Complex Systems 535/Physics 508: Homework 4

	species	is preyed on by		species	is preyed on by
0	phytoplankton	6, 7, 10, 11, 12, 21, 22, 33	17	crustacean deposit feeder	18, 24, 25, 26, 28, 31
1	bacteria in suspended poc	6, 7, 8, 10, 11, 12, 21, 22,	18	blue crab	32
2	bacteria in sediment poc	13, 14, 15, 16, 17	19	fish larvae	
3	benthic diatoms	16	20	alewife and blue herring	32
4	free bacteria	5	21	bay anchovy	26, 27, 29, 30, 31, 32
5	heterotrophic microflagel	6	22	menhaden	29, 31, 32
6	ciliates	7, 8, 10, 11, 12	23	shad	
7	zooplankton	8, 9, 19, 20, 21, 22, 23	24	croaker	
8	ctenophores	9	25	hogchoker	
9	sea nettle		26	spot	29
10	other suspension feeders	18	27	white perch	
11	mya arenaria	18, 25	28	catfish	
12	oysters		29	bluefish	
13	other polychaetes	24, 25, 26, 27, 28	30	weakfish	31
14	nereis	18, 24, 25, 26, 27, 28	31	summer flounder	
15	macoma	18, 26	32	striped bass	
16	meiofauna		33	dissolved organic carbon	4

1. **Food webs and trophic levels:** The table above gives the adjacency list for the food web of predator-prey interactions between marine (i.e., salt-water) species in the Chesapeake Bay, a large tidal bay on the Eastern Seaboard of the United States (D. Baird and R. E. Ulanowicz, The seasonal dynamics of the Chesapeake Bay ecosystem. *Ecological Monographs* **59**, 329–364 (1989)). You can also download the same data from here:

   `http://www-personal.umich.edu/~mejn/courses/2004/cscs535/chesapeake.adj`

   which might save you some typing time.

   (i) Calculate the eigenvalues of the adjacency matrix. Hence make a statement about whether this food web is cyclic or acyclic.

   (ii) The standard way of calculating trophic levels is to assign to each species a trophic level equal to the mean of the trophic levels of their prey, plus 1. Derive an expression for the vector $\mathbf{x}$ of trophic levels in terms of the adjacency matrix $\mathbf{A}$. The expression you get should involve the *directed graph Laplacian* $\mathbf{D}-\mathbf{A}$, where $\mathbf{D}$ is the diagonal matrix of in-degrees. This expression does not work for autotrophs—species with no prey. Such species are usually given trophic level 1. Suggest a modification of your method that will correctly assign trophic levels to these species, and hence to all species.

   (iii) Apply your formula to the Chesapeake Bay network and calculate the trophic levels. Which species is at the top of the food chain (i.e., has the highest trophic level)?

2. **Computational complexity of sorting algorithms:** A simple and stupid algorithm for sorting a set of numbers is the "bubblesort." You are given a set of n numbers that are to be sorted in ascending order. You go through them all in turn (except the last one) comparing each to the number after it in the list. If the first number is greater than the second, you swap the two. Otherwise you do nothing. If you repeat this whole process often enough, the numbers will eventually be sorted in order.

   (i) How many passes along the list will you have to make to sort the numbers in the worst case? Hence what is the runtime of the algorithm to leading order in n?

(ii) A much better sorting algorithm is "mergesort." If we have two lists of $\frac{1}{2}n$ numbers each, and each is correctly sorted in ascending order, then we can make a single correctly sorted list of all n numbers by comparing the smallest (i.e., first) items in each list and moving the smaller of the two to a new list. Repeatedly doing this until all items in the two original lists have been moved to the new list will give us a new list that is correctly sorted. How does the number of operations involved in this merge process scale with n?

(iii) Assuming n is a power of 2, we can now start with n lists of 1 item each (which are, by definition, sorted) and merge them repeatedly to make sorted lists of length 2, 4, 8, and so on up to n. How many such sets of merges will be needed? Hence what is the runtime of mergesort to leading order in n?

3. **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) Generating a degree sequence that is sorted in ascending order, for dense graphs and for sparse graphs. (You can assume the number of edges in the dense graph is not $o(n \log n)$.)

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

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

4. **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.)