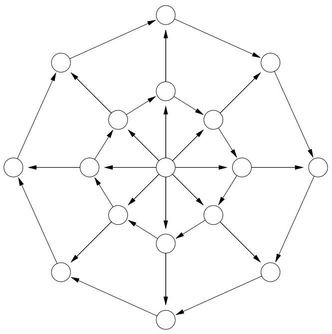
CPS 230 Midterm 2

Fall 2000

2:20pm-3.35pm, Wednesday November 28
Closed book exam

25 points **Problem 1**

A *wheel-graph* is a directed graph of the following form:



More precisely, a wheel-graph consists of a center vertex c with k outgoing "spokes" of s outward oriented edges each. Furthermore, the ith vertex ($i = 2, 3, \ldots, s+1$) of all the spokes are connected to form a directed cycle. All cycles are oriented the same way (refer to the figure, in which $k = 8$ and $s = 2$).

a) What is the number of edges m in a wheel-graph as a function of the number of vertices n?

Assume we are given a wheel-graph with positive integer edge-weights. We want to find the length of the shortest paths from the center c to all other vertices.

b) How long time would Dijkstra's algorithm use to solve the problem (as a function of n)?

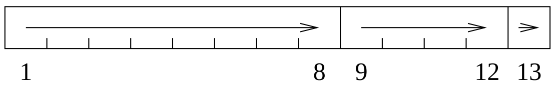
c) Describe a more efficient algorithm for solving the problem. Remember to argue for both running time and correctness.

40 points **Problem 2**

In this problem we consider a data structure for maintaining a set of elements under operations INSERT(x) and SEARCH(x).

The data structure containing n elements consists of $O(\log n)$ *sorted* lists such that there is a list of length 2^k if the kth bit of the binary representation of n is 1. All the lists are stored in order of decreasing length in a single array A of length n.

Example: A collection of $n = 13 = 1101_2$ elements is stored in three sorted lists as follows:



a) Describe how to implement SEARCH(x) in time $O(\log^2 n)$.

b) Assume that one can extend the length of array A in constant time. We implement INSERT(x) by storing x in $A[n+1]$ and reorganizing A such that it again consists of lists corresponding to the 1-bits in the binary notation of $n+1$. Describe how this can be done in $O(n)$ time.

c) Show that the amortized time for an insertion in a sequence of n insertions on an initially empty data structure is $O(\log n)$.

35 points **Problem 3**

A *substring* of a string $x = x_1x_2 \dots x_n$ of n characters is a contiguous set of characters $x_ix_{i+1} \dots x_j$ from x ($i \le 1 \le j \le n$). If a substring of x is also a substring of y we call it a *common substring* of x and y. If no other common substring is longer, we call it the *longest common substring*.

A substring of x that ends at x_n is called a *suffix* of x. If a suffix of x is also a suffix of y we call it a *common suffix* of x and y. If no other common suffix is longer, we call it the *longest common suffix*.

Example: The longest common substring and longest common suffix of ``supercharged'' and ``targeted'' are ``arge'' and ``ed'', respectively.

In the following let $x = x_1x_2 \dots x_n$ and $y = y_1y_2 \dots y_m$ be two strings of length n and m. Let $S[i,j]$ be the *length* of the longest common suffix of $x_1x_2 \dots x_i$ and $y_1y_2 \dots y_j$ ($1 \le i \le n$ and $1 \le j \le m$).

a) Assume that we know $S[i,j]$ and that we can read $S[i,j]$ in $O(1)$ time for all $1 \le i \le n$ and $1 \le j \le m$. Describe how to find the *longest common substring* of x and y in $O(nm)$ time. Remember to argue for correctness and running time.

b) Give a recurrence for $S[i,j]$ (Hint: Consider the connection between $S[i,j]$ and $S[i-1,j-1]$).

c) Describe an $O(nm)$ algorithm for computing $S[i,j]$ for all $1 \le i \le n$ and $1 \le j \le m$. Remember to argue for the correctness and running time.