Intelligent Systems/ Sistemas Inteligentes

Búsqueda en grafos/ Graph search

Antonio Bahamonde
Departamento de informática

Universidad de Oviedo en Gijón

Contents

(R&N, Ch-3)

- Problem-solving agents
- Problem formulation
- Example problems
- Basic search algorithms
  - uninformed
  - informed

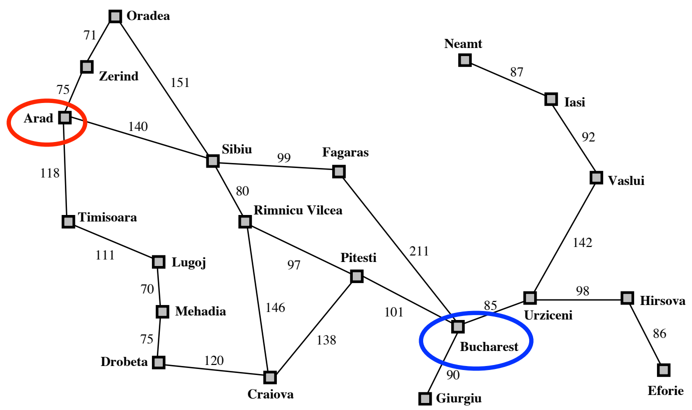
Introduction

Intelligent agents are supposed to maximize their performance measure. This is sometimes simplified if the agent can adopt a **goal** and aim at satisfying it.

This chapter describes one kind of goal-based agent called a **problem-solving agent**. Problem-solving agents think about the world using **atomic** representations: states of the world are considered as wholes.

Example: route in a map

Find a path on the map from Arad to Bucharest (Rumania)

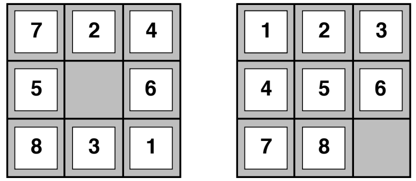


Formal Problem Formulation

A problem is defined by these items:
- initial state e.g., “at Arad”
- successor function $S(x)$ = set of action–state pairs
  - e.g., $S(\text{Arad}) = \{\langle \text{Arad} \rightarrow \text{Zerind}, \text{Zerind} \rangle, \ldots\}$
- goal test, can be
  - x = “at Bucharest”?
- path cost (additive)
  - sum of distances, number of 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

The 8-puzzle

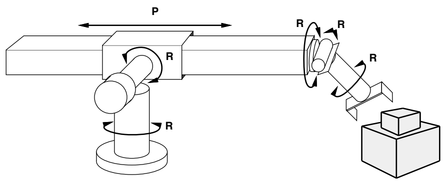


Start State   Goal State

states??: integer locations of tiles
actions??: move blank left, right, up, down
goal test??: = goal state (given)
path cost??: 1 per move

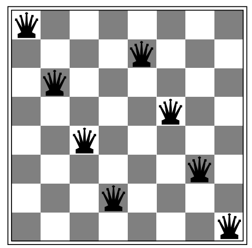
[Note: optimal solution of n-Puzzle family is NP-hard]

Robotic assembly



states??: real-valued coordinates of robot joint angles parts of the object to be assembled
actions??: continuous motions of robot joints
goal test??: complete assembly
path cost??: time to execute

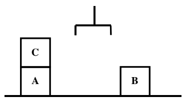
8-Queens problem



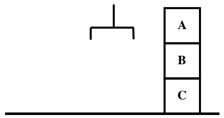
States: Any arrangement of 0 to 8 queens on the board is a state.
Initial state: No queens on the board.
Actions: Add a queen to any empty square.
Goal test: 8 queens are on the board, none attacked.

Planning

Estado Inicial



Estado Objetivo



SOBRE(C,A) SOBRE_LA_MESA(A) LIBRE(C)

MANO_LIBRE SOBRE_LA_MESA(B) LIBRE(B)

SOBRE(B,C) SOBRE(A,B)

Reglas

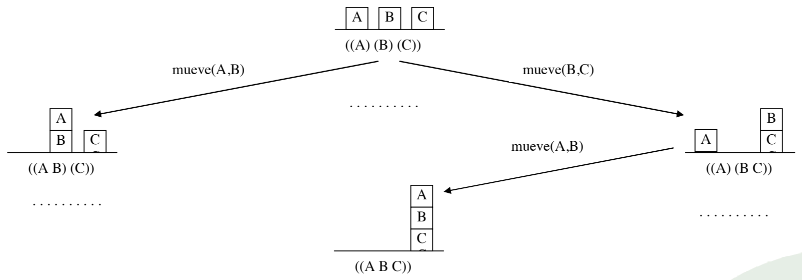
COGER(X)
P y B: SOBRE_LA_MESA(X), LIBRE(X), MANO_LIBRE
A: COGIENDO(X)

POSAR(X)
P Y B: COGIENDO(X)
A: SOBRE_LA_MESA(X), LIBRE(X), MANO_LIBRE

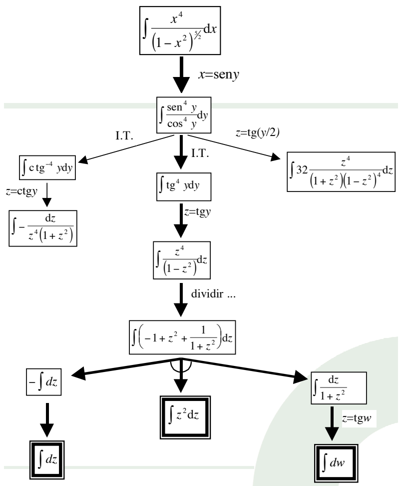
APILAR(X,Y)
P Y B: COGIENDO(X), LIBRE(Y)
A: MANO_LIBRE, SOBRE(X,Y), LIBRE(X)

DESAPILAR(X,Y)
P Y B: MANO_LIBRE, LIBRE(X), SOBRE(X,Y)
A: COGIENDO(X), LIBRE(Y)

Planning



Symbolic integration

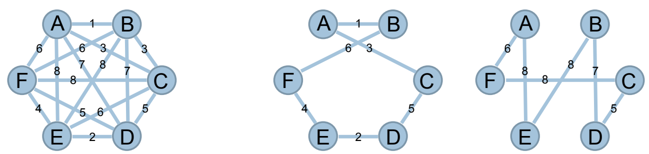


Traveling Salesman Problem (TSP)

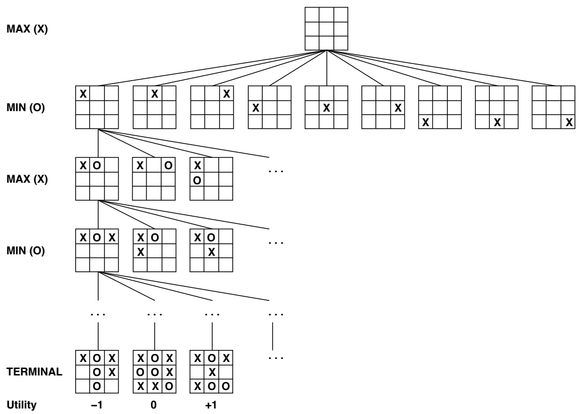
Is a touring problem in which each city must be visited exactly once. The aim is to find the *shortest* tour.

The problem is known to be NP-hard, but an enormous amount of effort has been expended to improve the capabilities of TSP algorithms.

These algorithms have been used for tasks such as planning movements of automatic of stocking machines on shop floors.



Game tree (2-player, deterministic, turns)



Searching for Solutions

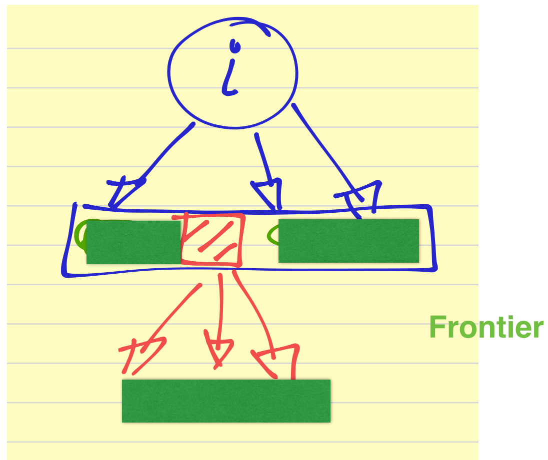
- Basic algorithms: trees, graphs
- Uninformed:
  - breadth first
  - uniform cost
  - depth first
  - (bidirectional)
- Informed
  - A*

Measuring problem-solving performance

We will evaluate an algorithm's performance in four ways. These provide us with criteria to choose among them.

- **Completeness**: Is the algorithm guaranteed to find a solution when there is one?
- **Optimality**: Does the strategy find the optimal solution?
- **Time complexity**: How long does it take to find a solution?
- **Space complexity**: How much memory is needed to perform the search?

Basic algorithms: trees



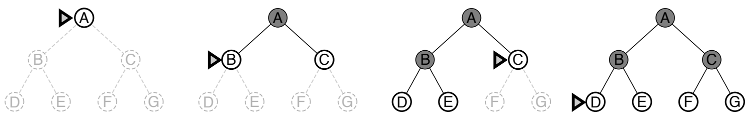
Basic algorithms: trees

function TREE-SEARCH(*problem*) **returns** a solution, or failure
initialize the frontier using the initial state of *problem*
loop do
if the frontier is empty **then return** failure
choose a leaf node and remove it from the frontier
if the node contains a goal state **then return** the corresponding solution
expand the chosen node, adding the resulting nodes to the frontier

Basic algorithms: graphs

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
initialize the frontier using the initial state of *problem*
initialize the explored set to be empty
loop do
if the frontier is empty **then return** failure
choose a leaf node and remove it from the frontier
if the node contains a goal state **then return** the corresponding solution
add the node to the explored set
expand the chosen node, adding the resulting nodes to the frontier
only if not in the frontier or explored set

Uninformed: breadth first



The memory requirements are a bigger problem for breadth-first search than is the execution time.

Exponential-complexity search problems cannot be solved by uninformed methods for any but the smallest instances.

Breadth-first search is only optimal when all step costs are equal.

Uninformed: Uniform cost

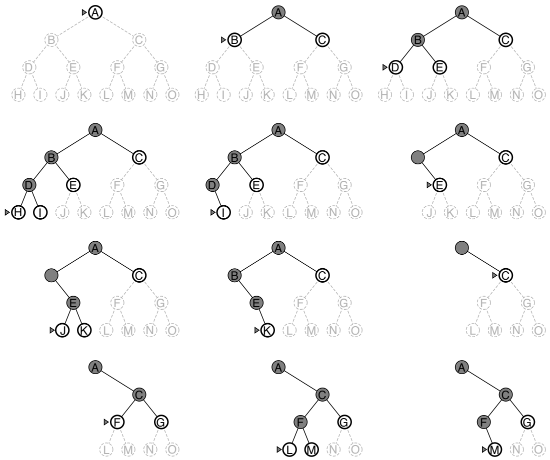
It is an extension of breadth-first.

It is optimal with any step cost function.

Instead of expanding the shallowest node, **uniform-cost search** expands the node n with the *lowest path cost* g(n). This is done by storing the frontier as a priority queue ordered by g.

The algorithm is identical to the general graph search algorithm, except for the use of a priority queue and the addition of an extra check in case a shorter path is discovered to a frontier state

Uninformed: Depth first



Uninformed: Backtracking

A variant of depth-first search that uses still less memory.

Only one successor is generated at a time rather than all successors; each partially expanded node remembers which successor to generate next.

In this way, only O(m) memory is needed rather than O(bm).

Uninformed: Comparison

Criterion	Breadth-First	Uniform-Cost	Depth-First
Complete?	Yes^{a}	Yes^{a,b}	No
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$
Optimal?	Yes^{c}	Yes	No

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;

(a) complete if b is finite;
(b) complete if step costs ≥ ε for positive ε;
(c) optimal if step costs are all identical

A*

Informed: A*

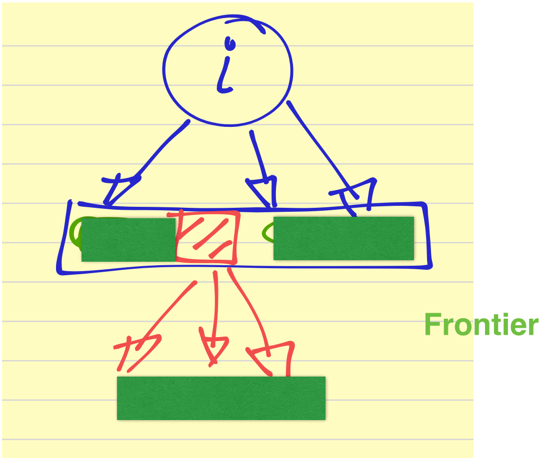
An **informed search** strategy uses problem-specific knowledge beyond the definition of the problem itself

They can find solutions more efficiently than an uninformed strategies.

The general approach we will consider is called **best-first search**. An instance of the general graph search algorithm in which a node is selected for expansion based on an **evaluation function**, f(n).

The evaluation function is construed as a cost estimate, so the node with the *lowest* evaluation is expanded first. The choice of f determines the search strategy.

Basic algorithms: trees



Informed: A*

The implementation of best-first search is identical to that for uniform-cost search, except for the use of f instead of g to order the priority queue.

The most widely-known form of best-first search is called **A* search** (pronounced "A-star search").

It evaluates nodes using

$$f(n) = g(n) + h(n)$$

h is the heuristic function

g(n) gives the path cost from the start node to node n, and
h(n) is the estimated cost of the cheapest path from n to the goal,
we have
f(n) = estimated cost of the cheapest solution through n.

A* is identical to Uniform-cost except that it uses g + h instead of g.

Informed: A*

Provided that the heuristic function h(n) satisfies certain conditions, A* search is both complete and optimal.

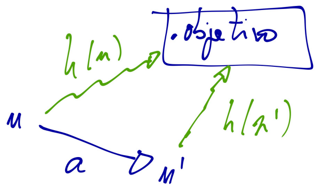
An **admissible** heuristic is one that *never overestimates* the cost to reach the goal.

Admissible heuristics are by nature optimistic, because they think the cost of solving the problem is less than it actually is.

admissible

air distance

consistent



A heuristic h(n) is **consistent** (**monotone**) if, for every node n and every successor n' of n generated by any action a, the estimated cost of reaching the goal from n is no greater than the step cost of getting to n' plus the estimated cost of reaching the goal from n:

$h(n) \leq c(n, a, n') + h(n')$

This is a form of the general **triangle inequality**

Informed: A*

Every consistent heuristic is also admissible.

The tree-search version of A* is optimal if h(n) is admissible, while the graph-search version is optimal if h(n) is consistent.

A* is complete if all step costs exceed some finite ε and if b is finite.

Informed: A*

The complexity of A* often makes it impractical to insist on finding an optimal solution. (Space complexity is also impractical)

One can use variants of A* that find **suboptimal** solutions quickly, or one can sometimes design heuristics that are more accurate but not strictly admissible.

In any case, the use of a good heuristic still provides enormous savings compared to the use of an uninformed search.

Search example

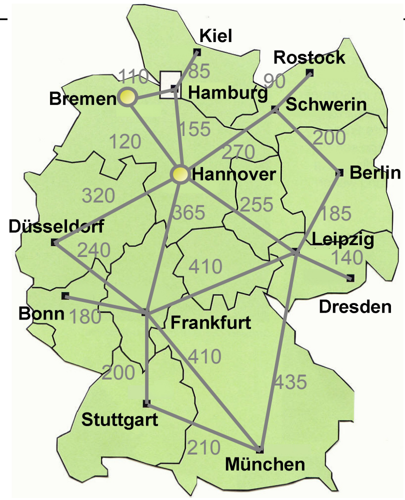
Find the shortest route between Hannover and München

- With uniform cost
- A * with heuristic the air distance

Search example

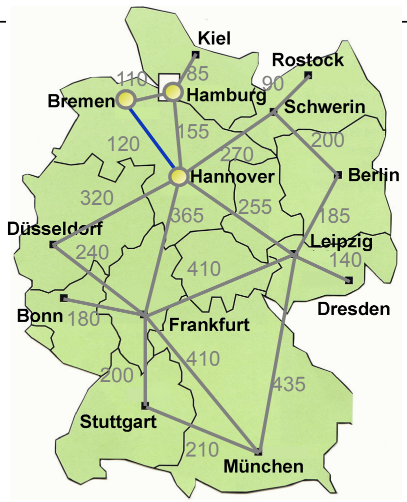
Hannover	0
Bremen	∞
Hamburg	∞
Kiel	∞
Leipzig	∞
Schwerin	∞
Duesseldorf	∞
Rostock	∞
Frankfurt	∞
Dresden	∞
Berlin	∞
Bonn	∞
Stuttgart	∞
München	∞

Search example



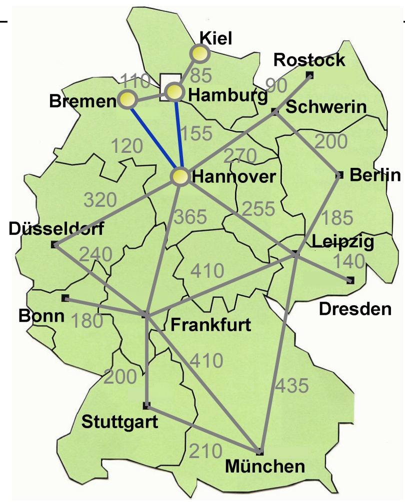
Hannover	0
Bremen	120
Hamburg	155
Kiel	∞
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	∞
Frankfurt	365
Dresden	∞
Berlin	∞
Bonn	∞
Stuttgart	∞
Muenchen	∞

Search example



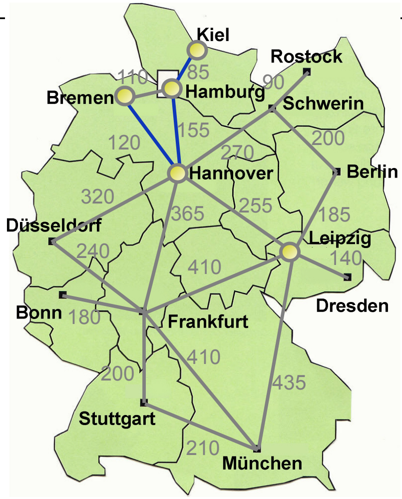
Hannover	0
Bremen	120
Hamburg	155
Kiel	∞
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	∞
Frankfurt	365
Dresden	∞
Berlin	∞
Bonn	∞
Stuttgart	∞
Muenchen	∞

Search example



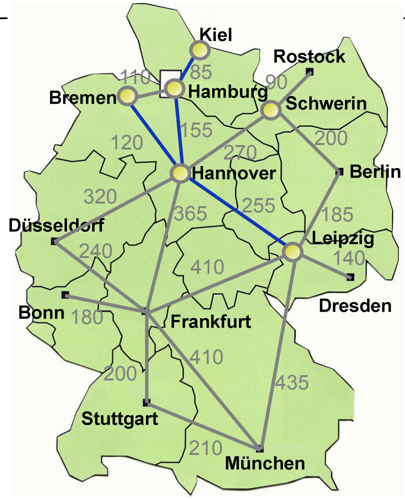
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	∞
Frankfurt	365
Dresden	∞
Berlin	∞
Bonn	∞
Stuttgart	∞
Muenchen	∞

Search example



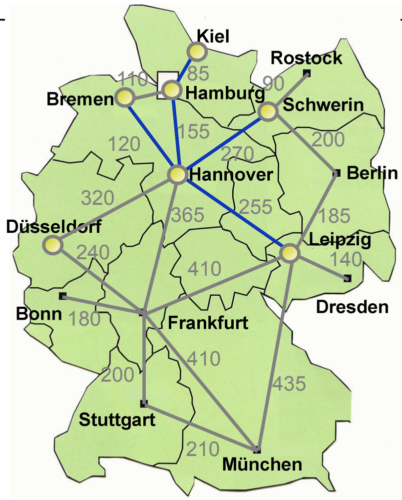
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	∞
Frankfurt	365
Dresden	∞
Berlin	∞
Bonn	∞
Stuttgart	∞
Muenchen	∞

Search example



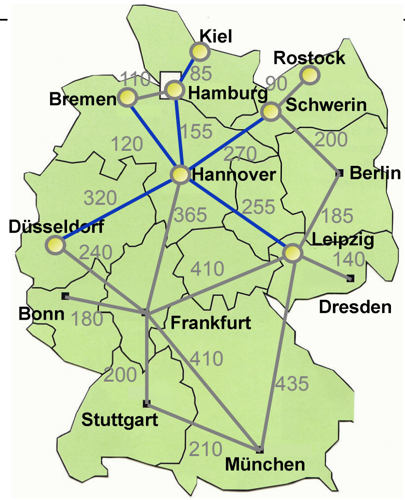
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	∞
Frankfurt	365
Dresden	395
Berlin	440
Bonn	∞
Stuttgart	∞
Muenchen	690

Search example



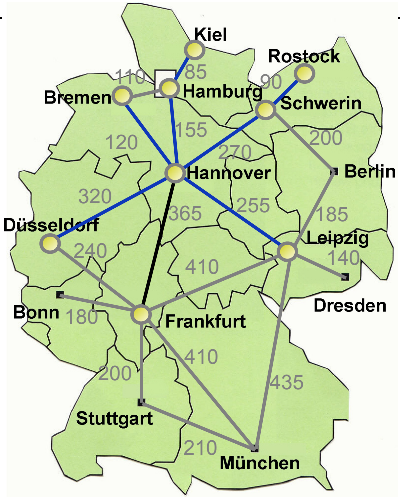
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	360
Frankfurt	365
Dresden	395
Berlin	440
Bonn	∞
Stuttgart	∞
Muenchen	690

Search example



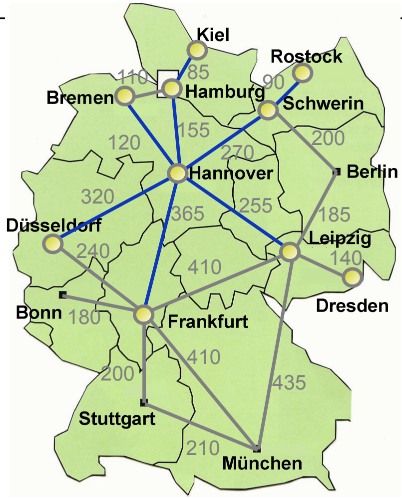
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	360
Frankfurt	365
Dresden	395
Berlin	440
Bonn	∞
Stuttgart	∞
Muenchen	690

Search example



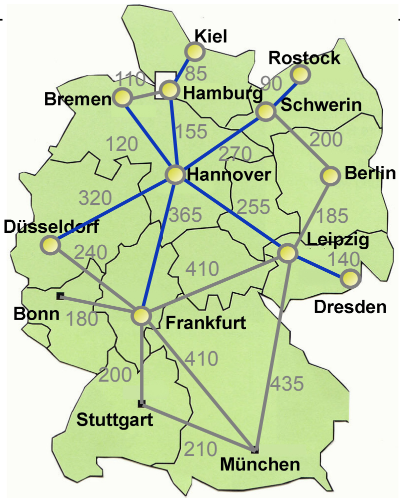
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	360
Frankfurt	365
Dresden	395
Berlin	440
Bonn	∞
Stuttgart	∞
Muenchen	690

Search example



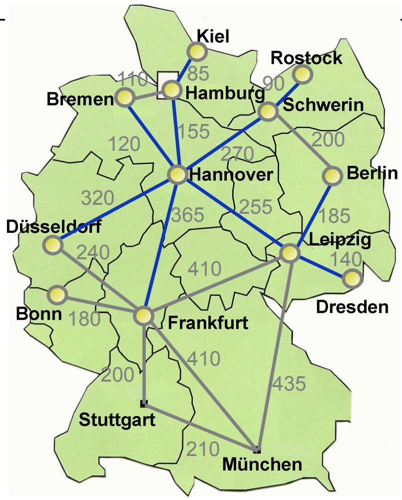
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	360
Frankfurt	365
Dresden	395
Berlin	440
Bonn	545
Stuttgart	565
Muenchen	690

Search example



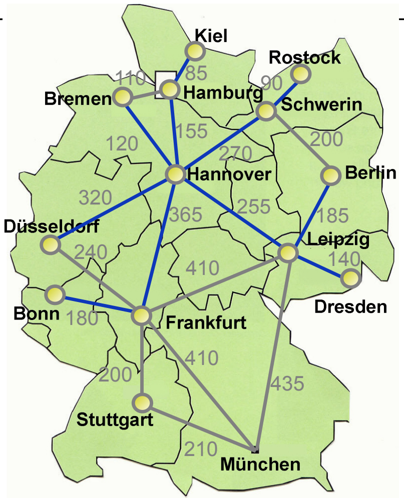
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	360
Frankfurt	365
Dresden	395
Berlin	440
Bonn	545
Stuttgart	565
Muenchen	690

Search example



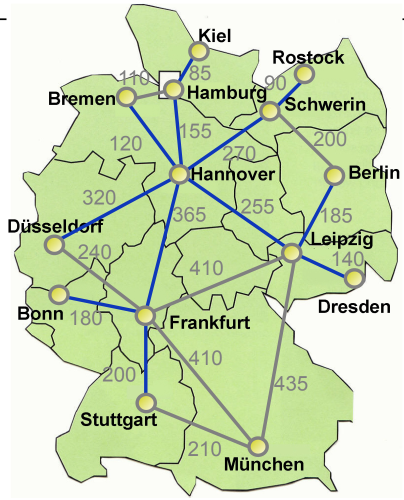
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	360
Frankfurt	365
Dresden	395
Berlin	440
Bonn	545
Stuttgart	565
Muenchen	690

Search example



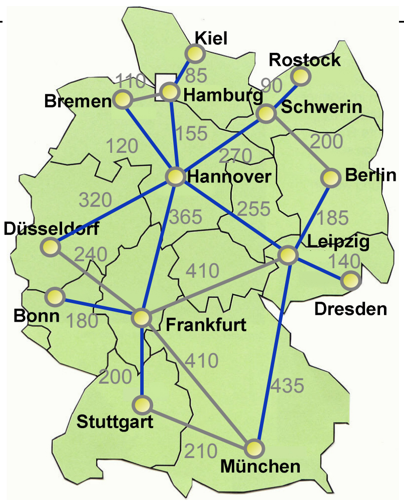
Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	360
Frankfurt	365
Dresden	395
Berlin	440
Bonn	545
Stuttgart	565
Muenchen	690

Search example



Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	360
Frankfurt	365
Dresden	395
Berlin	440
Bonn	545
Stuttgart	565
Muenchen	690

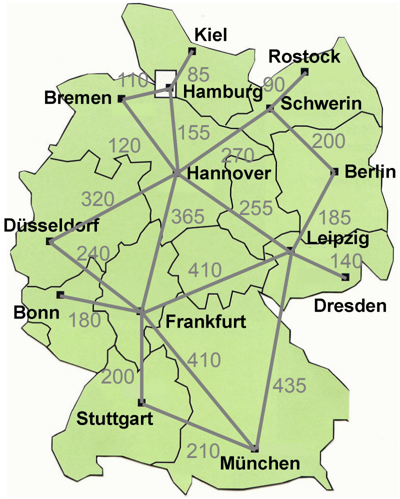
Search example

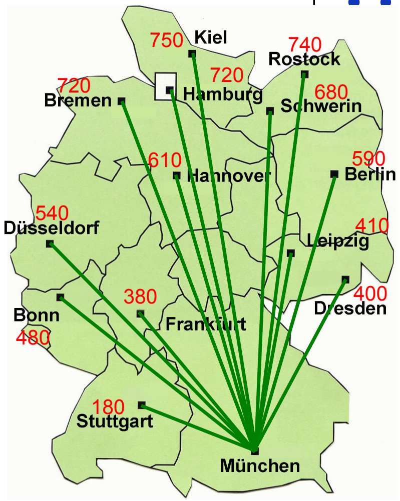


Hannover	0
Bremen	120
Hamburg	155
Kiel	240
Leipzig	255
Schwerin	270
Duesseldorf	320
Rostock	360
Frankfurt	365
Dresden	395
Berlin	440
Bonn	545
Stuttgart	565
Muenchen	690

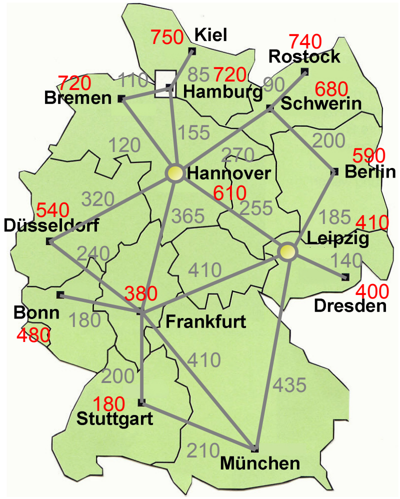
Air distance

Search example



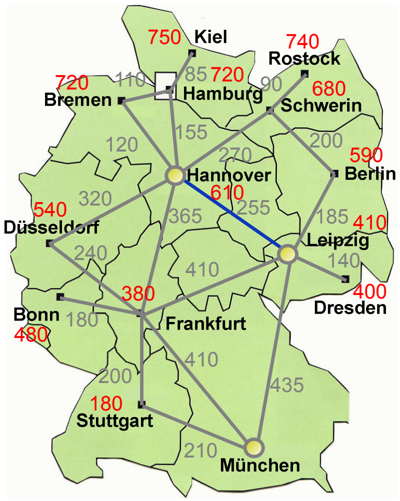


Search example



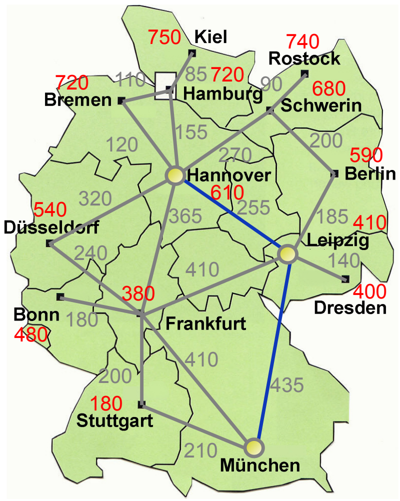
Hannover	0 + 610 = 610
Bremen	120 + 720 = 840
Hamburg	155 + 720 = 875
Kiel	∞ + 750 = ∞
Leipzig	255 + 410 = 665
Schwerin	270 + 680 = 950
Duesseldorf	320 + 540 = 860
Rostock	∞ + 740 = ∞
Frankfurt	365 + 380 = 745
Dresden	∞ + 400 = ∞
Berlin	∞ + 590 = ∞
Bonn	∞ + 480 = ∞
Stuttgart	∞ + 180 = ∞
Muenchen	∞ + 0 = ∞

Search example



Hannover	0 + 610 = 610
Bremen	120 + 720 = 840
Hamburg	155 + 720 = 875
Kiel	∞ + 750 = ∞
Leipzig	255 + 410 = 665
Schwerin	270 + 680 = 950
Duesseldorf	320 + 540 = 860
Rostock	∞ + 740 = ∞
Frankfurt	365 + 380 = 745
Dresden	395 + 400 = 795
Berlin	440 + 590 = 1030
Bonn	∞ + 480 = ∞
Stuttgart	∞ + 180 = ∞
Muenchen	690 + 0 = 690

Search example



Hannover	0 + 610 = 610
Bremen	120 + 720 = 840
Hamburg	155 + 720 = 875
Kiel	∞ + 750 = ∞
Leipzig	255 + 410 = 665
Schwerin	270 + 680 = 950
Duesseldorf	320 + 540 = 860
Rostock	∞ + 740 = ∞
Frankfurt	365 + 380 = 745
Dresden	395 + 400 = 795
Berlin	440 + 590 = 1030
Bonn	∞ + 480 = ∞
Stuttgart	∞ + 180 = ∞
Muenchen	690 + 0 = 690