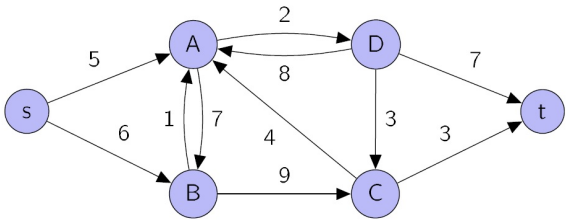
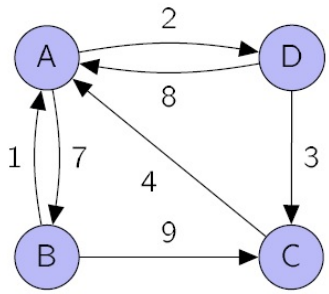
Werkcollege 11

This series of problems tests if you still remember some important concepts and algorithms presented in the previous lectures. Try to see how many of the problems you can solve without consulting the textbook or the slides. During the exam emphasis will be more on design of new algorithms, so this Werkcollege is not representative for the exam. However, in order to design new algorithms knowledge of existing algorithms is often crucial.

11.1. Give a maximal flow and a minimal cut for the following flow network:



11.2. Run the Floyd-Warshall algorithm on the following graph



For each iteration, fill in the intermediate weight values the following table

①	A	B	C	D
A				
B				
C				
D				

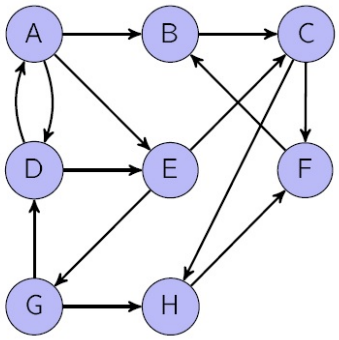
②	A	B	C	D
A				
B				
C				
D				

③	A	B	C	D
A				
B				
C				
D				

④	A	B	C	D
A				
B				
C				
D				

⑤	A	B	C	D
A				
B				
C				
D				

11.3. Draw the component graph of the directed graph G below:



For each vertex in the component graph, indicate which strongly connected component of G corresponds to it.

11.4. Sort the following array using the Mergesort algorithm:

4	3	6	5	2	2	1	6

Give the complete array after all sequences of length 2 have been sorted, after all sequences of length 4 have been sorted, and then the final result:

11.5. Sort for the functions specified below their $\mathcal{O}$-classes $\mathcal{O}(a(n))$, $\mathcal{O}(b(n))$, $\mathcal{O}(c(n))$, $\mathcal{O}(d(n))$ and $\mathcal{O}(e(n))$. Use the relations $\subset$ and $=$ to indicate the relationships between these classes. The next example illustrates the notation (but functions f_1 to f_5 have nothing to do with the functions a to e below):

$$\mathcal{O}(f_4(n)) \subset \mathcal{O}(f_3(n)) = \mathcal{O}(f_5(n)) \subset \mathcal{O}(f_1(n)) = \mathcal{O}(f_2(n)).$$

You do not have to prove the equalities or inclusions.

$a(n) = n^2 \cdot \log_2 n + 42 \quad b(n) = 2^n + n^4 \quad c(n) = 2^{2 \cdot n} \quad d(n) = 2^{n+3} \quad e(n) = \sqrt{n^5}$

11.6. The Levenshtein distance is a metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. Given two strings $a = a_1 a_2 \cdots a_n$ and $b = b_1 b_2 \cdots b_m$, the following recursion equations specify the Levenshtein distance between the subsequences $a_1 a_2 \cdots a_i$ and $b_1 b_2 \cdots b_j$:

$$D(i,j) \;=\; \begin{cases} j & \text{if } i = 0 \\ i & \text{if } j = 0 \\ D(i-1, j-1) & \text{if } a_i = b_j \\ \min(D(i-1, j-1), D(i-1, j), D(i, j-1)) + 1 & \text{otherwise} \end{cases}$$

Use these recursion equations and the principles of dynamic programming to fill the table below and to compute the Levenshtein distance between the strings AACHEN and ATHEN.

		A	A	C	H	E	N
A							
T							
H							
E							
N							