CSE421: Design and Analysis of Algorithms

Homework 2

Anup Rao Due:

Each problem is worth 10 points:

1. Prove that in every undirected graph, there must be two vertices that have the same degree. HINT: Argue that If the degrees are all distinct then the vertices must have degrees $0, 1, 2, \ldots, n-1$, and this cannot happen.

   Solution. The maximum possible degree of a vertex in the graph is $n-1$ which is achieved when a vertex is connected to all the other vertices of the graph. So there are n possible degrees that a vertex can have: $0, 1, \ldots, n-1$. If every node has distinct degree, there must be exactly one node with each of the possible degrees. However, a vertex with degree 0 cannot be connected to any other nodes, and a vertex with degree $n-1$ must be connected to all other nodes, so you cannot simultaneously have a vertex with degree 0 and a vertex with degree $n-1$ in the graph. So, we cannot simultaneously have a vertex of degree 0 and a vertex of degree $n-1$.

2. A walk of length k in a graph is a sequence of vertices $v_0, v_1, \ldots, v_k$ such that v_i is a neighbor of v_{i+1} for $i = 0, 1, 2, \ldots, k-1$. Suppose the product of two $n \times n$ matrices can be computed in time $O(n^\omega)$ for a constant $\omega \geq 2$. Give an algorithm that counts the number of walks of length k in a graph with n vertices in time $O(n^\omega \log k)$. HINT: If A is the adjacency matrix, prove that the $(i,j)'th$ entry of A^k is exactly the number of walks of length k that start at i and end at j. Repeatedly square the adjacency matrix to compute A^k.

   Solution. The algorithm is given below.

   (a) **Proof of correctness:** The key claim is that the i, j-th entry of A^k counts the number of k-length walks from i to j, for all i, j. Thus summing over the matrix gives us the total number of k-length walks.

   The proof of the claim is via induction. The base case, $k = 1$, follows from the fact that a one length walk between any two vertices corresponds to a (directed) edge. Since the ij-th entry of A computes the number of edges from i to j, it also computes the number of one length walks between from i to j. Now we proceed to the inductive case, assuming the claim holds for some $k \geq 1$. To this end, we will prove that

   $$\#(k+1 \text{ length walks from } i \text{ to } j) = \sum_{j' : A_{j'j} = 1} \#(k \text{ length walks from } i \text{ to } j'). \tag{1}$$

   Assuming Equation (1), we are done because by the inductive hypothesis, the number of k length walks from i to j' is given by $A^k_{ij'}$. Therefore, the number of $k+1$ length walks from i to j would then be $\sum_{j' : A_{j'j} = 1} A^k_{ij'} = \sum_{j'} A^k_{ij'} A_{j'j} = A^{k+1}_{ij}$.

```
Input: Adjacency matrix A and a natural number k
Result: A^k
Result ← I;
while k ≠ 0 do
    if k mod 2 = 0 then
        k ← k/2;
        A = A ∗ A
    end
    else
        k ← k − 1;
        Result ← A ∗ Result;
    end
end
count ← 0 for i from 1 to n do
    for j from 1 to n do
        count ← count + A_{i,j}
    end
end
return count;
```

We prove Equation (1) in two parts. First, note that we can form a walk from i to j of length $k+1$ by starting at i and walking to some neighbor of j, say j', and then walking on the edge from j' to j. Every choice of the length k walk from i to j' followed by the edge from j' to j leads to a distinct length $k+1$ walk from i to j. Hence,

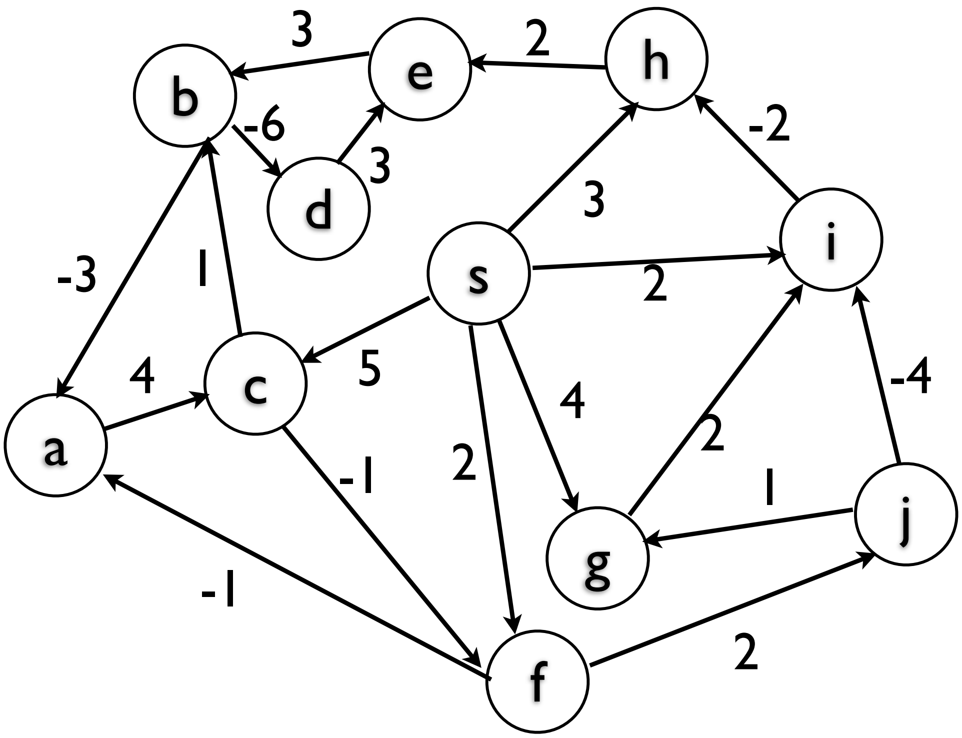
$$\#(k+1 \text{ length walks from } i \text{ to } j) \geq \sum_{j' : A_{j'j}=1} \#(k \text{ length walks from } i \text{ to } j').$$

Moreover, every $k+1$ length walk from i to j, can be decomposed into two parts, the k length walk starting at i and ending at j's neighbor and the edge from the neighbor to j. Furthermore, no two $k+1$ length walks can have the same decomposition, for otherwise, the two walks would be identical. Therefore,

$$\#(k+1 \text{ length walks from } i \text{ to } j) \leq \sum_{j' : A_{j'j}=1} \#(k \text{ length walks from } i \text{ to } j').$$

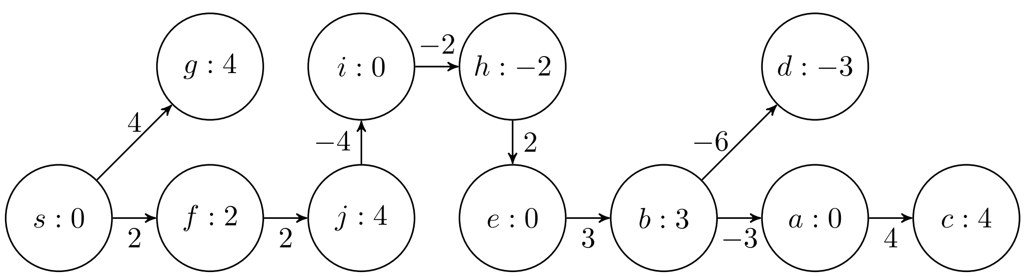
(b) **Runtime analysis:** Since each matrix multiplication is of runtime $O(n^w)$, a total of $\log k$ multiplications gives $n^w \log k$. Summing over the matrix is of order n^2, making the total runtime of the algorithm $O(n^w \log k)$.

3. Compute the shortest path tree for the following graph to find all shortest path distances from s:



You only need to show the shortest path tree for full credit.

Solution.



Note that the numbers inside the node are distances

4. Prove that in any tree with n vertices, the number of vertices with 3 or more neighbors is at most $2(n-1)/3$. Use the fact that every tree on n vertices has exactly $n-1$ edges, and apply the identity $\sum_v deg(v) = 2m$.

Solution. Let k be the number of vertices with $deg(v) \geq 3$. Then as in the hint we have

$$2(n-1) = \sum_v deg(v) \geq \sum_{v:deg(v)\geq 3} deg(v) \geq 3k.$$

But this means that $k \leq 2(n-1)/3$.