Complex Systems 535/Physics 508: Homework 5

1. **Complexity of simple network operations:** State the time complexity of the following operations in terms of the number of vertices n and edges m (with brief arguments why):

   (i) Multiplying the adjacency matrix of a dense graph into an arbitrary vector.

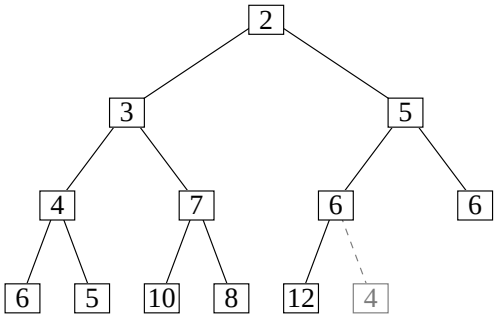
   (ii) Multiplying the adjacency matrix of a sparse graph into an arbitrary vector. (You can assume that you also have the adjacency list of the sparse graph.)

   (iii) Calculating the degree sequences of dense graphs and sparse graphs from their adjacency matrices or lists.

   (iv) Calculating the clustering coefficient of an undirected graph from an adjacency list.

   (v) Calculating the reciprocity of a digraph from an adjacency list.

2. **The binary heap:** In class we studied Dijkstra's algorithm, and mentioned that efficient implementations make use of the binary heap. A heap is a partially ordered binary tree in which numbers are stored at the vertices of the tree in such a way that all numbers are greater than or equal to their "parent" in the tree:



   (Ignore for the moment the extra item at the bottom with the dashed line, which is obviously not greater than its parent.)

   (i) If there are n numbers in the tree, what is the depth of the tree (i.e., the number of levels)?

   (ii) The process of adding an item to the heap involves "sifting" the elements. First, you add the item at the bottom of the heap, as with the "4" at the bottom of the picture. Then you compare it to its parent, and if it is smaller, you exchange parent and child, and you do that repeatedly until the new item finds a level at which it is, correctly, greater than its parent. Draw a picture of the heap above after the item "4" has been sifted up the tree.

   (iii) What is the time complexity in terms of n for adding an item to a binary heap and why?

   (iv) Removing an item is similar. You remove the item in question leaving a hole in the tree. You fill the hole by moving the last item in the tree into it, then you "sift" this item down the tree. That is, you compare it to the smaller of its two children, and if it is greater than that child, you swap the two. You repeat this process until the item finds its correct level. Draw the heap above after the item "3" has been removed from it and the tree sifted.

   (v) What is the time complexity for removing an item from a heap and why?

   (vi) What is the time complexity for finding the smallest item in a heap and why?

   (vii) If we use a binary heap to store the estimated distances of vertices from the source in Dijkstra's algorithm, what will be the overall time complexity of the algorithm in terms of the number of vertices n and the number of edges m in the graph?

3. **Complexity of centrality measures:** Consider the Katz centrality $\mathbf{x} = (\mathbf{I} - \alpha\mathbf{A})^{-1} \cdot \mathbf{y}$. Matrix inverses take $O(n^3)$ time in general, but for a sparse graph the inverse can be calculated more quickly by expanding $(\mathbf{I} - \alpha\mathbf{A})^{-1} = \sum_{r=0}^{\infty} (\alpha\mathbf{A})^r$, and performing the sum up to some finite number of terms $r_{\max}$, i.e., up to *but not including* the term $r = r_{\max}$. The matrix error involved in doing this is $\Delta = \sum_{r_{\max}}^{\infty} (\alpha\mathbf{A})^r$.

   (i) Assuming the network is sparse and symmetric, we can estimate the size of this error by calculating the Frobenius norm divided by n:

$$\frac{\|\Delta\|}{n} = \sqrt{\frac{\sum_{ij} \Delta_{ij}^2}{n^2}} = \sqrt{\frac{\mathrm{Tr}\,\Delta^2}{n^2}},$$

   which is the root-mean-square error on an element of $(\mathbf{I} - \alpha\mathbf{A})^{-1}$. Show that the trace is equal to

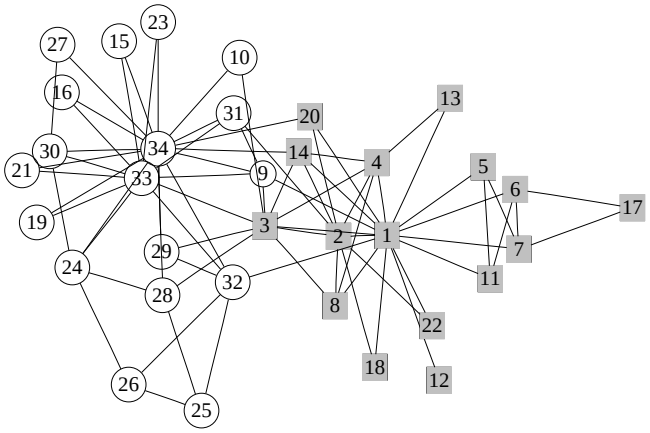
$$\mathrm{Tr}\,\Delta^2 = \sum_{i=1}^{n} \left[\frac{(\alpha\lambda_i)^{r_{\max}}}{1 - \alpha\lambda_i} \right]^2,$$

   where λ_i is the ith eigenvalue of the adjacency matrix.

   (ii) Hence for given n, and assuming that $\alpha\lambda_i \ll 1$ for all i, find an approximate expression for the leading-order scaling of the runtime taken by the algorithm to calculate an answer to any desired degree of precision (measured by $\|\Delta\|/n$) in terms of n, λ_1, and α, where λ_1 is the leading eigenvalue.

   (iii) For k-regular graphs how does the runtime scale? (You should find that this is indeed better than the standard $O(n^3)$ inversion algorithms, for given α.)

4. **Spectral bisection:** Here is a famous graph from the social networks literature:



   It represents friendships between students at a karate club at a US university (W. W. Zachary, An information flow model for conflict and fission in small groups. *Journal of Anthropological Research* **33**, 452–473 (1977)). You can download the adjacency matrix for this network from here: `http://www-personal.umich.edu/~mejn/courses/2004/cscs535/karate.adj`

   (i) Carry out a spectral bisection of this network. Give the algebraic connectivity of the graph and the corresponding eigenvector. Thus determine the bisection of the graph.

   (ii) During the course of Zachary's study of the karate club, a dispute arose between the members and the club eventually split in two. The two factions are shown by the squares and circles in the figure above. How well do these correspond to the results of your bisection?