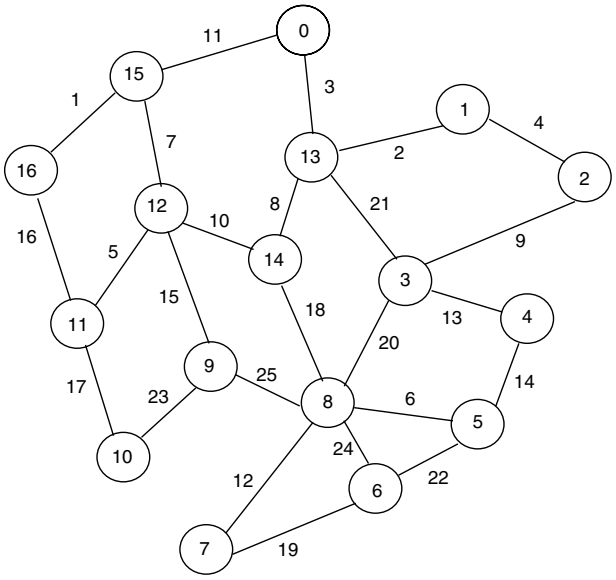
Lab Class S:IV

By December 10th 2014, solutions for the following exercises have to be submitted: 1, 2, 5, 6, 7

Exercise 1 : Search in a graph

Given is the following undirected state space graph with start node 0 and no solution nodes. In which order are the nodes expanded during search? You can assume that some mechanism assures that no node is inserted into OPEN a second time (even if the node is no longer in OPEN or CLOSED). In case of ties during insertion in the lists, the node with smallest number takes precedence. Note that the edge weights are ignored by depth-first search and breadth-first search.



(a) Use depth-first search. Also list the search depth (LIFO principle) of each node.

(b) Use breadth-first search. Also list the depth of each node.

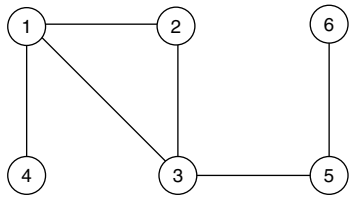
(c) Use uniform cost search. Also list the (minimum) cost of each node.

Exercise 2 : Uniform Cost Search

Given a graph G whose edge weights all have the value 1, do breadth-first search and uniform cost search behave the same way on G? Explain your answer.

Exercise 3 : Search trees

Let there be the following undirected graph G:



Draw the resulting search tree (see the Example: 4-Queens problem) for depth-first search for the following conditions: 1 is the start node; generated nodes are added to the LIFO-stack highest number first; only nodes up to (including) a LIFO-principle depth of 3 are generated; and on node expansion...

(a) all successors are generated.

(b) all successors except those which are on the current CLOSE-list are generated.

(c) all successors except those which are on the current CLOSED-list or OPEN-list are generated.

Exercise 4 : Termination on infinite graphs

Let G be a graph with an infinite number of nodes, where each node is incident to a finite number of edges. Assume that G describes the structure of your search space, and that a goal node can be reached within a finite depth from the start node. Compare the behavior of the algorithms Backtracking (with and without depth bound), Breadth-First Search, Depth-First Search (with and without depth bound) and Best-First Search on this search space: Which of them are guaranteed to find a solution if one exists? Explain your answers.

Exercise 5 : Special cost functions

Given the cost function "maximum absolute difference of the weights of two edges (e_1, e_2) in $P(n)$," where $P(n)$ is the path to n. Formally,

$$f(n) = \mathsf{max}_{e_1,e_2 \in P(n)} |W(e_1) - W(e_2)|$$

where $W(e)$ gives the weight of edge e. You want to find the solution (a path from the start to a goal node) with minimal cost.

(a) Is the statement "a path p is optimal $\Leftrightarrow$ every subpath of p is optimal" always true with regards to f?

(b) Consider the graph in exercise 1. Let 6 be the start node and 0 the only solution node. What is the solution with minimal cost?

(c) Is every solution base of your solution from (b) optimal?

Exercise 6

Explain the concept of a recursive cost function.

Exercise 7 : State space search

You need to send out all the objects listed in the following table in postal packages.

Object	A	B	C	D	E	F	G	H	I	J
Weight (kg)	1	4	6	7	7	9	11	13	15	16

One package can weigh at most 20 kilograms. The shipping costs per package are 9,90 EUR. Your goal is to pack all items into as few packages as possible.

(a) You can represent the problem as a state-space graph (OR graph) or a problem-reduction graph (AND-OR graph). Which problem representation is more appropriate? Explain your answer.

(b) What information is represented by the nodes in the graph; what operations are represented by the edges?

(c) What are the characteristics of solution nodes in your chosen representation, what are dead ends?

(d) How do you determine the costs associated with the edges in your representation?

(e) Given a solution base S, can you give a lower bound on the cost of every single solution candidate described by S?

(f) Write a program that solves package-packing problems. Your program should take as input (1) a list of object weights, and (2) the maximum weight per package; its output should be a list of lists of object weights, corresponding to the cheapest packing for the given inputs. Choose one of the algorithms BF* or GBF* as your control strategy. Explain your choice (e.g., in the comments).

(g) Apply your program to the packing problem given at the beginning of this exercise, and report your results.