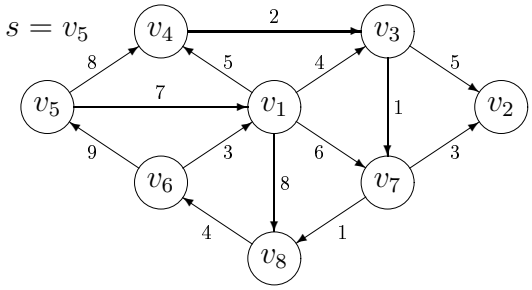
TND004 Data Structures
Lab 5 Part A
Shortest Paths

Overview

The main goal of the first part of this lab is to implement the algorithms presented during lecture 11 (or possibly 12) for the following graph problems:

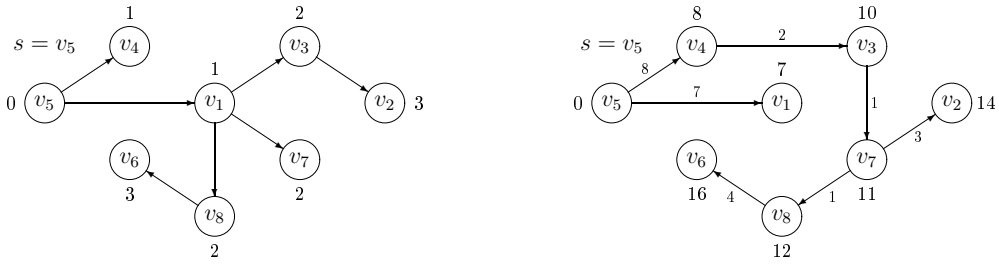
> *Unweighted Single-Source Shortest Paths* (UWSSSP):
> Given a (unweighted) directed graph $G = (V, E)$ and a start vertex $s \in V$, find the shortest unweighted path from s to every other vertex in V.
>
> *Positive Weighted Single-Source Shortest Paths* (PWSSSP):
> Given a weighted directed graph $G = (V, E)$ and a start vertex $s \in V$, find the shortest weighted path from s to every other vertex in V.

The input for these problems is exemplified in figur 1.



Figur 1: A (weighted) directed graph G and a start vertex s.

The output, a shortest path-tree, is exemplified in figur 2.
In particular, the output can be represented by two arrays.



dist	1	3	2	1	0	3	2	2
v	1	2	3	4	5	6	7	8
path	5	3	1	5	0	8	1	1

dist	7	14	10	8	0	16	11	12
v	1	2	3	4	5	6	7	8
path	5	7	4	5	0	8	3	7

Figur 2: Unweighted (*left*) and weighted (*right*) shortest path-tree for s.

The following files can be copied from the course directory `S:\TN\D\004\lab\lab5a`.

- `list.*` : classes for adjacency lists.
- `digraph.*` : class for directed graphs.
- `queue.h` : class for generic queues.
- `main.cpp` : meny-driven test program.
- `lab5a.cbp` : project file for CodeBlocks.
- `digraph1.txt` : data for the digraph in figure 1.
- `digraph2.txt` : data for the digraph in figure 9.8 from the course book.

The testprogram should be self-explanatory (if not, simply ask the lab assistent).

Exercise

Copy the files to your computer. Then implement the following member functions from class Digraph (that is, implement the algorithms presented during the lecture):

- `void Digraph::uwsssp(int s)`
  construct the unweighted shortest path-tree for the current digraph and a given start vertex `s`.
- `void Digraph::pwsssp(int s)`
  construct the weighted shortest path-tree for the current digraph and a given start vertex `s`.

Note that the arrays for representing the shortest path-tree are declared as data members of digraph, and that these arrays are allocated by the digraph constructor.

Finally, implement the following member function from class Digraph.

- `void Digraph::printPath(int t) const`
  print out the shortest path from the start vertex `s` to a given target vertex `t` and the corresponding path length (hint: recursion.)

Presenting your solution

Demonstrate your program to the lab assistent. No print-outs are required.