

Coursework I

Oliver Kullmann, November 3, 2008

Deadline: November 17, 2008 Submission only at the student's office.

This course works constitutes 10% of the overall marks for CS_232.

The possible marks for individual exercises sum up to 155; results over 100 will be capped. The expected amount of work to spend for an answer for an exercise is roughly proportional to the marks which can be gained. Obviously, you do not need to do all exercises, but you should at least read through them and understand them, so that all exercises seem plausible to you.

When handing in your course work, please state clearly on top of your first sheet the module code, your name and student number, "Coursework I" and the date. For each single exercise, state the exercise number together with the heading for the exercise. Please answer clearly, and in complete sentences. Write legibly!

When using external sources, then you must make references (even for the lecture script)! The complete coursework must be written in its entirety by you. If you are using material from the scripts of previous years, then you must also thoroughly rephrase it in your own words.

If you are asked to develop an *efficient* algorithm, then you must accompany your algorithm with some analysis showing that it actually is efficient.

Since we cannot return your coursework to you, please keep a *copy for yourself*.

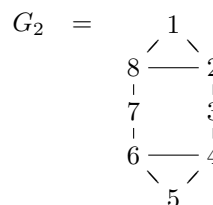
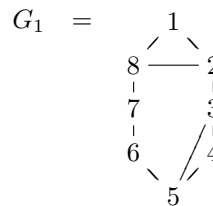
1 Graph isomorphisms (40 marks)

1. Explain in your own words what it means for two graphs to be isomorphic, and what additionally has to be observed for general graphs. [5 marks]

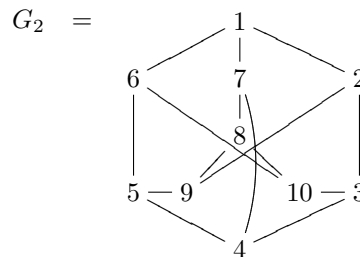
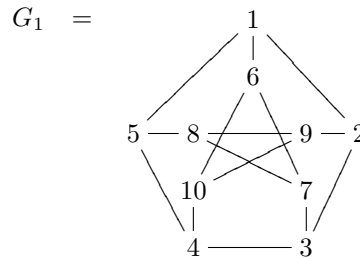
2. A powerful general technique for hard problems is *backtracking*. Explain in a paragraph the general idea (of course, cite your sources!). Describe how to decide whether two graphs are isomorphic via backtracking. State the time complexity of your algorithm using the Big Oh notation. [15 marks]

3. For each of the following pairs of graphs, decide whether G_1 is isomorphic to G_2 or not; in case they are isomorphic show how to rename the vertices of one of them to get the other graph, while in case they are not isomorphic give a criterion that separates them, and argue, that this criterion is invariant under isomorphisms (that is, can be used to show that two graphs are non-isomorphic).

(a)



(b)



[10 marks]

4. Draw all non-isomorphic connected graphs for $n = 0, 1, 2, 3$ vertices. Draw all non-isomorphic trees with $n = 4$ vertices. [10 marks]

2 graph_traversal validated (35 marks)

Give arguments as good as possible for the following central properties of `graph_traversal`, where we assume that the input G is connected (argue directly with the pseudo-code given for `graph_traversal`):

- There are exactly $|V(G)| - 1$ tree edges and $|E(G)| - |V(G)| + 1$ back edges. [10 marks]
- When we consider the situation exactly before the new vertex is marked as visited, then the buffer contains exactly two types of edges:
 - edges connecting vertices already discovered with vertices not yet discovered;
 - edges connecting two already discovered vertices.

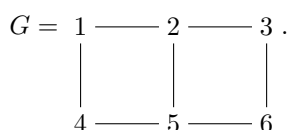
Regarding type (i), show that the buffer must contain all such “frontier edges.” And regarding type (ii), show that all these edges have been identified already as back-edges and will be dropped from the buffer while searching for a new tree edge.

[20 marks]

- Loops are never entered into the buffer but are immediately recognised as back-edges. [5 marks]

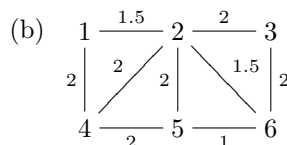
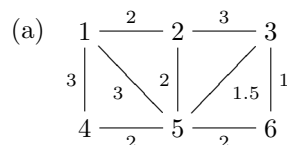
3 Spanning trees (50 marks)

- Compute all spanning trees for



(Justify that you got really all spanning trees.) [10 marks]

- Explain Prim’s algorithm for computing MST’s in your own words. [5 marks]
- Compute MST’s for the following weighted graphs, showing the steps of the computations:



[10 marks]

- Explain Dijkstra’s algorithm for computing SPT’s in your own words. [5 marks]
- Compute SPT’s for the graph of 3a and all six choices for the root, showing the steps of the computations. Derive the distance matrix (for the whole graph, containing all pairwise distances). [10 marks]
- Assume edge e is part of *every* spanning tree for graph G — can you characterise such edges so that that we can identify them efficiently? [10 marks]

4 Depth-first search (30 marks)

- Describe a simplified algorithm for a “depth-first only” form of `graph_traversal`, not using a general buffer, but only a stack, and this only implicitly. Use a simple recursive approach (“to explore a vertex, visit recursively its neighbours”), which offers the same event points for the visitor as `graph_traversal` itself. [20 marks]
- Discuss the differences in traversing graphs between this recursive approach and `graph_traversal`. [10 marks]