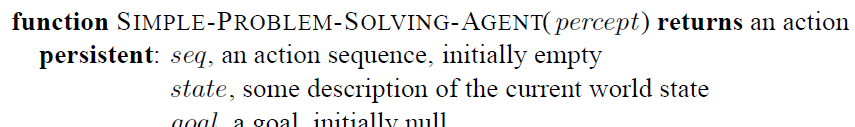
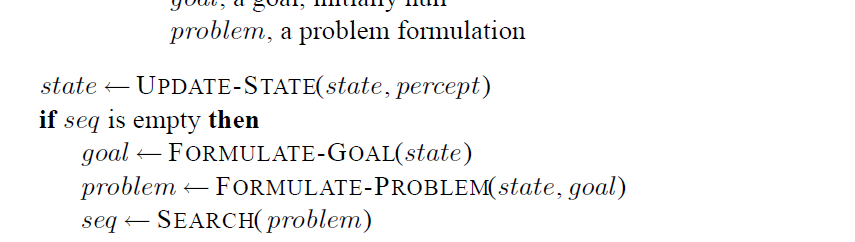
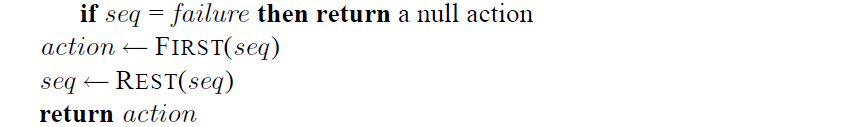
1 Artificial Intelligence Chapter 3: Solving Problems by Searching Andreas Zell After the Textbook: Artificial Intelligence After the Textbook: Artificial Intelligence, A Modern Approach by Stuart Russel and Peter Norvig (3rd Edition) • Reflex agents are too simple and have great difficulties in learning desired action sequences • Goal-based agents can succeed by considering Solving Problems by Searching Goal based agents can succeed by considering future actions and the desirability of their outcomes. • We now describe a special type of goal-based agents called problem-solving agents, which try to find action sequences that lead to desirable states states. • These are uninformed algorithms, they are given no hints or heuristics for the problem solution other than its definition. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 2



2 • We first need a goal formulation, based on the current situation and the performance measure. • Problem formulation is the process of deciding 3.1 Problem-Solving Agents Problem formulation is the process of deciding what actions and states to consider, given a goal. • In general, an agent with several options for action of unknown value can decide what to do by first examining different possible sequences of actions that lead to states of known value and of actions that lead to states of known value, and then chooseing the best sequence. • A search algorithm takes a problem as input and returns a solution in form of an action sequence. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 3 Simple Problem-Solving Agent Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 4

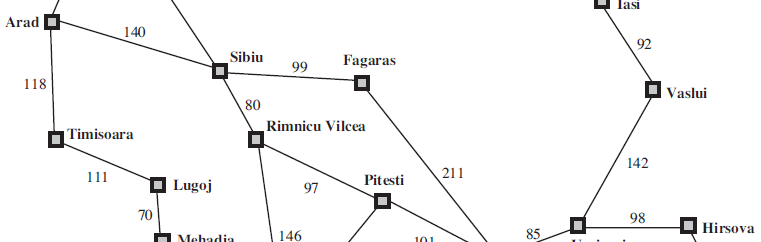


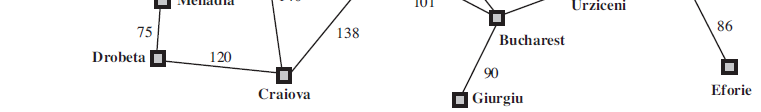




3 • On holiday in Romania; currently in Arad. • Flight leaves tomorrow from Bucharest 3.2 Example Problem: Romania Tour • Formulate goal: • be in Bucharest • Formulate problem: • states: various cities • actions: drive between cities • Find solution: • sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 5 Example Problem: Romania Tour Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 6

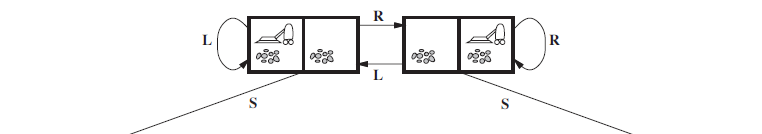


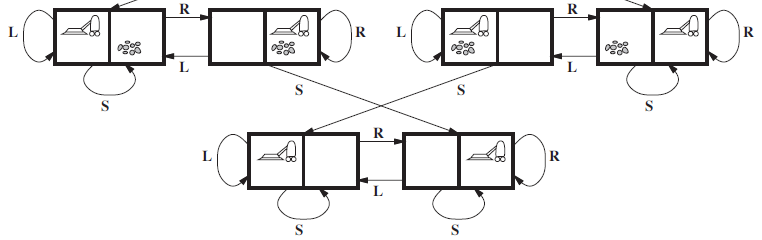




4 • Deterministic, fully observable single-state problem • Agent knows exactly which state it will be in; solution is a sequence Problem Type of Romania Tour • Non-observable sensorless problem (conformant problem) • Agent may have no idea where it is; solution is a sequence • Nondeterministic and/or partially observable  contingency problem • percepts provide new information about current state • often interleave search and execution • Unknown state space exploration problem Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 7 A problem is defined by four items: 1. Initial state e.g., "at Arad" 2. Actions or successor function Single-State Problem Formulation • S(x) = set of action–state pairs • e.g., S(Arad) = {<Arad Zerind, Zerind>, … } 3. Goal test, can be • explicit, e.g., x = "at Bucharest" • implicit, e.g., Checkmate(x) 4. Path cost function (additive) • e g sum of distances # actions executed etc • e.g., sum of distances, # actions executed, etc. • c(x,a,y) is the step cost, assumed to be ≥ 0 • A solution is a sequence of actions leading from the initial state to a goal state Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 8

5 • Real world is absurdly complex, therefore state space must be abstracted for problem solving • (Abstract) state = set of real states Selecting a State Space • (Abstract) action = complex combination of real actions • e.g., "Arad Zerind" represents a complex set of possible routes, detours, rest stops, etc. • For guaranteed realizability, any real state "in Arad“ must get to some real state "in Zerind" • (Abstract) solution = set of real paths that are solutions in the real world the real world • Each abstract action should be "easier" than the original problem Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 9 Vacuum World State Space Graph • States: dirt and robot location • Actions: Left, Right, Suck • Goal test: no dirt at all locations • Path cost: 1 per action Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 10

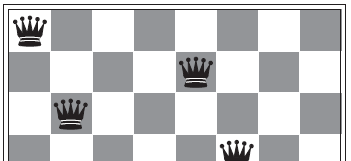


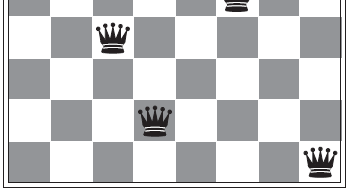


6 Example: The 8-Puzzle • States: locations of tiles • Actions: move blank left, right, up, down Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 11 Actions: move blank left, right, up, down • Goal test: state matches goal state (given) • Path cost: 1 per move (Note that the sliding-block puzzles are NP-hard) 8-Queens Problem Almost a solution • States: any arrangement of 0-8 queens (because of white diagonal) on the board • Actions: add a queen to an empty square • Goal test: 8 queens on board, none attacked • Path cost: 1 per move Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 12

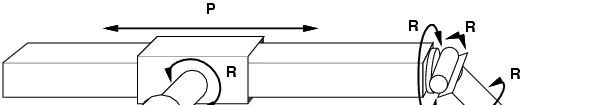


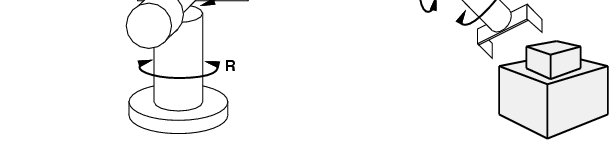




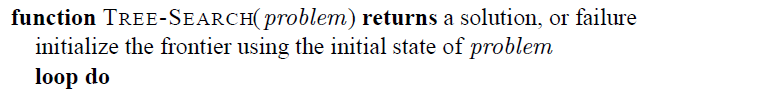


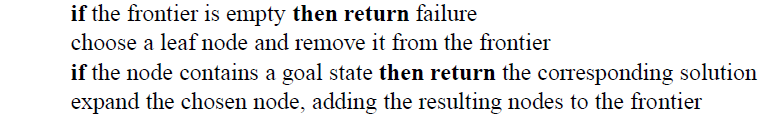
7 Example: Robotic Assembly • States: coordinates of robot joint angles, t f bj t t b bl d parts of object to be assembled • Actions: continuous motions of robot joints • Goal test: complete assembly • Path cost: time to execute 13 • Route-finding problems • GPS-based navigation systems, Google maps Touring problems Real-World Search Problems • Touring problems • TSP problem • VLSI layout problems • Robot navigation problems • Automatic assembly sequencing • Internet searching • Searching paths in metabolic networks in bioinformatics Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 14

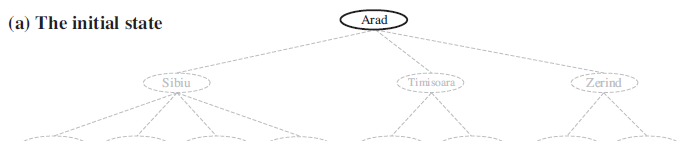


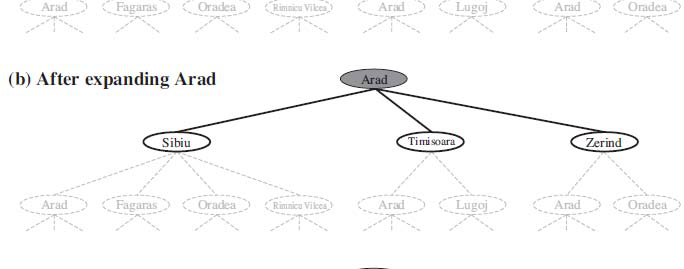


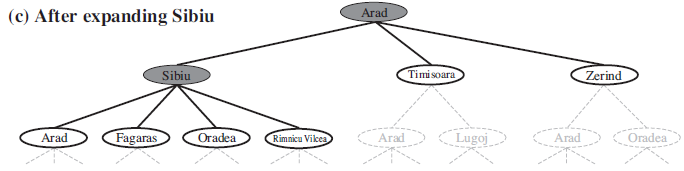
8 • Basic idea of tree search algorithms: • offline, simulated exploration of state space by generating successors of already-explored states 3.3 Searching for Solutions generating successors of already explored states (a.k.a.~expanding states) Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 15 Search Tree, Example Romania Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 16



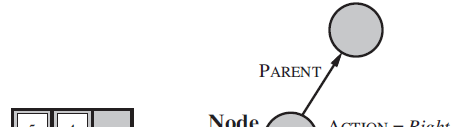


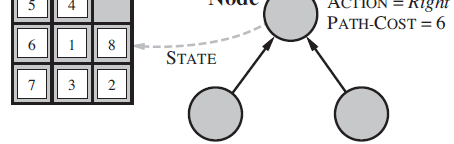


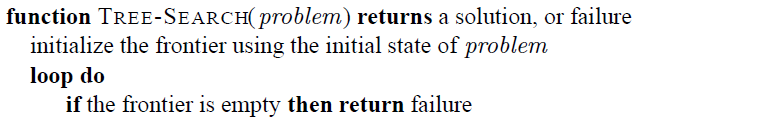


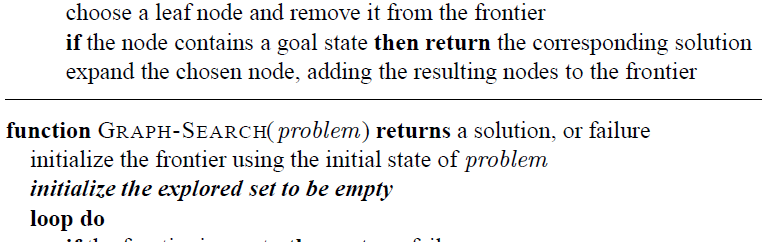


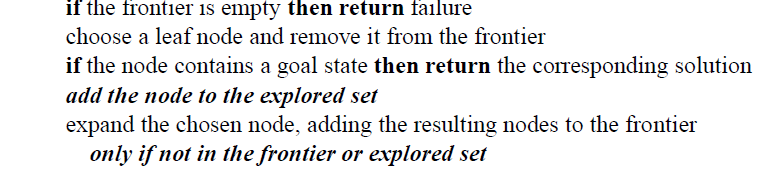
9 Search Tree Data Structures Nodes are the data structures from which Queue data structure to store frontier of unexpanded nodes: • Make Queue(element ) creates queue w given elements the search tree is constructed • Make-Queue(element, …) creates queue w. given elements • Empty?(queue) returns true iff queue is empty • Pop(queue) returns first elem. and removes it • Insert(element, queue) inserts elem. in queue, returns q. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 17 Tree Search and Graph Search Algorithms Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 18





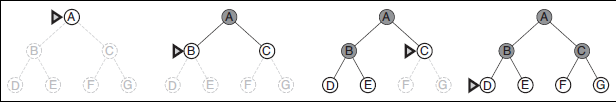


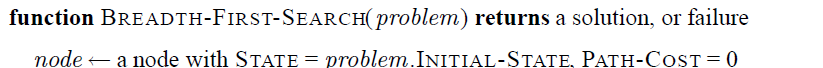


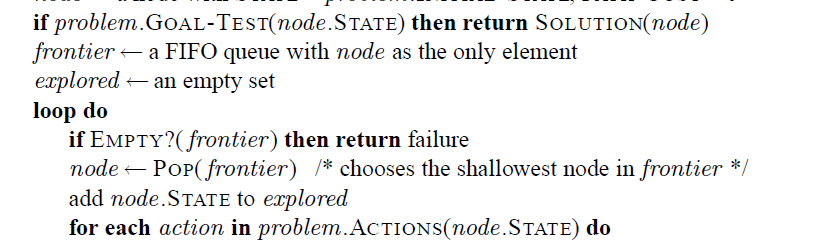


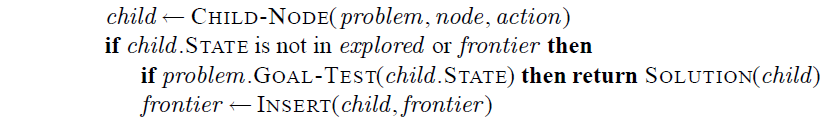
10 • A search strategy is defined by picking the order of node expansion • Strategies are evaluated along the following dimensions: Search Strategies • Strategies are evaluated along the following dimensions: • completeness: does it always find a solution if one exists? • time complexity: number of nodes generated • space complexity: maximum number of nodes in memory • optimality: does it always find a least-cost solution? • Time and space complexity are measured in terms of b i b hi f t f th h t • b: maximum branching factor of the search tree • d: depth of the least-cost solution • m: maximum depth of the state space (may be ∞) 19 • Uninformed (blind) search strategies use only the information available in the problem definition 3.4 Uninformed Search Strategies definition • Breadth-first search (BFS) • Uniform-cost search • Depth-first search (DFS) • Depth-limited search • Iterative deepening search 20

11 • The root node is expanded first • Then all successors of the root node are expanded • Then all their successors 3.4.1 Breadth-First Search (BFS) Then all their successors • … and so on • In general, all the nodes of a given depth are expanded before any node of the next depth is expanded. • Uses a standard queue as data structure Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 21 Breadth-First Search (BFS) Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 22



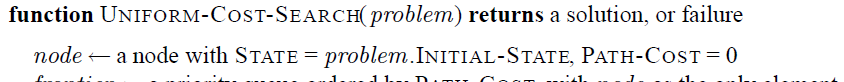


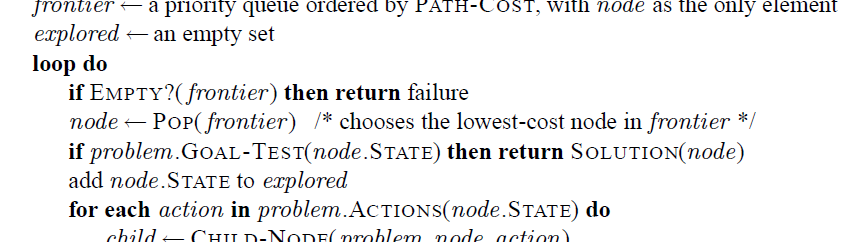


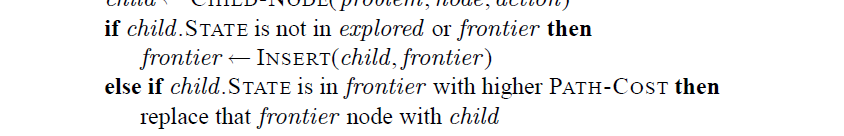


12 Example Breadth-First Search A B C C D E A head tail Status of Queue D E F G E F G H F G H I J H I J K L G H I J B C D E F G H I J K L time Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 23 I J K L J K L M K L M L M M M Empty nodes: unvisited, Grey nodes: frontier (= in queue), Red nodes: closed (= removed from queue) • BFS is complete (always finds goal if one exists) • BFS finds the shallowest path to any goal node. If multiple goal nodes exist, BFS finds the shortest path Breadth-First Search Properties path. • If tree/graph edges have weights, BFS does not find the shortest length path. • If the shallowest solution is at depth d and the goal test is done when each node is generated then BFS generates b + b2 + b3 + … +bd = O(bd) nodes, i.e. has a time complexity of O(bd). y ( ) • If the goal test is done when each node is expanded the time complexity of BFS is O(bd+1). • The space complexity (frontier size) is also O(bd). This is the biggest drawback of BFS. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 24

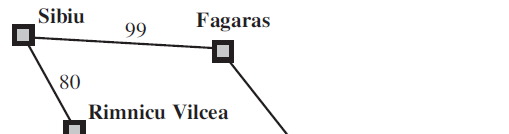
13 • Modification to BFS generates Uniform-cost search, which works with any step-cost function (edge weights/costs): 3.4.2 Uniform-Cost Search (UCS) ( g g ) • UCS expands the node n with lowest summed path cost g(n). • To do this, the frontier is stored as a priority queue. (Sorted list data structure, better heap data structure). Th l t t i li d t d h l t d • The goal test is applied to a node when selected for expansion (not when it is generated). • Also a test is added if a better node is found to a node on the frontier. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 25 Uniform-Cost Search Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 26

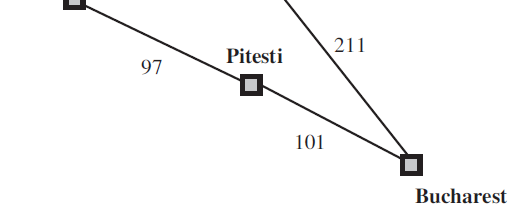




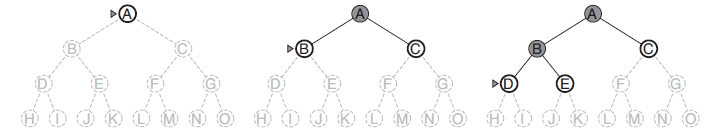


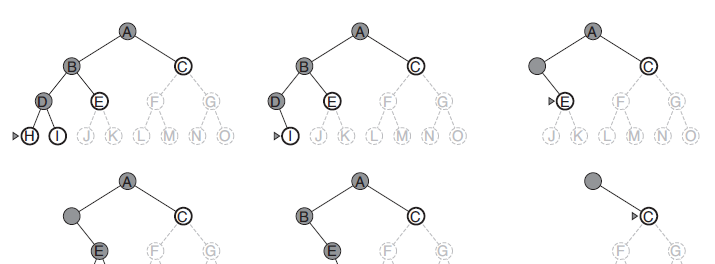
14 Uniform-Cost Search 0 80 99 • Uniform cost search is similar to Dijkstra’s algorithm! 80 177 310 278 • Uniform-cost search is similar to Dijkstra s algorithm! • It requires that all step costs are non-negative • It may get stuck if there is a path with an infinite sequence of zero cost steps. • Otherwise it is complete Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 27 • DFS alwas expands the deepest node in the current frontier of the search tree. It uses a stack (LIFO queue last in first out) 3.4.3 Depth-First Search (DFS) • It uses a stack (LIFO queue, last in first out) • DFS is frequently programmed recursively, then the program call stack is the LIFO queue. • DFS is complete, if the graph is finite. • The tree search version of DFS is complete on a finite tree, if a test is included whether the node has already been visited • DFS is incomplete on infinite trees or graphs. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 28

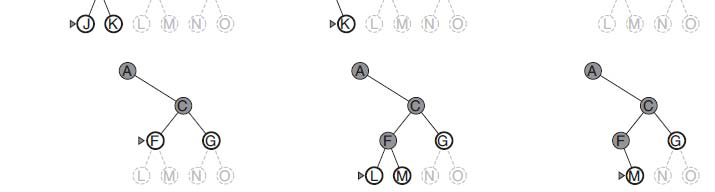




15 Example Depth-First Search A C B C E D A bottom top status of stack C E H C E C J I C J C C J M B C D E F G H I J K L time Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 29 C G F G L K L M Empty nodes: unvisited, Grey nodes: frontier (= on stack), Red nodes: closed (= removed from stack) DFS Example, AIMA book fig. 3.16 N d t Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 30 Nodes at depth 3 have no successors and M is the only goal node.

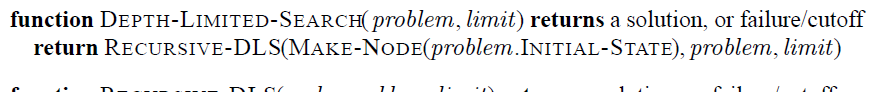


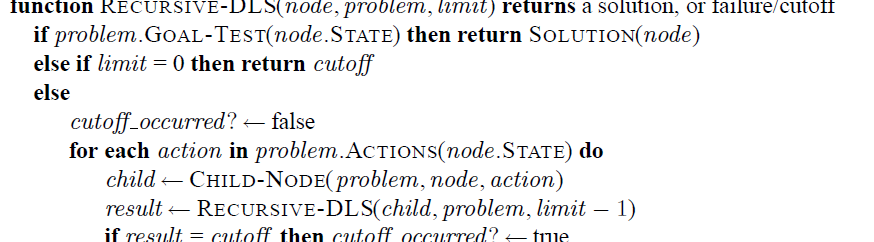


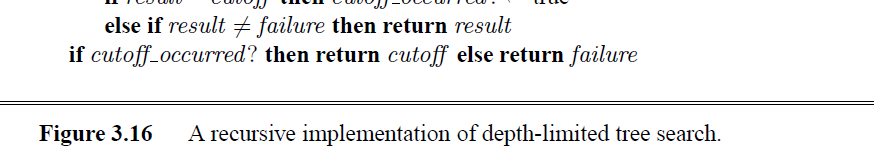


16 • DFS has time complexity O(bm), if m is the maximum depth of any node (may be infinite). • DFS has space complexity of O(b m) Depth-First Search DFS has space complexity of O(b m). • In many AI problems space complexity is more severe than time complexity • Therefore DFS is used as the basic algorithm in • Constraint satisfaction (chapter 6) • Propositional satisfiability (chapter 7) • Logic programming (chapter 9). • Backtracking search, a variant of DFS, uses still less memory, only O(m), by generating successors not at once, but one at a time. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 31 • The failure of DFS in infinite search spaces can be prevented by giving it a search limit l. This approach is called depth limited search 3.4.4 Depth-Limited Search • This approach is called depth-limited search. • Unfortunately, it is not complete if we choose l < d, where d is the depth of the goal node. • This happens easily, because d is unknown. • Depth-limited search has time complexity O(bl). • It has space complexity of O(b l). • However, in some applications we know a depth limit (# nodes in a graph, maximum diameter, …) Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 32

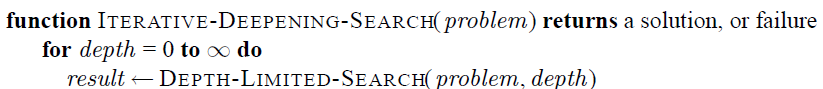
17 Depth-Limited Search Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 33 Example Depth-Limited Search A C B C E D A bottom top status of stack Depth limit = 2 C E C G B C D E F G H I J K L time Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 34 M A variant of depth-limited search with a stack, which does the depth limit test of children before putting them on the stack. This is like the recursive version in the AIMA book.



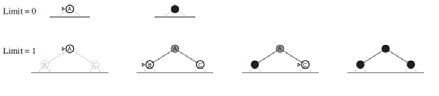


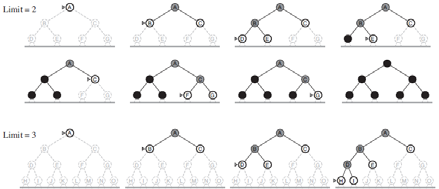


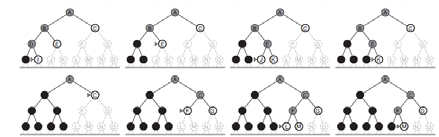
18 3.4.5 Iterative Deepening Search • The iterative deepening search algorithm repeatedly applies depth-limited search with increasing limits. It terminates when a solution is found or if the depth- terminates when a solution is found or if the depth limited search returns failure, meaning that no solution exists. • This search is also frequently called depth-first iterative deepening (DFID) Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 35 Example Iterative Deepening Search A Depth 0 D h 1 B C D E F G H I J K L Depth 1 Depth 2 Depth 3 Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 36 M Empty nodes: unvisited, Grey nodes: frontier (= on stack), Red nodes: closed (= removed from stack) Depth 4



19 Iterative Deepening, AIMA ex. fig. 3.19 Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 37 • Repeatedly visiting the same nodes seems like a waste of time (not space). How costly is this? • This heavily depends on the branching factor b and the f th h t Iterative Deepening Search sparseness of the search tree. • Assume b = 2, d = 10, full binary tree, then • N(IDS) = 11\*20 + 10\*21 + 9\*22 + …+ 1\*210 = 4083 • N(DFS) = 1\*20 + 1\*21 + 1\*22 + …+ 1\*210 = 2047 • i.d. IDS generates at most twice the #nodes of DFS or BFS • Assume b = 10, d = 10, full 10-ary tree, then • N(IDS) = 11\*100+10\*101+9\*102+ +1\*1010 = 12 345 679 011 • N(IDS) = 11 100+10 101+9 102+…+1 1010 = 12.345.679.011 • N(DFS) = 1\*100 + 1\*101+1\*102+…+1\*1010 = 11.111.111.111 • i.d. IDS generates only 11% more nodes than DFS or BFS • IDS is the preferred uninformed search method when the search space is large and the solution depth is unknown. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 38

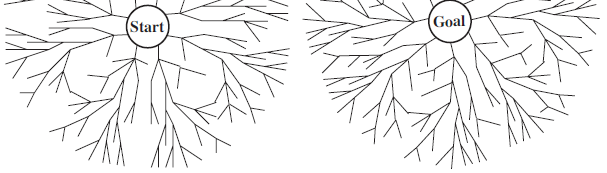


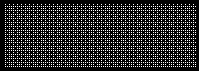




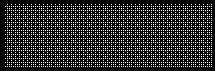
20 3.4.6 Bidirectional search • Advantage: delays exponential growth by reducing the exponent for time and space complexity in half • Disadvantage: at every time point the two fringes must be compared. This requires an efficient hashing data structure • Bidirectional search also requires to search backward (predecessors of a state). This is not always possible Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 39 3.4.7 Comparing Uninformed Search Strategies Criterion Breadth- first Uniform- cost Depth- first Depth- limited Iterative deepen. Bidirectio nal Complete Yesa Yesa,b No No Yesa Yesa,d • Evaluation of tree-search strategies. b is the branching factor, d is the depth of the shallowest solution, m is the maximum depth of the search tree l is the depth limit Time O(bd) O(b1+[C\*/ε]) O(bm) O(bl) O(bd) O(bd/2) Space O(bd) O(b1+[C\*/ε]) O(b m) O(b l) O(b d) O(bd/2) Optimal Yesc Yes No No Yesc Yesc,d Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 40 maximum depth of the search tree, l is the depth limit. • a complete, if b is finite; b complete, if step costs >= ε > 0; c optimal, if step costs are all identical; d if both directions use BFS

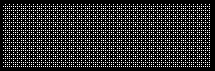






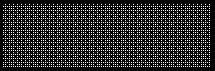


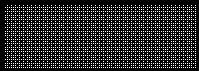


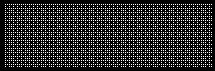




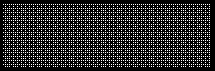






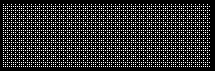










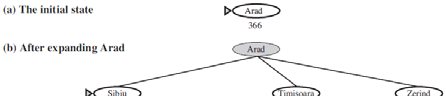


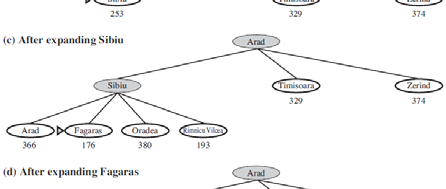
21 • A search strategy that uses problem-specific knowledge can find solutions more efficiently than an uninformed strategy. 3.5 Informed (Heuristic) Search Strat. gy • We use best-first search as general approach • A node is selected for expansion based on an evaluation function f(n). • Informed search algorithms include a heuristic function h(n) as part of f(n). • Often f(n) = g(n) + h(n) • h(n) = estimate of the cheapest cost from the state at node n to a goal state; h(goal) = 0. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 41 • Greedy best-first search tries to expand the node that it estimates as being closest to the goal. • It uses only the heuristic function h(n), • f(n) = h(n) 3.5.1 Greedy Best-First Search • f(n) = h(n). • We use a straight-line distance heuristic hSLD for the route-finding in Romania: straigt line distances to the goal Bucharest (as given in the table below). hSLD ( ) = Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 42

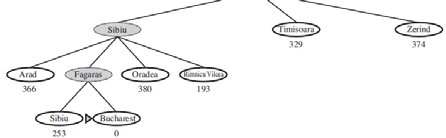




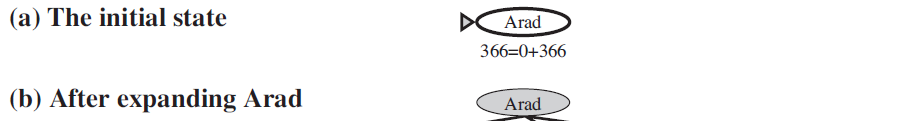
22 Greedy Best-First Search Stages in a greedy best-first tree search for Bucharest with the straight-line distance heuristic hSLD. Nodes are labeled with their hSLD - values. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 43 • The algorithm is called “greedy”, because in each step the algorithm greedily tries to get as close to the goal as possible Greedy Best-First Search close to the goal as possible. • GBFS as tree is not complete, even in a finite state space (it may not find the shortest solution to the goal). • The graph search version of GBFS is complete in finite search spaces but not in infinite ones in finite search spaces, but not in infinite ones. • Suggestion for improvement: Use the accumulated path distance g(n) plus a heuristic h(n) as cost function f(n). This leads to A\* Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 44

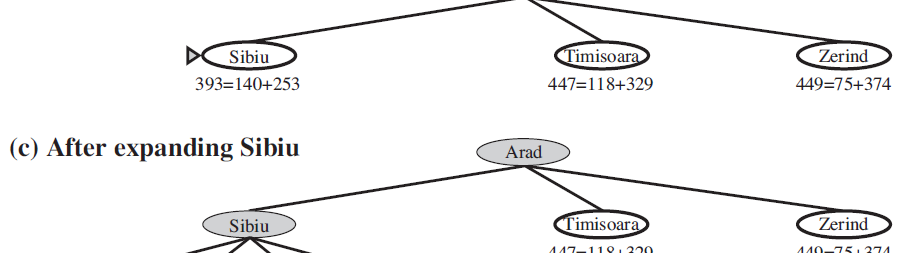


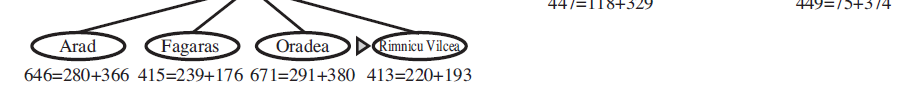




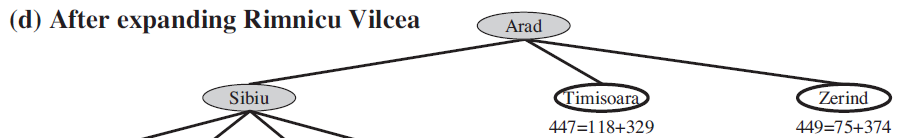
23 • A\* search is one of the most popular AI search algorithms • It combines two path components: 3.5.2 A\* Search It combines two path components: • g(n), the travelled path component from the start node to the node n, and • h(n), a heuristic component, the estimated cost to reach the goal. • f(n) = g(n) + h(n) • f(n) = estimated cost of the cheapest solution to • f(n) = estimated cost of the cheapest solution to the goal passing through node n. • Under certain conditions for h(n), A\* is both complete and optimal. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 45 A\* Search Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 46 Stages in an A∗search for Bucharest. Nodes are labeled with f = g + h. The h values are the straight-line distances to Bucharest

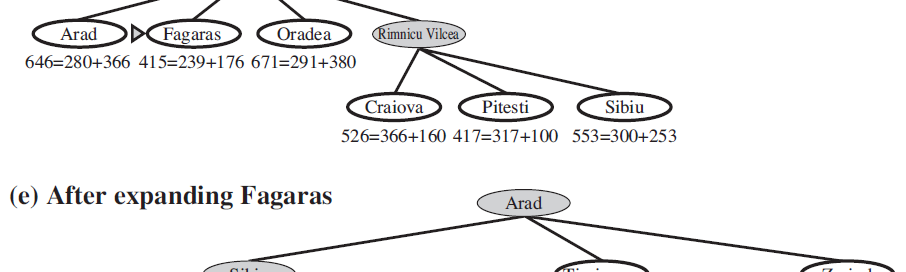


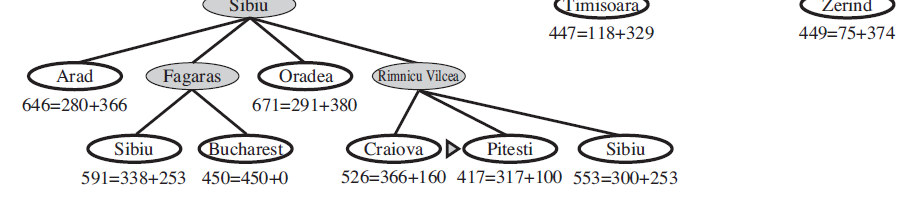


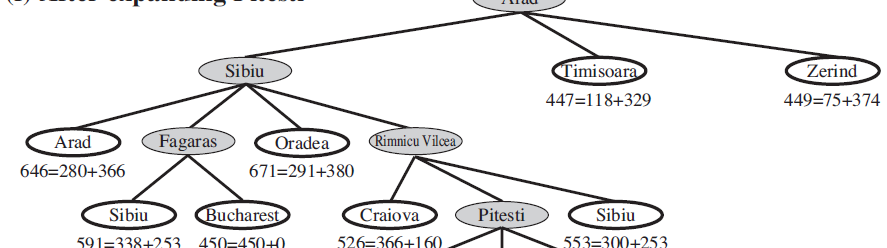


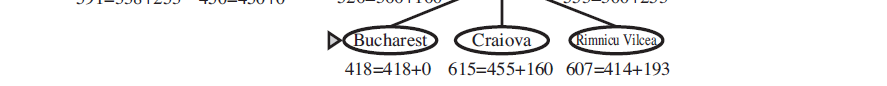
24 A\* Search Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 47 A\* Search Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 48







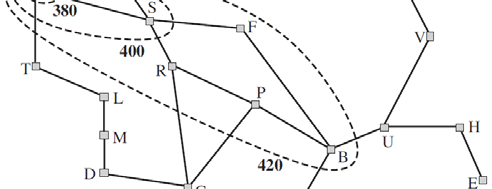




25 In order for A\* to be optimal • h(n) must be admissible, i.e. it never overestimates the cost to reach the goal. f( ) ( ) ( ) Conditions for Optimality of A\* • Then, as a consequence, f(n) = g(n) + h(n) never overestimates the true cost of a solution along the current path through n. • h(n) must be consistent (monotonic) in graph search, i.e. for every node n and every successor n’ of n generated by action a, ( ) ( ') ( ') h n c n a n h n ≤ + n n’ a • This is a form of the triangle inequality. • Every consistent heuristic is also admissible. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 49 ( ) ( , , ) ( ) h n c n a n h n ≤ + n g n • The graph-search version of A\* is optimal if h(n) is consistent. We show this in 2 steps: 1. If h(n) is consistent, then the values of f(n) along any path are nondecreasing Optimality of A\* (Graph Search) path are nondecreasing. • Suppose n’ is a succ. of n, then • Therefore 2. Whenever A\* selects a node n for expansion, the optimal path to that node has been found. ( ') ( ') ( ') ( ) ( , , ') ( ') ( ) ( ) ( ) f n g n h n g n c n a n h n g n h n f n = + = + + ≥ + = ( ') ( ) ( , , ') g n g n c n a n = + • otherwise there must be another frontier node n’ on the optimal path from start node to n; as f is nondecreasing along any path, n’ would have lower f- cost than n and would have been selected first. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 50

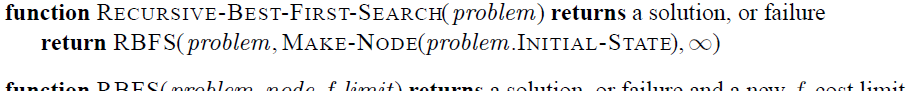
26 • As f-costs are nondecreasing along any path we can draw contours in the state space. • See figure 3.25 on the following page Contours in State Space • With uniform-cost search (A\* with h(n) = 0), contours are circular around the start state. • With more accurate heuristics, the bands will stretch towards the goal state and become more narrow • If C\* is the cost of the optimal solution path, then • A\* expands all nodes with f(n) < C\* A\* d d ith f( ) C\* b f fi di th l • A\* may expand some nodes with f(n) = C\* before finding the goal state. • Completeness requires that there are only finitely many nodes with cost less than or equal to C\*, which is true if all step costs exceed some finite ε and if b is finite. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 51 Contours in State Space Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 52 Map of Romania showing contours at f = 380, f = 400, and f = 420, with Arad as the start state. Nodes inside a given contour have f-costs less than or equal to the contour value.

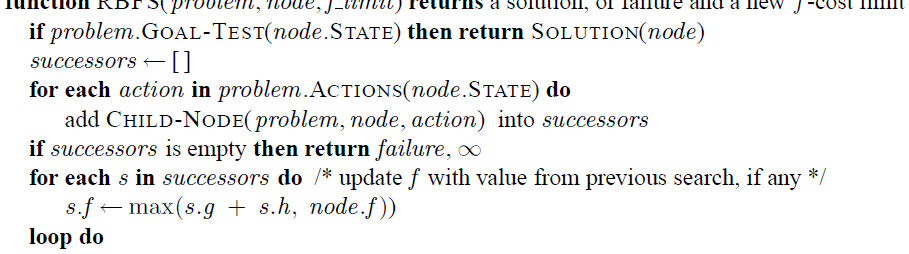


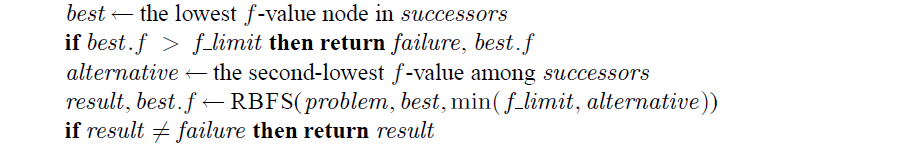


27 • A\* is optimally efficient for any given consistent heuristic, i.e. no other optimal algorithm is guaranteed to expand fewer nodes than A\* A\* is Optimally Efficient guaranteed to expand fewer nodes than A (except possibly for nodes with f(n) = C\*). • This is because any algorithm that does not expand all nodes with f(n) < C\* may miss the optimal solution. • So A\* is complete, optimal and optimally efficient, but it still is not the solution to all search , problems: • As A\* keeps all generated solutions in memory it runs out of space long before it runs out of time. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 53 • How to reduce memory requirements of A\*? • Use iterative deepening A\* (IDA\*) C ff f( ) ( ) ( ) 3.5.3 Memory-bounded Heuristic Search • Cutoff used is f(n) = g(n) + h(n) rather than the depth. • At each iteration, the cutoff value is the smallest f-cost of any node that exceeded the cutoff on the previous iteration. • This works well with unit step costs. • It suffers from severe problems with real-valued costs. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 54

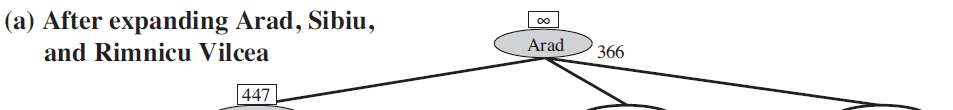
28 • Simple recursive algorithm that tries to mimic the operation of standard best-first search in linear space (see algorithm on next page) Recursive Best-First Search (RBFS) • It is similar to recursive DFS, but uses a variable f\_limit to keep track of the best alternative path available from any ancestor of the current node. • If the current node exceeds f\_limit, the recursion unwinds back to the alternative path. Then, RBFS replaces the f-value of each node on the path with a backed-up value, the best value of its children. • In this way, RBFS remembers the f-value of the best leaf in the forgotten subtree and may decide to re- expand it later Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 55 Recursive Best-First Search Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 56

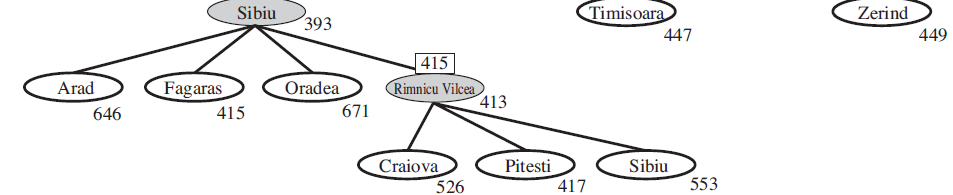


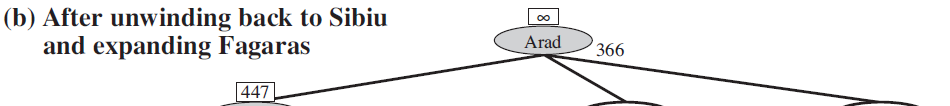




29 RBFS Search AIMA example f\_limit f\_limit best alternative best alternative best > f\_limit f\_limit must continue with alternative Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 57 Stages in an RBFS search for the shortest route to Bucharest. The f-limit value for each recursive call is shown on top of each current node, and every node is labeled with its f-cost. (a) The path via Rimnicu Vilcea is followed until the current best leaf (Pitesti) has a value that is worse than the best alternative path (Fagaras). RBFS Search AIMA example f\_limit f\_limit best leaf of subtree is backed up new f\_limit Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 58 (b) The recursion unwinds and the best leaf value of the forgotten subtree (417) is backed up to Rimnicu Vilcea; then Fagaras is expanded, revealing a best leaf value of 450.

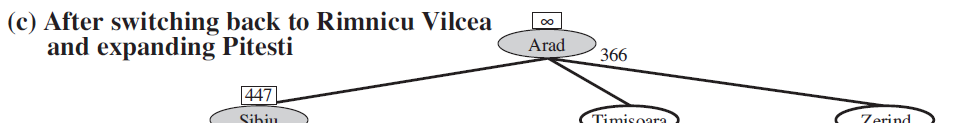


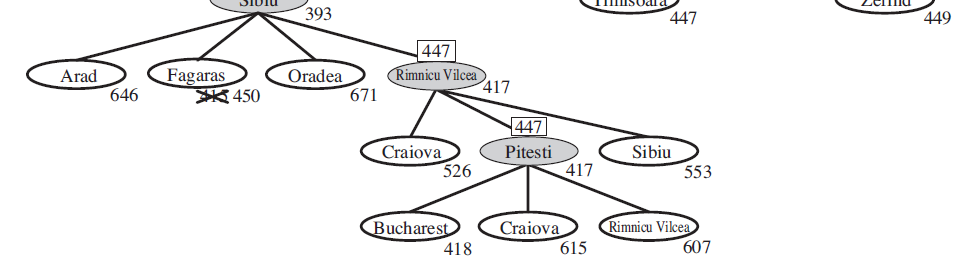






30 RBFS Search AIMA example f\_limit f\_limit f\_limit best best Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 59 (c) The recursion unwinds and the best leaf value of the forgotten subtree (450) is backed up to Fagaras; then Rimnicu Vilcea is expanded. This time, because the best alternative path (through Timisoara) costs at least 447, the expansion continues to Bucharest. • IDA\* and RBFS use too little memory • IDA\* retains only 1 number between iterations (f-cost limit) • RBFS retains more information, but uses only linear space Simplified Memory-bounded A\* (SMA\*) RBFS retains more information, but uses only linear space • SMA\* proceeds just like A\*, expanding the best leaf until memory is full. Then it must drop an old node. • SMA\* always drops the worst leaf node (highest f- value). (In case of ties with same f-value, SMA\* expands the newest leaf and deletes the oldest leaf) • Like RBFS, SMA\* then backs up the value of the forgotten node to its parent. In this way the ancestor of a forgotten subtree knows the quality of the best path in that subtree. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 60





31 • The quality of any heuristic search algorithm depends on its heuristic • Two admissible heuristics for the 8-puzzle: 3.6 Heuristic Functions Two admissible heuristics for the 8 puzzle: • h1 = # misplaced tiles • h2 = sum of Manhattan dist. of all tiles to goal position • h2 is better for search than h1. See Fig. 3.29 in book. • The effective branching factor b\* may characterize the quality of a heuristics: • If N is the # nodes generated by a search algorithm and d is the solution depth, then b\* ist the branching factor, which a uniform tree of depth d would need to have to contain N+1 nodes. Thus Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 61 2 1 1 \* ( \*) ... ( \*)d N b b b + = + + + + • Search is used in environments which are deterministic, observable, static and completely known. • First a goal must be identified and a well-defined problem must be formulated Chapter 3 Summary 1 problem must be formulated. • A problem consists of five parts: the initial state, a set of actions, a transition function describing the results of actions, a goal test function and a path cost function. • The environment of the problem is represented by a state space. A path from initial state to a goal state is a solution. • Search algorithms treat states and actions as atomic Search algorithms treat states and actions as atomic, they do not consider their internal structure. • The Tree-Search algorithm considers all possible paths to find a solution, whereas Graph-Search avoids redundant paths. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 62

32 • Search algorithms are judged on the basis of completeness, optimality, time and space complexity. • Uninformed search methods have access to only the bl d fi iti ( h i ti ) Chapter 3 Summary 2 problem definition (no heuristics). • Breadth-first search expands the shallowest nodes first; it is complete, optimal for unit step costs, but has exponential space complexity. • Uniform-cost search expands the node with the longest path cost, g(n), and is optimal for general step costs. • Depth-first search expands the deepest unexpanded node first. It is neither complete nor optimal but has linear space complexity is neither complete nor optimal but has linear space complexity. • Iterative deepening search calls DFS with increasing depth limits. It is complete, optimal for unit step costs, has time complexity comparable to DFS and linear space complexity. • Bidirectional search may reduce time complexity enormously, but is not always applicable and may require too much space. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 63 • Informed search methods have access to a heuristic function h(n) that estimates the cost of a solution from n. • The generic best-first search selects a node for expansion according to an evaluation function. Chapter 3 Summary 3 g • Greedy BFS expands nodes with minimal h(n). It is not optimal. • A\* search expands nodes with minimal f(n) = g(n) + h(n). It is complete and optimal, if h(n) is admissible (for tree-search) or consistent (for graph-search). Space complexity is exponential • RBFS (recursive best-first search) and SMA\* (simplified memory-bounded A\*) are robust, optimal search algorithms that use limited amounts of memory • The performance of heuristic search algorithms depends h li f h i h i i f i on the quality of their heuristic function. • Methods to construct good heuristics are: relaxing the problem definition, using a pattern DB with precomputed solution costs, or automated learning from experience. Zell: Artificial Intelligence (after Russel/Norvig, 3rd Ed.) 64