Problem Set 2
Assigned: 2/3/2019
Due by class time on 2/12/2019
Total points 56

You may express the algorithm in the form of pseudo-code or a description of its behavior. In either case, please provide an intuitive explanation of why the algorithm is correct.

We begin with some background material for Question 1:

Real-values matrices:

Consider an n-by-n matrix *A* wherein each element of *A* is a real number. Then the standard matrix product $B = AA = A^2$ is given by the following:

$$b_{ij} = \sum_{k=1}^{n} a_{ik}.a_{kj}$$

where a_{ik} denotes matrix element *A[i,k]*. As an example, when *n*=4, we have

$$b_{24} = a_{21}.a_{14} + a_{22}.a_{24} + a_{23}.a_{34} + a_{24}.a_{44}$$

Note that, for standard matrix multiplication, the . operation above denotes multiplication of real numbers, and + operation denotes addition of real numbers. In other words, for real-valued x and y, we assume above,

x.y = product of x and y
x+y = sum of x and y

Path in a graph: Path (a,b,c,d,e) from vertex a to vertex e is said to exist in graph G=(V,E) provided that edges (a,b), (b,c), (c,d), (d,e) all exist in E. Note that, if the graph is directed, the direction of the edges matters in determining a path.

Length of a path is the number of edges in the path. In path (a,b,c,d,e), there are 4 edges, and thus its length is 4.

Using matrix "multiplication" for adjacency matrices:

Now suppose that matrix A is the adjacency matrix for a graph G=(V,E). Recall that each element of A is either 1 or 0, which can also be interpreted as TRUE or FALSE (signifying the presence of the corresponding edge in E). In particular, a_{ij} is 1 if and only if there exists edge (i,j) in E; it is 0 otherwise. To put it differently, a_{ij} is 1 if and only if there exists a **path of length 1** from i to j; it is 0 otherwise.

Again consider the expression for b_{24} :

$$b_{24} = a_{21}.a_{14} + a_{22}.a_{24} + a_{23}.a_{34} + a_{24}.a_{44}$$

As discussed in the class on 1/31/2019, if we define the operation . above as a logical-AND and + as a logical-OR, then the following is true:

> b_{24} is 1 (or TRUE) if and only if
> there is a path of length exactly 2 from vertex 2 to vertex 4;
> else it is 0.

With the above definition of . and + operations for binary x and y,

> x.y = x AND y
> x+y = x OR y

The matrix product $B = AA = A^2$ is defined as before, with the exception that we have changed the meaning of . and + operations to AND and OR as stated above. In general,

$$b_{ij} = \sum_{k=1}^{n} a_{ik}.a_{kj}$$

b_{ij} here is 1 if and only if there is a path of length exactly 2 from vertex i to vertex j; it is 0 otherwise. In the above, we treat elements of the adjacency matrix as binary values.

Question 1 (12 points)

In this question, elements of matrix A are treated as real numbers. Consider graph G=(V,E).

(a) Let A denote the adjacency matrix of graph G. The elements of A are 1 and 0, and treated as real numbers in this part. In part (a) , suppose that the . operation above is treated as an addition, and + operation above is defined as the *minimum*. That is, for real-valued x and y, let

> x.y = sum of x and y
> x + y = minimum of x and y

Consider matrix product $B = AA = A^2$. As before, we have

$$b_{ij} = \sum_{k=1}^{n} a_{ik}.a_{kj}$$

where . and + are interpreted as sum and minimum operations, respectively.

What do the elements of B signify? For instance, if b_{ij}=2, what does it mean? Provide a justification for your answer.

(b) Now suppose that each edge in E has a cost. For instance, if an edge (i,j) represents a highway from i to j in a road network, then the cost of edge (i,j) may be the toll to be paid on that part of the highway.

Suppose that we re-define matrix A as follows for graph G=(V,E).

a_{ij} = cost of edge (i,j), if edge (i,j) is in E
a_{ij} = ∞, if edge (i,j) is not in E

Similar to part (a), in part (b) also, the . and + operations are defined as sum and minimum.

What do the elements of $B = AA = A^2$ signify with the above definition of A? For instance, if b_{ij}=2, what does it mean? Provide a justification for your answer.

Question 2 (10 points)

Suppose that a graph G(V,E) is represented using an adjacency list. In particular, Adj[i] is a pointer to the head of a linked list, with the list elements storing vertices to which vertex i has an edge, in an increasing order.

Write an algorithm that determines whether there exists a path of length exactly 2 from vertex i to vertex j. Your algorithm should take the graph representation, |V|, i, and j are as arguments, and return 1 if a path of length 2 exists from vertex I to vertex j, and return 0 otherwise.

Determine the Big-Θ bound for the worst-case execution time of your algorithm as a function of |V| and |E|. Explain your answer intuitively (you need not show a formal proof of the bound).

Question 3 (10 points)

(a) Present a *randomized* divide-and-conquer algorithm that finds the sum of all the elements in array C[1,…,n]. Your algorithm should take as arguments array C, and two integers p, r, such that $1 \leq p \leq r \leq$ length(C) = n, and return the sum of the values in C[p,...,r].

(b) Write a recurrence for the expected execution time of your solution in part (a), and provide a brief explanation.

Question 4. (12 points)

Mergesort algorithm recursively performs sort on two halves of the input array, and then merges the sorted halves. Obtain a modification of Mergesort, which recursively sorts three subarrays of length one-third of the original array, and then merges the three sorted subarrays.

(a) Show that it is possible to merge the three sorted subarrays in Θ(n) time, where n is the size of the input array.
(b) Write a recurrence for the step complexity of your algorithm.
(c) Solve the above recurrence using the master method.

Question 5 (12 points)

The goal here is to design an algorithm to sort an array *A* of strings (corresponding to English words) as per their order in a dictionary. As an example, for strings *a, an, and, by, byte, cat*, the order is given by *a < an < and < by < byte < cat.*

A[i] is a pointer to the i-th string in array *A*. Assume that a procedure *length(str)* is available that returns length of string with pointer *str*. Also, assume that there is a $\Theta(1)$ time algorithm *find-char(string)* available that returns the i-th character from the beginning of the *string*, when i<= length of the *string*, and returns a special character NULL if i>length of the *string*.

Assume that the array is of size *k*.

Assume that the sum of the length of all the strings in the array is *n*. Assume that each string contains at least one character.

If an algorithm with $\Theta(n)$ time complexity exists for the above problem, present such an algorithm and argue that its step complexity is $\Theta(n)$. Otherwise, explain why such an algorithm does not exist, and present an alternate algorithm to solve the problem.

SUGGESTED EXERCISES

You need not turn in solutions for suggested exercises. No credit is assigned for suggested exercises.

(SUGGESTEDT EXERCISE) In Question 1, consider the matrix product A^n for the definition of . and + in each part above: what do the elements of A^n signify?

(SUGGESTED EXERCISE) Solve Question 2 again using adjacency matrix as the graph representation.

(SUGGESTED EXERCISE) Can we use the adjacency matrix to find the shortest path between each pair of vertices? Try to come up with an algorithm for this.

(SUGGESTED EXERCISE) Solve the recurrence in Question 3.
