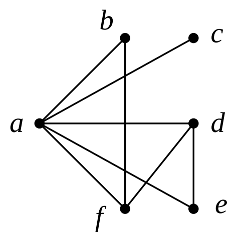
CS69003: Computing Systems Lab – I
Autumn 2007

Assignment 3

Implementation of an algorithm for undirected graphs, based on heap operations

Due: August 17, 2007

Let $G = (V, E)$ be a simple undirected graph with n vertices. The *degree* $d(v)$ of a vertex $v \in V$ is the number of nodes adjacent to v in G. The *degree sequence* of G is the sequence of degrees of its vertices and is unique up to permutations of the vertices. For example, the degree sequence of the following graph is $5, 2, 1, 3, 2, 3$.



Vertex	Degree
a	5
b	2
c	1
d	3
e	2
f	3

Given a representation of a graph, one can easily compute the degree sequence of the graph. In this assignment, you solve the converse problem. A finite sequence of non-negative integers is called *graphic* or *realizable* if there exists a simple undirected graph whose degree sequence equals the given sequence. For example, the sequence $5, 2, 1, 3, 2, 3$ is graphic and so also is its permutation $5, 3, 3, 2, 2, 1$. On the other hand, the sequence $6, 5, 4, 3, 2, 1$ is not graphic, since in a simple graph with six vertices, no vertex can have degree ≥ 6. Also the sequence $5, 4, 3, 2, 2, 1$ cannot be graphic, since the sum of the degrees in the sequence is odd.

Write a program that inputs a sequence from a file and computes a graph whose degree sequence equals the sequence read. If no such graph exists (i.e., if the input sequence is not graphic), your program should terminate with an appropriate error message.

Part I

Implement the **max-heap** data structure. More precisely, provide a suitable type-definition for the data structure heap and implement the insertion and deletion routines on data of type heap. Use the following prototypes. The insert and delete functions must return the modified heaps. A function that returns the root (i.e., the maximum node) of the heap without deleting that node should also be implemented.

```
heap insert ( heap, node );                 /* Inset a new node in a heap */
heap deleteMax ( heap );                /* Delete the maximum from a heap */
node getMax ( heap );    /* Obtain the maximum of a heap without deletion */
```

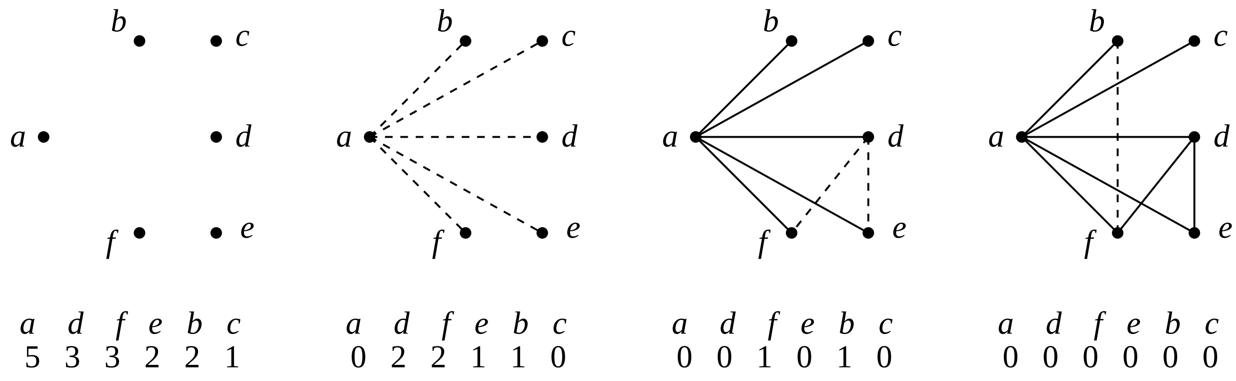
Note that `heap` is the custom-designed data type as you have declared earlier. On the other hand, `node` is a data type containing a key value based on which the heap structure is organized. Since you are going to use these heap functions for solving the graphic sequence problem, each `node` may contain additional data items which are not relevant with respect to the heap property. For example, an item in the heap would consist of the id of a graph vertex and a degree information about that vertex, each of which can be represented as a non-negative integer.

Part II

Havel (1955) and Hakimi (1962) proved a necessary and sufficient condition for a sequence to be graphic. This condition immediately leads to a recursive (or iterative) algorithm for computing a graph (if existent) belonging to a given sequence. The condition goes like this.

A sequence $d_1, d_2, \ldots, d_n$ of n non-negative integers with $d_1 \geq d_2 \geq \cdots \geq d_n$ is graphic if and only if the sequence $d_2 - 1, d_3 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2}, \ldots, d_n$ is graphic.

This result indicates that one creates all the links for a node with the largest degree and then recursively computes the rest of the graph. This procedure is illustrated in the following figure. We start with the sequence $5, 2, 1, 3, 2, 3$ corresponding to nodes a, b, c, d, e, f. Edges added in each step are shown by dashed lines.



Read a text file storing the number n of nodes followed by a sequence of n non-negative integers. Note that the input sequence is not necessarily supplied in the decreasing order. Create a heap based on the degree values and perform a series of insert and delete operations on the heap so as to obtain a graph, if existent, corresponding to the input sequence. While printing the output graph, name the vertices as $a, b, c, d, \ldots$ in the order they are read from the input file.

Example

Consider the following input file.

```
5
3
1
2
4
2
```

The file says that there are five nodes. Node a has degree 3, node b has degree 1, and so on. This sequence happens to be graphic. A sample execution of your program on this file would look like the following.

```
Heap size = 5
[d,  4] [a,  3] [c,  2] [b,  1] [e,  2]

Output graph
  a b c d e
a 0 0 1 1 1
b 0 0 0 1 0
c 1 0 0 1 0
d 1 1 1 0 1
e 1 0 0 1 0
```