Asymptotic complexity

**1**. Firstly, verify that statement (*n* ∈ **N**) ( *n* > 4 ⇒ *n*2 − 2*n* > 0.5*n*2) is true. Because it is true then the graph of the function *f*(*n*) = 0.5*n*2 lies below the graph of the function *g*(*n*) = *n*2 − 2*n* whenever  *n* > 4. Moreover, the difference g(*n*) − f(*n*) grows to infinity with growing *n*.

Use the definition of the set O(*g*(*n*)) to prove that even in these circumstances the relation *g*(*n*) ∈ O(*f*(*n*)) holds.

**2.** Symbol lg denotes logarithm to base 2. Sort the following functions according to their order of growth. Explain the relation of each two neighbouring functions in this ordering.

      

**3.** Algorithm P processes an array of length N and the time it takes to process element at position *k* is *c*⋅*k* seconds. The factor *c* is a constant. Determine the asymptotic complexity of processing the array.

**4.** Algorithm P processes a two-dimensional array of size N×N and the time it takes to process element at position (*k*,*m*) is equal to A: *c*⋅(*k*+*m*), B: *c*⋅*k*⋅*m* milliseconds. The factor *c* is a constant. Determine the asymptotic complexity of processing of the array in case A and in case B.

Graph representation

**5.** Describe particular representations of graphs in the computer memory you know. For each possible pair of representations (R1, R2) determine the asymptotic complexity of transforming the graph from representation R1 to R2.

**6.** List L contains all edges of the graph, each edge (= element of the list) is represented as pair <node, node>. The graph is undirected, not connected, has N nodes and contains a component K which has more than N/2 nodes. The task is to create a new list of all edges in the component K. Describe an effictive algorithm which solves the task and determine its asymptotic complexity. The order of edges in both lists is not specified.

**7.** Second power of a graph G is a graph G2, which has the same nodes as graph G and its set of edges is defined as follows: G2 contains an edge {u, v} if and only if there is another node w such that both edges {u, w} and {w, v}

are in G. Describe an algorithm which transforms G into G2 and

(A) uses linked link representation of the graph,

(B) uses adjacency matrix representation of the graph.

Which representation (A) or (B) yields asysmptotically faster solution?

**8.** There are two algorithms A1 and A2 both process simple undirected graph with *n* nodes and *m* edges. Both algorithms solve the same problem and produce identical output for any input graph. Asymptotic complexity of

A1 is Θ(*n* *m* log(*n*)), asymptotic complexity of A2 je Θ(*n*2 log(*m*)). Decide which of the algorithms is more effective and for what types of graph should each of them be used.

BFS DFS

**9.** An undirected graph G = (V, E) with N nodes labeled from 0 to N −1 is represented by a list of its edges, each edge is a pair {node\_label, node\_label}. There is no other information about the graph available. You have to perform algorithm BFS using only this representation, one queue of nodes and at most constant ammount of additional memory (you cannot build another representation of the graph). What will be the complexity of BFS in this case? Solve this problem also for algorithm DFS (with only one stack available).

**10.** A given graph has *n* nodes and Θ(*n*2) edges. The asymptotic complexity of DFS processing of this graph is usually claimed to be Θ(*n*2). Such claim relies on assumption that there is an instant access (in constant time) to each node and each edge. In this problem we have to determine how the DFS slows down when the time of access to any node will take time proportional to *n1/2* and also access to any edge will take time proportional to *n1/2*. Solve the same problem for BFS algorithm.