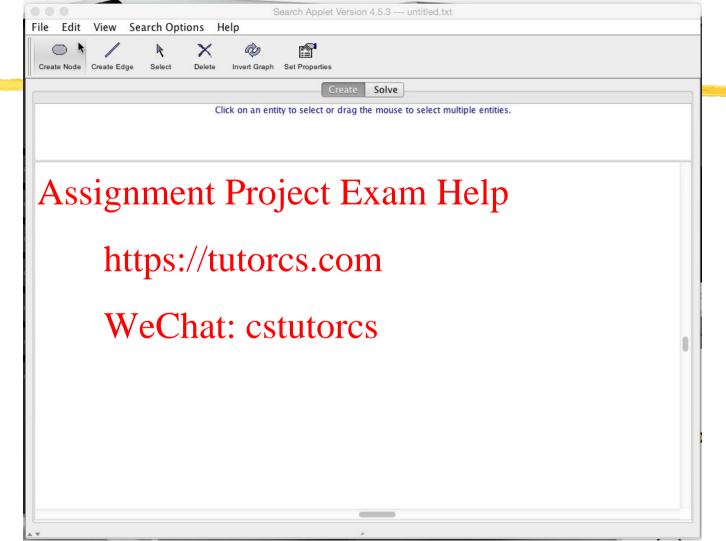
Space complexity of breadth-first

Largest number of nodes in QUEUE is reached on the level d+1
just beyond the goal node.



- QUEUE contains all nodes. (Thus: 4).
- In General: $b^{d+1} b \sim b^{d}$

Demo



Expand least-cost unexpanded node

Implementation:

QASSignatint Project Exem Helps

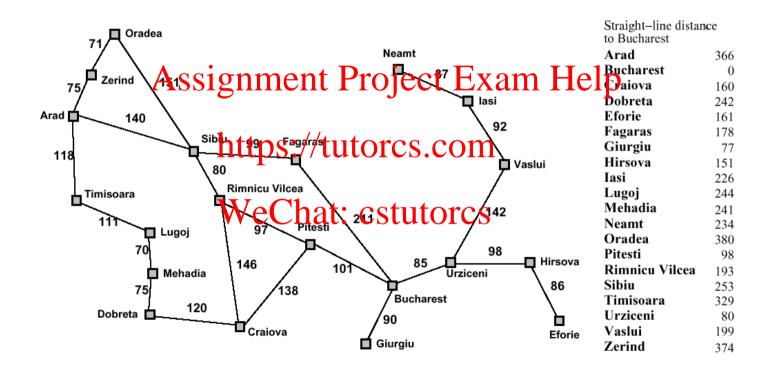
https://tutorcs.com

So, the queueing function keeps the node list sorted by increasing path cost, and we expand the first unexpanded node (hence with smallest particus) Stutores

A refinement of the breadth-first strategy:

Breadth-first = uniform-cost with path cost = node depth

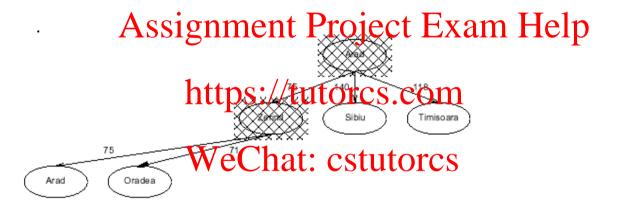
Romania with step costs in km

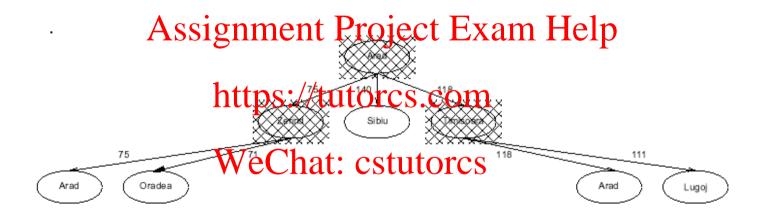


Assignment Project Exam Help

https://tutorcs.com

WeChat: cstutorcs





Properties of uniform-cost search

- Completeness: Yes, if step cost $\varepsilon > 0$
- Time complexity: # nodes with $g \le cost$ of optimal solution, $\le O(b^d)$
- Space completely mentals will get cost x fedge male of the confidence of the confi
- Optimality: Yes, as long as path cost never decreases

WeChat: cstutorcs

g(n) is the path cost to node n

Remember:

b = branching factor

d = depth of least-cost solution

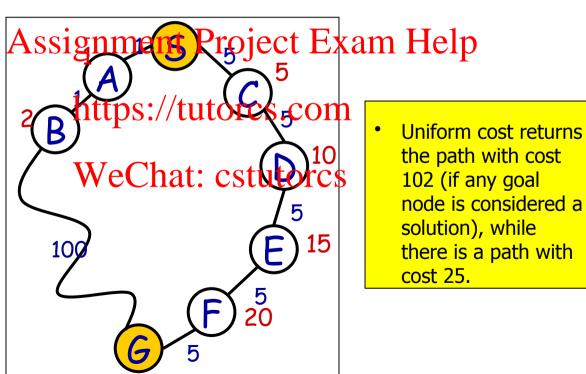
Implementation of uniform-cost search

- Initialize Oueue with root node (built from start state)
- Assignment Project Exam Help Repeat until (Queue empty) or (first node has Goal state):
 - Remove first node from front of Queue

 - Expand node (find its children)
 - Reject those children that have already been considered, to avoid loops
 - Add remaining children to Queue, in a way that keeps entire queue sorted by increasing path cost
- If Goal was reached, return success, otherwise failure

Caution!

 Uniform-cost search not optimal if it is terminated when any node in the queue has goal state.



Note: Loop Detection

- In class, we saw that the search may fail or be sub-optimal if:
 - no loop detection: then algorithm runs into infinite cycles

 Assignment Project Exam Help
 - not queuing-up a node that has a state which we have already isited that a state which we have
 - simply avoiding to go back to our parent: looks promising, but we have not proven that now rks

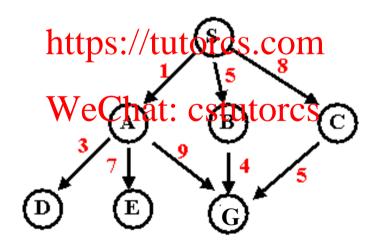
Solution? do not enqueue a node if its state matches the state of any of its parents (assuming path costs>0).

Indeed, if path costs > 0, it will always cost us more to consider a node with that state again than it had already cost us the first time.

Is that enough??

From: http://www.csee.umbc.edu/471/current/notes/uninformed-search/

Assignment Project Exam Help



Breadth-First Search Solution

From: http://www.csee.umbc.edu/471/current/notes/uninformed-search/

Breadth-First Search

rAssignment Project Feam Help

exp. node nodes list

```
hetps://tutorcs.com

A BCDEG;

C CEGGGG C CStutorcs

C (DEGGGG')

D (EGGGG')

E (GGG')

G (G'G')
```

Solution path found is SAG <-- this Galso has cost 10

Number of nodes expanded (including goal node) = 7

Uniform-Cost Search Solution

From: http://www.csee.umbc.edu/471/current/notes/uninformed-search/

Uniform-Cost Search

```
GENARSSIZAMMENTUPTOJECTUENZAMTHETP
exp. node nodes list
```

Solution path found is SBG <-- this Ghas cost 9, not 10

Number of nodes expanded (including goal node) = 7

Note: Queueing in Uniform-Cost Search

In the previous example, it is wasteful (but not incorrect) to queue-up three nodes with G state, if our goal if to find the least-cost solution:

Although they represent different paths, we know for sure that the one with smallest path cost (9 in the example) will into our cost (9 in the example) will be a support of the example of the exampl

So we can refine the queueingtiunction by itores.com

- queue-up node if

1) its state does not match the state of any parent

and

2) path cost smaller than path cost of any unexpanded node with same state in the queue (and in this case, replace old node with same state by our new node)

Is that it??

A Clean Robust Algorithm

```
Function UniformCost-Search(problem, Queuing-Fn) returns a solution, or failure
  open ← make-queue(make-node(initial-state[problem]))
  closed ← [empty]
  loop do Assignment Project Exam Help
             if open is empty then return failure
             currnode ← Remove-Front(open)
             if Goal dest problem Papplies to State (currnode) then return currnode
             children ← Expand(currnode, Operators[problem])
             while this empty tutores
                                   [... see next slide ...]
             end
             closed ← Insert(closed, currnode)
             open ← Sort-By-PathCost(open)
  end
```

A Clean Robust Algorithm

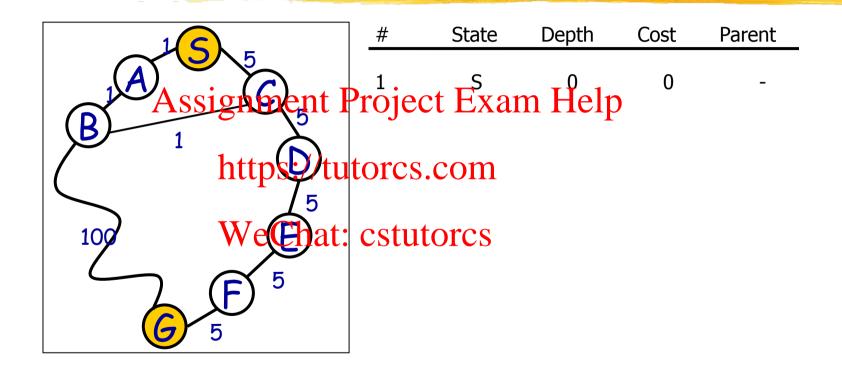
[... see previous slide ...]

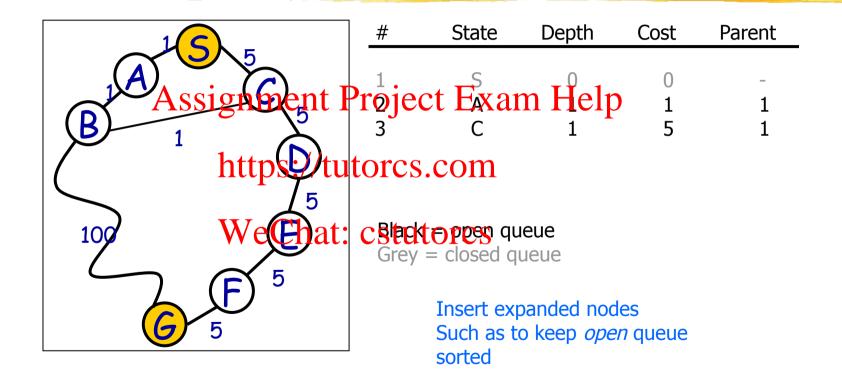
```
[... see previous slide ...]
               children ← Expand(currnode, Operators[problem])
               while children not empty
          Assignment-Reojecto-Exam Help
                           if no node in open or closed has child's state
                  https://tuthere.cists Queuing-Fn(open, child)

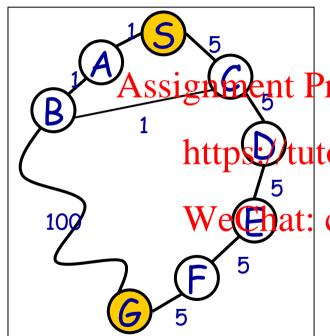
open ← Queuing-Fn(open, child)

in open that has child's state
                                      if PathCost(child) < PathCost(node)
                   WeChat: cstutorcspen ← Delete-Node(open, node)
                                                  open ← Queuing-Fn(open, child)
                           else if there exists node in closed that has child's state
                                      if PathCost(child) < PathCost(node)
                                                  closed ← Delete-Node(closed, node)
                                                  open ← Queuing-Fn(open, child)
               end
```

CS 561. Sessions 2-3



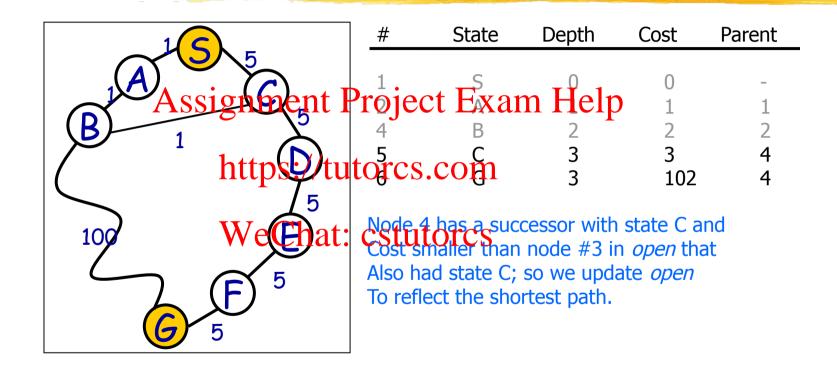


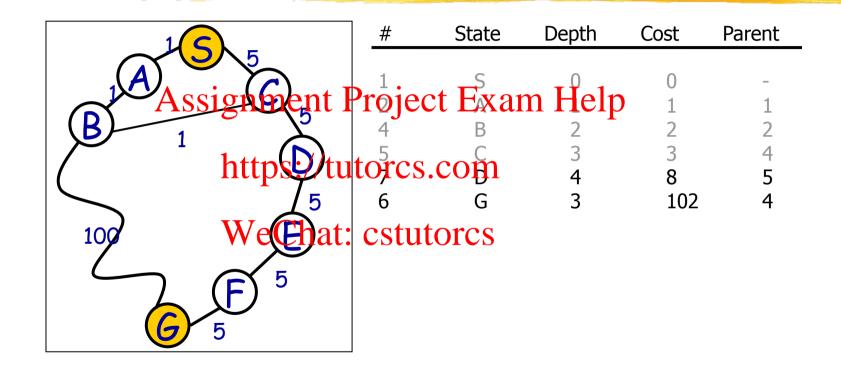


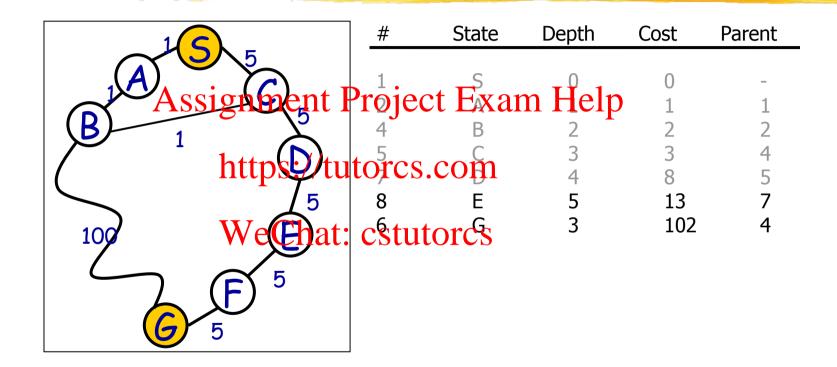
| # | State | Depth | Cost | Parent |
|----------------------|-------|-------|------------------|------------------|
| oject 4 orcs.c | Exam | Help | 0 1 2 5 | - 1 2 1 |

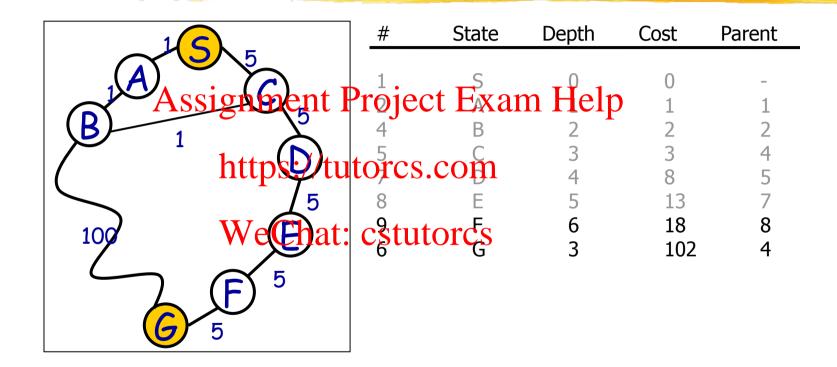
Node 2 has 2 successors: one with state B and one with state S.

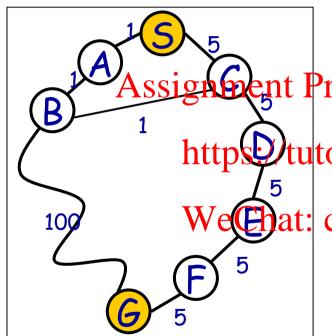
We have node #1 in *closed* with state S; but its path cost 0 is smaller than the path cost obtained by expanding from A to S. So we do not queue-up the successor of node 2 that has state S.





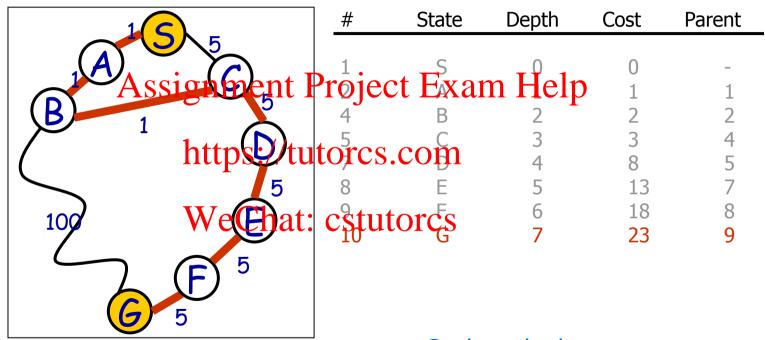






| | # | State | Depth | Cost | Parent |
|-----|---------------|-------|--------|------|--------|
| | 1. | S | TD-1 | 0 | - |
| M | O ject | Exai | m Help | 1 | 1 |
| | 4 | В | 2 | 2 | 2 |
| 14, | orcs.c | C | 3 | 3 | 4 |
| μu | ACS.C | OHI | 4 | 8 | 5 |
| | 8 | Е | 5 | 13 | 7 |
| : (| estuto | orcs | 6 | 18 | 8 |
| | 10 | G | / | 23 | 9 |

The node with state G and cost 102 has been removed from the open queue and replaced by cheaper node with state G and cost 23 which was pushed into the open queue.



Goal reached

Expand deepest unexpanded node

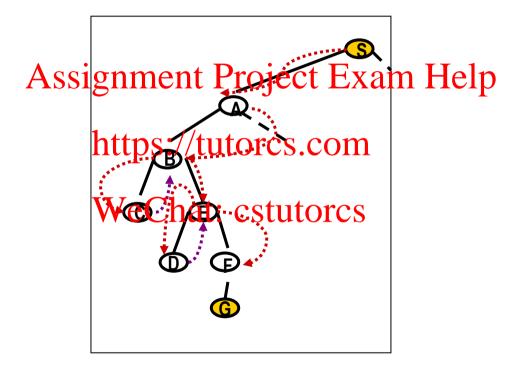
Implementation:

Assignments Projecto Extent Melye

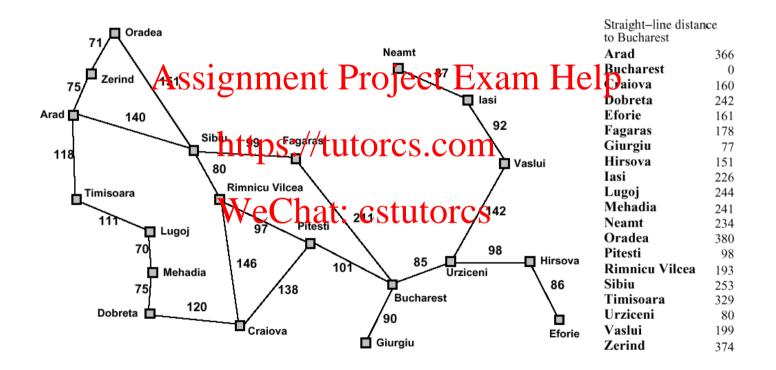
https://tutorcs.com

WeChat: cstutorcs

Depth First Search



Romania with step costs in km



Assignment Project Exam Help

https://tutorcs.com

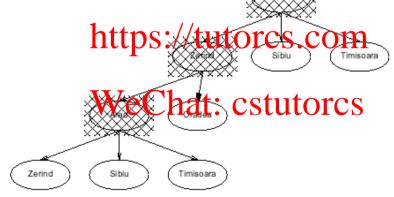
WeChat: cstutorcs

Assignment Project Exam Help

https://tutorcs.com

WeChate cstutorcs





I.e., depth-first search can perform infinite cyclic excursions Need a finite, non-cyclic search space (or repeated-state checking)

CS 561, Sessions 2-3

Properties of depth-first search

Completeness: No, fails in infinite state-space (yes if finite

state space)

Time complexity: $O(b^m)$ Space complexity mental project Exam Help

Optimality: No

https://tutorcs.com

WeChat: cstutores

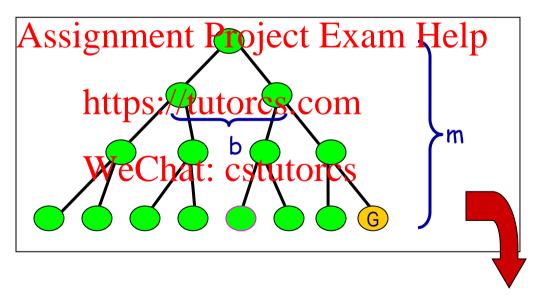
Remember:

b = branching factor

m = max depth of search tree

Time complexity of depth-first: details

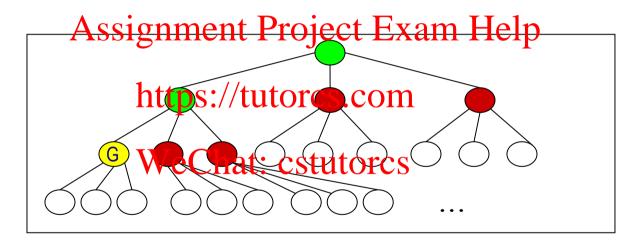
- In the worst case:
 - the (only) goal node may be on the right-most branch,



- Time complexity == $b^{m} + b^{m-1} + ... + 1 = b^{m+1} 1$
- Thus: O(b^m)

Space complexity of depth-first

- Largest number of nodes in QUEUE is reached in bottom left-most node.
- Example: m = 2, b = 3:



- QUEUE contains all nodes. Thus: 4.
- In General: ((b-1) * m)
- Order: O(m*b)

Avoiding repeated states

In increasing order of effectiveness and computational overhead:

- do not return to state we come from, i.e., expand function will skip possible successors that Aresing a mestate as jede's parent. Help
- do not create paths with cycles, i.e., expand function will skip possible successors that are in starps state to large of orders ancestors.
- do not generate any state that was ever generated before, by keeping track (in memory) of every water generated before, by keeping track is lower than last time we reached it.

Depth-limited search

Assignment Project Exam Help

Is a depth-first search with depth limit l

Implementation: https://tutorcs.com

Nodes at depth *l* have no successors.

Complete: if cutoff chosen appropriately then it is guaranteed to find a solution.

Optimal: it does not guarantee to find the least-cost solution

Iterative deepening search

```
Function Iterative-deepening-Search(problem) returns a solution,

or failure

for depth = 0 to ∞ do nt Project Exam Help
result ← Depth-Limited Search(problem, depth)

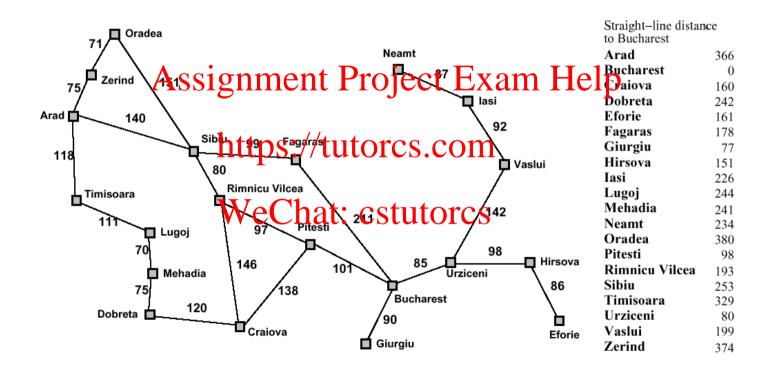
if result succeeds then return result
end https://tutorcs.com
return failure
```

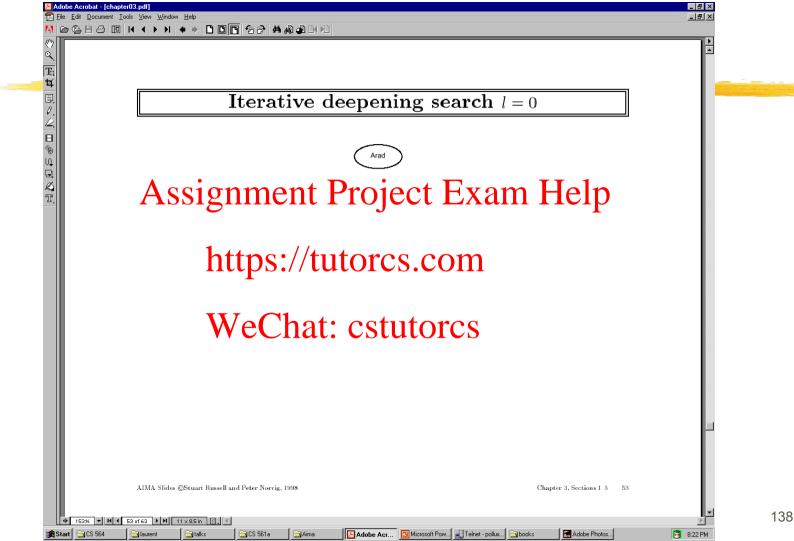
Combines the best of breadth-first and depth-first search strategies.

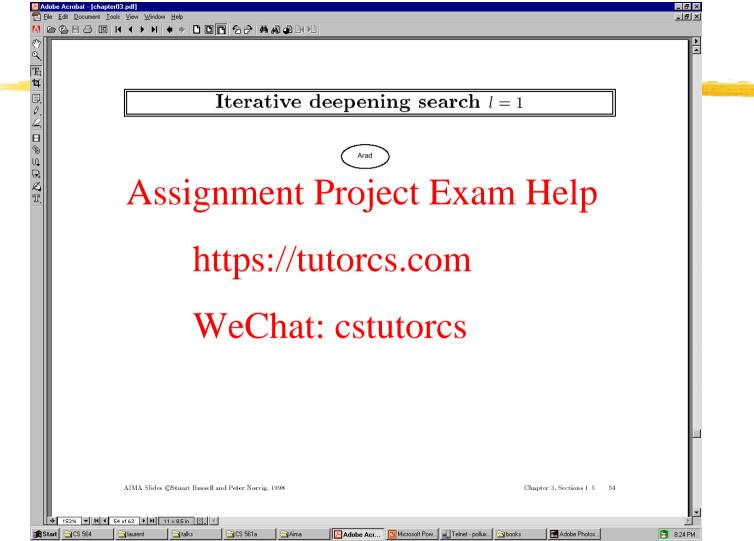
Completeness: Yes,
 Time complexity: O(b^d)
 Space complexity: O(bd)

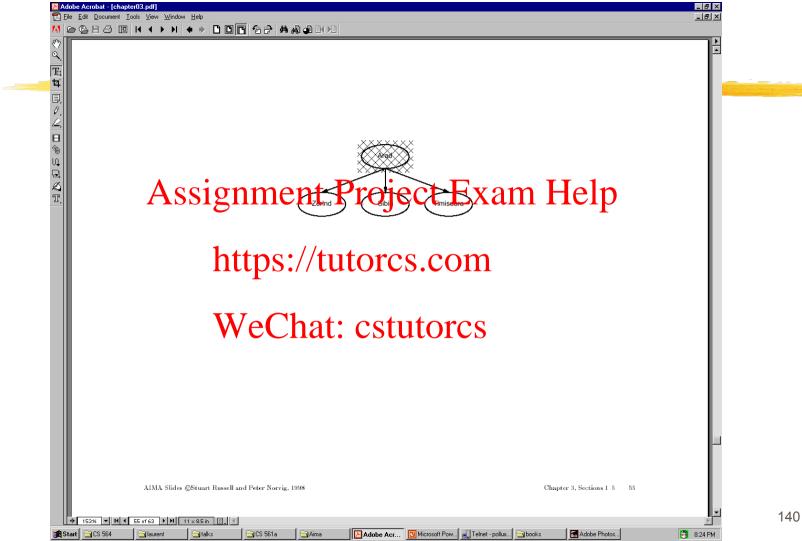
Optimality: Yes, if step cost = 1

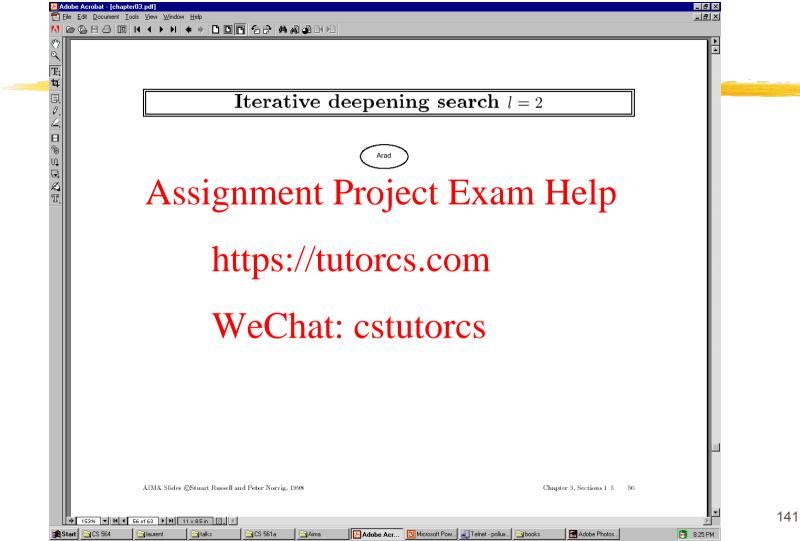
Romania with step costs in km

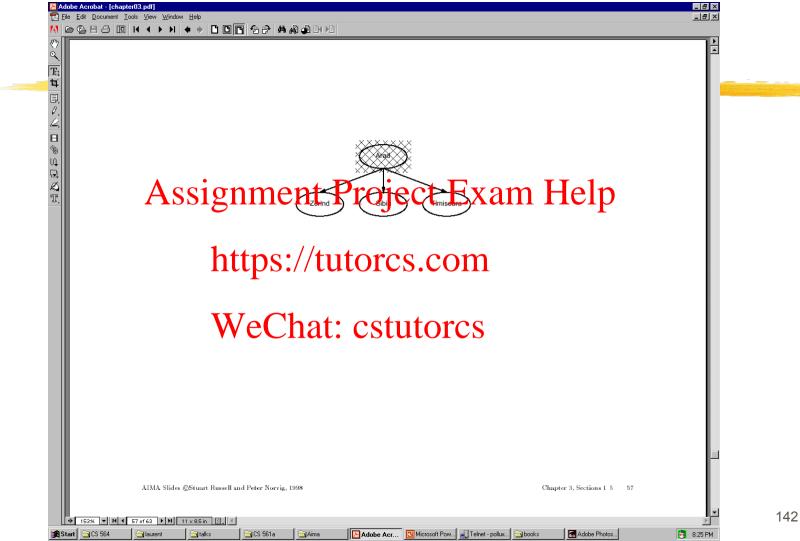


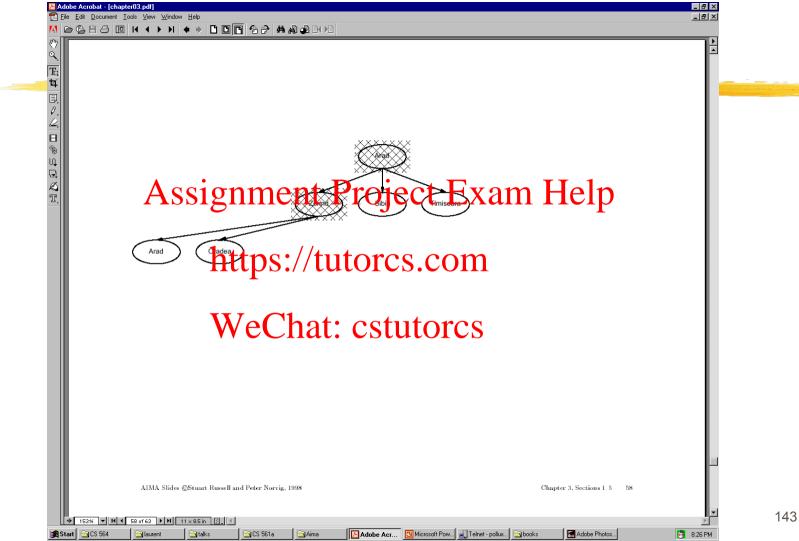


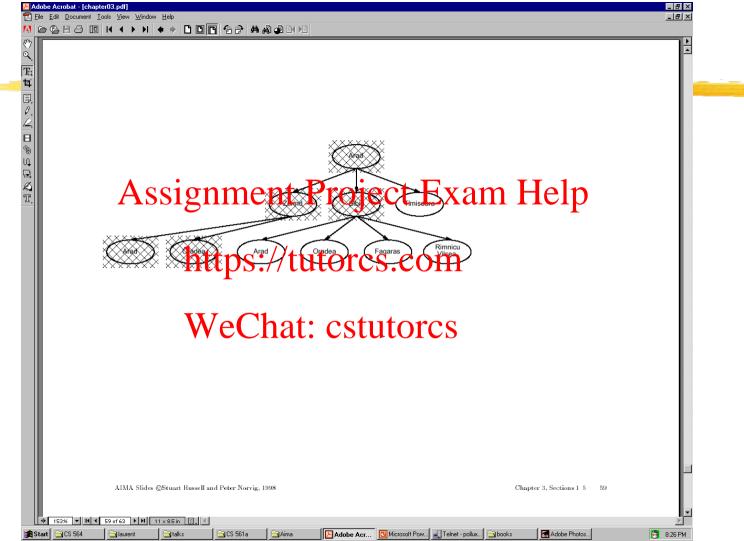


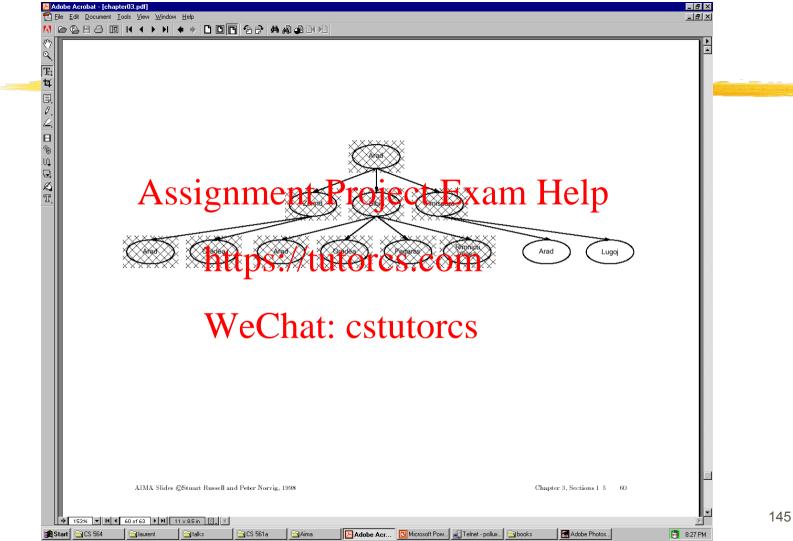












Iterative deepening complexity

Iterative deepening search may seem wasteful because so many states are expanded multiple times.

Assignment Project Exam Help
In practice, however, the overhead of these multiple expansions is small, because most of the nodes are towards leaves (bottom) of the search tree: thus, the nodes that are evaluated several times

(towards topyof tree) are in relatively small number.

Iterative deepening complexity

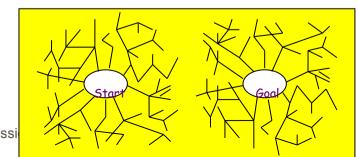
In iterative deepening jodes a horogen level above twice, etc. up to root (expanded d+1 times) so total number of expansions is:

(d+1)1 + (d)b + (d-1)b^2 http3b//(d12tor2b3(d-1)m1b^d = O(b^d)

WeChat: cstutorcs

 In general, iterative deepening is preferred to depth-first or breadth-first when search space large and depth of solution not known.

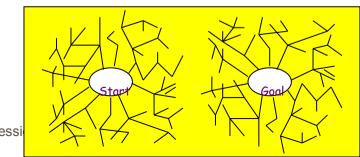
- Both search forward from initial state, and backwards from goal.
- Stop when the two searches meet in the middle.
- Problem: how do we saignment Project Exam Help
 - predecessor of node n = all nodes that have n as successor
 - this may not always be tasy so computer cs.com
 - if several goal states, apply predecessor function to them just as we applied successor (only works well if goals are explicitly known; may be difficult if goals only characterized implicitly).



- Problem: how do we search backwards from goal?? (cont.)
 - ...
 - for bidirectional search to work well, there must be an efficient way to check whether a given node belongstby the new search of the control of the contro
 - select a given search algorithm for each half.

https://tutorcs.com

WeChat: cstutorcs

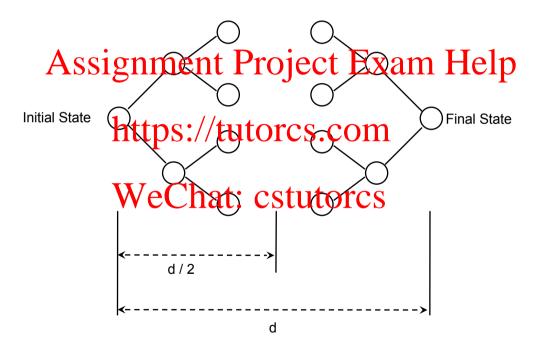


```
1. QUEUE1 <-- path only containing the root;
  QUEUE2 <-- path only containing the goal;
Assignment Project Exam Help
2. WHILE both QUEUEs are not empty
  AND QUEUE1 and QUEUE2 do NOT share a state
           https://tutorcs.com
   DO remove their first paths;
         create their new paths (to all children);
         reject Cheil halv. pat 8 4 Wit 10 16 6 ps;
         add their new paths to back;
3. IF QUEUE1 and QUEUE2 share a state
         THEN success;
         ELSE failure;
```

- Completeness: Yes,
 Time complexity: Signment Broject Exam Help
- Space complexity: O(b m/2) / https://tutorcs.com
 Optimality: Yes

WeChat: cstutorcs

- To avoid one by one comparison, we need a hash table of size $O(b^{m/2})$
- If hash table is used, the cost of comparison is O(1)



- Bidirections in Project Exam Help
 Big difference for problems with branching factor b in
 - both directions://tutorcs.com
 A solution of length d will be found in $O(2b^{d/2}) = O(b^{d/2})$

 - For b = 10 and d = 6, only 2,222 nodes are needed instead of 1,111,111 or breadth-fiss seal of 1.

- Bidirectional search issues
 - · Predecessors dessignement designetates am Help
 - Difficult when operators are not reversible

https://tutorcs.com

- What to do if there is no *explicit list of goal* states? WeChat: cstutorcs
- For each node: *check if it appeared in the other search*
 - Needs a hash table of O(b^{d/2})
- What is the best search strategy for the two searches?

Comparing uninformed search strategies

| Criterion | Breadth- first | Uniform cost | Depth- first | Depth- limited | Iterative deepening | Bidirectional (if applicable) |
|-----------|-------------------|-----------------|-----------------|-------------------|---------------------|-------------------------------|
| Time | A§ŝignr | nhêht P | rbîject i | Ēâam | Help | b^(d/2) |
| Space | b^d | b^d | bm | bl | bd | b^(d/2) |
| Optimal? | Yes http | os://tut | orcs.co | Om | Yes | Yes |
| Complete? | Yes We | Chat: | c\stuto1 | if Id | Yes | Yes |

- *b* − max branching factor of the search tree
- *d* − depth of the least-cost solution
- $m \max$ depth of the state-space (may be infinity)
- /- depth cutoff

Summary

- Problem formulation usually requires abstracting away real-world details to define a state space that can be explored using computer algorithms X am Help
- Once problem is formulated in post/act form; completing analysis helps us picking out best algorithm to solve problem.
- Variety of uninformed search strategies; difference lies in method used to pick node that will be further expanded.
- Iterative deepening search only uses linear space and not much more time than other uniformed search strategies.