CISC 7200X – Analysis of Algorithms

Realizing Degree Sequences
Programming Project

Nov 10, 2020 – Dec 15, 2020

Objective: Let $D = \langle d_1 \geq d_2 \geq \cdots \geq d_n \rangle$ be a non-increasing sequence of n non-negative integers in which $d_1 \leq n - 1$. Code a program that decides if D is graphic. If D is graphic, the program outputs one of the graphs that realizes D and if D is not graphic, the program announces that this is the case.

Method: Implement the algorithm taught in class (the Havel Hakimi (HH) algorithm). Iteratively, starting with the input sequence, in each round, select the vertex whose current degree d is the highest, connect it to d vertices with the current highest degrees (preferring the vertices that appeared first in the original sequence), and update the sequence. Terminate with a statement that D is not graphic if you cannot continue to the next round and terminate with a statement that D is graphic if all the degrees in the new sequence are 0.

Programming language: You may use any programming language.

Input: A valid input is represented by a sequence of $n + 1$ integers. The first integer, $n \geq 1$, is the length of D and it must be followed by a sequence of n integers in the range $[0..n - 1]$ that appear in a non-increasing order. Two ways to enter the input should be implemented:

- An interface that allows iteratively typing valid inputs one after another. The next valid input may be typed after the output of the previous input is displayed.
- A data file containing k lines in which each line is a valid input.

Remark: For each input line that does not represent a valid input, the program should output: “invalid input”.

Output:

- If D is not graphic, print the message “The degree sequence D is not graphic”. Include the sequence D with the message.
- If D is graphic, draw one of the graphs that realizes D. Inside (or near) each vertex print its degree. Show the sequence D below the illustration of the graph.

Extras: Implement as many variants of the HH algorithm from the list below. In each round, call the selected vertex the *pivot* vertex.

- MAX-HH: The pivot vertex is one of the vertices with the highest degree.
- MIN-HH: The pivot vertex is one of the vertices with the lowest degree.
- UR-HH: Every vertex is selected as the pivot vertex with a uniform probability.
- PR-HH: Vertex v_i whose current degree is d_i is selected as the pivot vertex with probability

$$p_i = \frac{d_i}{\sum_{j=1}^{j=n} d_j}$$

- ParR-HH: Given a parameter x, vertex v_i whose current degree d_i is selected as the pivot vertex with probability

$$p_i = \frac{d_i^x}{\sum_{j=1}^{j=n} d_j^x}$$

If you implement more than one variant but not the last variant, in case of D is graphic illustrate one graph for each implemented version. If you implemented the last variant, in case D is graphic illustrate the realizations from the following 5 variants in this order: MAX-HH, ParR-HH with $x = 1$, ParR-HH with $x = 0$, ParR-HH with $x = -1$, MIN-HH.

Remark: Note that the first four variants are special cases of the fifth variant. For MAX-HH set $x = \infty$, for MIN-HH set $x = -\infty$, for UR-HH set $x = 0$, and for PR-HH set $x = 1$.

Efficiency: For more credit, you need to efficiently implement the HH algorithm (any version of it). The total time complexity should be $\Theta(m)$ where $m = \sum_{j=1}^{j=n} d_j$.

Testing the code: Once you are ready, send an email to amotz@sci.brooklyn.cuny.edu. This email should contain the following:

- Instructions for how to test your code with both ways for entering valid inputs.
- The portion of the code that implements the Havel Hakimi algorithm.
- A detailed explanation of the complexity of the implementation of the algorithm.

Next, I will send you a document containing list of tasks that you have to do for the final submission. Finlay, we will have around 30-minute video meeting in which you show me how your project performs and explain to me your implementation.

Grading: You lose nothing by trying to complete the project because your grade will be 100 regardless of how well you do. For those who do well, the percentage by which the project will count in their final grade could be as high as 25% depending on the factors listed below. On the other hand, if you get it entirely wrong, the percentage could be zero. In any case, it will not hurt you to try, it can only help improve your grade.

Grading main factors: Correctness; an efficient implementation of the HH algorithm; number of variants implemented; a friendly interface for the input and clarity of the displayed output; and unexpected initiatives shown by the students.