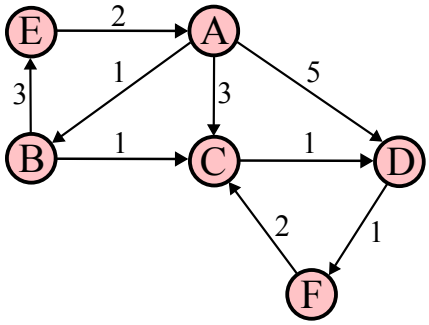
FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE
Exercise 2a: Uninformed Search – Problems Winter Semester 2023/24
Sebastian Mair

1 Presented Problems

Problem 2.1: Search Algorithms Basics

Consider the following graph. We start from A and our goal is F. Path costs are shown on the arcs. If there is no clear preference, which child node should be visited next during expansion, we choose the node that is first alphabetically.



Perform Breadth-First, Depth-First and Uniform-Cost search. For each step, write down the node which is currently expanded, the frontier, and the reached set or lookup table (if applicable).
Hint: You can generally write $X(C, P)$ for a node n, where $X =$ `n.STATE`, $C =$ `n.PATH-COST`, and $P =$ `n.PARENT.STATE`. You can omit C for Breadth-First and Depth-First search (as it is not needed during search).

Problem 2.2: Application of Search Algorithms: Transport

I have bought 120 bricks at the hardware store, which I need to transport to my house. I have the following actions a:

1. take the bus (b) – I can carry up to 30 bricks, costing 3 €
2. take my car (c) – it can carry up to 100 bricks, costing 5 €
3. take a delivery (d) – it delivers all the bricks, costing 29 €

Assume that I apply actions in the order: bus, car, delivery; and we start at the initial node s. For clarity, we name states after the actions taken to get to them (e.g. after taking the bus twice, the state is labeled with bb); and the order of actions reaching a state is not important (e.g. the states bc and cb are considered to be the same).

Problem 2.2.1: Assuming I want to make as few trips as possible, which search method should I use and what is the resulting solution?

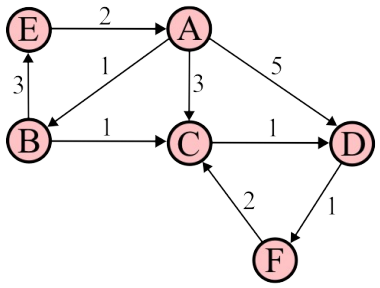
Problem 2.2.2: Assuming I want to minimize the cost, which search method should I use and what is the resulting solution?

Problem 2.2.3: Perform depth-first search.

Problem 2.2.4: Perform depth-first search again, but now assume I explore actions in the order: delivery, car, bus.

Problem 2.1: Search Algorithms Basics

Consider the following graph. We start from A and our goal is F. Path costs are shown on the arcs. If there is no clear preference, which child node should be visited next during expansion, we choose the node that is first alphabetically.



A → F

FiFO LIFO

Perform Breadth-First, Depth-First and Uniform-Cost search. For each step, write down the node which is currently expanded, the frontier, and the reached set or lookup table (if applicable).

Hint: You can generally write $X(C, P)$ for a node n, where $X =$ `n.STATE`, $C =$ `n.PATH-COST`, and $P =$ `n.PARENT.STATE`. You can omit C for Breadth-First and Depth-First search (as it is not needed during search).

BFS FiFO

node	frontier	reached set
A(-)	∅	A(-)
A(-)	~~B(A)~~ C(A) D(A)	B(A) C(A) D(A)
B(A)	~~C(A)~~ D(A) E(B)	E(B)
C(A)	~~D(A)~~ E(B)	
D(A)	E(B)	

Find

frontier 先进先出

DFS LiFO

node	frontier	reached set
A(-)	∅	A(-)
A(-)	B(A) C(A) ~~D(A)~~	B(A) C(A) D(A)
D(A)	B(A) C(A) ~~F(D)~~	F(D)
F(D)	B(A) C(A)	

Find

UCS

node	frontier	Reached set
A(0,-)	∅	A(0,-)
A(0,-)	~~B(1,A)~~ C(3,A) D(5,A)	B(1,A) ~~C(3,A)~~ ~~D(5,A)~~
B(1,A)	~~C(2,B)~~ E(4,B) D(5,A)	C(2,B) E(4,B)
C(2,B)	~~D(3,C)~~ E(4,B)	D(3,C)
D(3,C)	~~E(4,B)~~ F(4,D)	F(4,D)
E(4,B)	~~F(4,D)~~	
F(4,D)	∅	

Problem 2.2: Application of Search Algorithms: Transport

I have bought 120 bricks at the hardware store, which I need to transport to my house. I have the following actions a:

1. take the bus (b) – I can carry up to 30 bricks, costing 3 €
2. take my car (c) – it can carry up to 100 bricks, costing 5 €
3. take a delivery (d) – it delivers all the bricks, costing 29 €

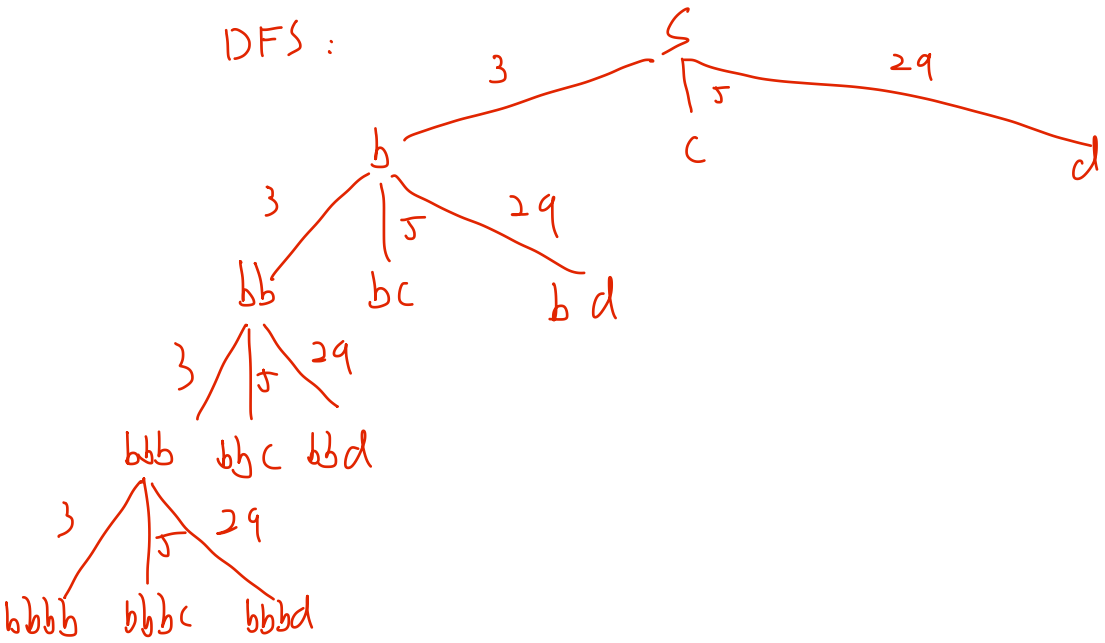
Assume that I apply actions in the order: bus, car, delivery; and we start at the initial node s. For clarity, we name states after the actions taken to get to them (e.g. after taking the bus twice, the state is labeled with bb); and the order of actions reaching a state is not important (e.g. the states bc and cb are considered to be the same).

Problem 2.2.1: Assuming I want to make as few trips as possible, which search method should I use and what is the resulting solution? BFS

Problem 2.2.2: Assuming I want to minimize the cost, which search method should I use and what is the resulting solution? UCS

Problem 2.2.3: Perform depth-first search.

Problem 2.2.4: Perform depth-first search again, but now assume I explore actions in the order: delivery, car, bus.



2 Additional Problems

Problem 2.3: Application of Search Algorithm: Train Journey

From Problem 2.6 (of Exercise 2b), perform Breadth-First, Depth-First, and Uniform-Cost search for both time and price cost. For each step, write down the node which is currently expanded, the frontier, and the reached set or lookup table (if applicable).
Hint: You can again write $X(C, P)$ for a node n, where $X =$ `n.STATE`, $C =$ `n.PATH-COST`, and $P =$ `n.PARENT.STATE`. You can omit C for Breadth-First and Depth-First search (as it is not needed during search).

Problem 2.4: General Questions on Uninformed Search

Problem 2.4.1: (from *Russell & Norvig 3ed.* q. 3.18) Describe a state transition graph in which Iterative Deepening Search performs much worse (in time) than Depth-First Search (e.g. $\mathcal{O}(n^2)$ vs. $\mathcal{O}(n)$).

Problem 2.4.2: (from *Russell & Norvig 3ed.* q. 3.15) Consider a state transition graph where the start state is the number 1 and the successor function for state n returns two states, numbers $2n$ and $2n + 1$.

a. Draw the portion of the state transition graph for states 1 to 15.

b. Suppose the goal state is 11. List the order in which nodes will be generated (not expanded) for Breadth-First search, Depth-Limited search with limit 2, and Iterative Deepening search. Would bidirectional search be appropriate for this problem?

c. What is the branching factor in each direction of the bidirectional search?

d. Does the answer to **c.** suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

Problem 2.4.3: (from *Russell & Norvig, 2ed.* q. 3.6) Does a finite state transition graph always lead to a finite search tree? How about a finite state transition graph that is a tree?