Write the solution to each problem on a single page.

The discussion of questions and solutions before the due date is not discouraged, but you must formulate your own solution.

The deadline for handing in solutions is 28 October before the lecture.

Problem 1. (15 points). In class we discussed how to implement INSERT and DELETEMIN operations in a binary heap. Describe how the following operations can be implemented:

(a) `DECREASEKEY(i, newRank)`. // there holds `newRank ≤ A[i]`.
(b) `INCREASEKEY(i, newRank)`. // there holds `newRank ≥ A[i]`.
(c) `DELETE(i)`. // `DELETE(1)` would be equivalent to `DELETEMIN()`.

Here \( i \) is the index of the item in the array (so \( 1 \leq i \leq n \)). You can use the subroutines presented in class, such as `SIFTUP` and `SIFTDOWN`.

PS: “Key” is another name for a “rank” (or “priority”). “DecreaseKey” should probably be better called “DecreaseRank”, but “DecreaseKey” is a more accepted name.

Problem 2. (25 points). Assume that we are given a directed acyclic graph \( G = (V, E) \) stored using the adjacency list representation. A path from vertex \( i \) to vertex \( j \) is a sequence of nodes \( i_0, i_1, \ldots, i_\ell \) with \( i_0 = i, i_\ell = j \) such that all edges \( (i_0, i_1), (i_1, i_2), \ldots, (i_{\ell-1}, i_\ell) \) belong to \( E \). The number of edges \( \ell \) is the length of this path.

(a) Give an algorithm for computing the length of the longest path in \( G \).
(b) Give an algorithm for computing the number of distinct paths between two specified vertices \( i \) and \( j \).

Analyse the complexity of your algorithms. (More efficient solutions will receive more points.)

Hint: Both problems can be solved via the dynamic programming idea.

![Figure 1: Example of a directed acyclic graph. (a) The length of the longest path is 4. There are two paths of such length: \( (c, e, d, a, b) \) and \( (f, e, d, a, b) \). (b) There 5 paths from \( f \) to \( b \): \( (f, e, b), (f, e, d, b), (f, d, b), (f, e, d, a, b), (f, d, a, b) \). Note, for some pairs the number of paths may be zero, e.g. there are no paths from \( d \) to \( c \).]