CSCE 629-601 Analysis of Algorithms

Fall 2022

Instructor: Dr. Jianer Chen
Office: PETR 428
Phone: (979) 845-4259
Email: `chen@cse.tamu.edu`
Office Hours: MWF 3:50 pm - 5:00 pm

Teaching Assistant: Vaibhav Bajaj
Office: EABC 107B
Phone: (979) 739-2707
Email: `vaibhavbajaj@tamu.edu`
Office Hours: T,TR 2:00 pm - 3:00 pm

Solutions to Assignment #4

1. Suppose that we have a sequence of MakeSet-Find-Union operations in which no Find appears before any Union. What is the computational time for this sequence?

Solutions. Suppose that the length of the sequence of MakeSet-Find-Union operations is m and that the number of elements in the set S is n. We claim that the computational time for this sequence is $O(m+n)$.

Each of the MakeSet and Union operations takes constant time. Thus, the total time for all MakeSet and Union operations is bounded by $O(m)$.

For the Find operations, we follow the same idea as presented in class, but define boundary/internal nodes differently. A node on a Find-path in a tree T is a *boundary node* if the node is a child of the root of T, and is an *internal node* if it is not a child of the root. Since there are at most m Find operations in the sequence, the total number of boundary nodes is bounded by m. Moreover, we claim that for any element a in the set S, there is at most one internal node labeled a. To see this, note that if a node v labeled a is an internal node on a Find-path, then the node v will become a child of the root r of the tree after the Find operation. Since no Union appears after any Find, the tree root r remains as a tree root for the rest of the computation, thus, the node v labeled a will remian as a child of the root r and cannot be an internal node. As a consequence, there are at most n internal nodes, each is labeled by a different element in the set S. Therefore, the total number of boundary nodes and internal nodes, which is equal to the sum of the lengths of Find-paths in the sequence, is bounded by $m+n$. As shown in class, this means that the total time spent on all the Find operations in the sequence is bounded by $O(m+n)$. Thus, the total computational time for the sequence is also bounded by $O(m+n)$. $\square$

2. Based on the Bellman-Ford algorithm, develop an algorithm of time $O(nm)$ that on a weighted and directed graph G and vertices s and t, either constructs a shortest path from s to t, or reports that t is not reachable from s, or reports that the shortest path problem from s to t is meaningless.

Solutions. This question requires a good understanding of the correctness proof for the Bellman-Ford algorithm. Let dist$[n]$ be the array computed by the Bellman-Ford algorithm for the vertex pair (s,t). As given in class for the proof of the correctness of the Bellman-Ford algorithm, for any vertex v, if there is no path from s to v that passes through a negative cycle, then dist$[v]$ is equal to the length of the shortest paths from s to v. Therefore, for any edge $[v,w]$, if we have dist$[w] >$ dist$[v]+$ wt(v, w), then either there is a path from s to v that passes through a negative cycle or there is a path from s to w that passes through a negative cycle. In either case, this gives a path from s to w that passes through a negative cycle. Moreover, this shows that the vertex w must be reachable from s.

Moreover, also shown in the proof of the correctness of the Bellman-Ford algorithm, every negative cycle contains at least one edge $[v, w]$ satisfying dist$[w] >$ dist$[v]+$ wt(v, w). Therefore, if we construct the set N that consists of all vertices w such that there is an edge $[v, w]$ making dist$[w] >$ dist$[v]+$ wt(v, w) (as noted above, this also implies that w is reachable from s), then for every vertex w in N, there is a path from s to w that passes through a negative cycle, and every negative cycle contains at least one vertex in N. Thus, there is a path from s to t that passes through a negative cycle if and only if there is a vertex w in N such that t is reachable from w.

Finally, it is obvious that dist$[t] = +\infty$ if and only if t is not reachable from s.

This idea is implemented in the following algorithm.

```
Enhanced Bellman-Ford(G, s, t)
1.  for (v = 1; v ≤ n; i++) dist[v] = +∞;
2.  dist[s] = 0; ;
3.  loop n-1 times
      for (each edge [v, w])
        if (dist[w] > dist[v] + wt(v, w))
          dad[w] = v;  dist[w] = dist[v] + wt(v, w);
4.  if (dist[t] = +∞) Stop ('t is not reachable from s.');
5.  N = ∅;
6.  for (each edge [v, w])
      if (dist[w] > dist[v] + wt(v, w))  \\ this implies dist[w] ≠ ∞
        N = N ∪ w;
7.  for (each vertex w in N)
      if (t is reacheable from w)
        Stop ('s-t shortest path is not meaningful.');
8.  use the dad array to output a shortest path from s to t.
```

Steps 1-6 and 8 are basically the Bellman-Ford algorithm, so they take time $O(nm)$. To implement step 7, we can apply DFS on each vertex v in N, which takes time $O(n+m)$. Thus, step 7 in total takes time $O(n(n+m)) = O(nm)$ (without loss of generality, we can assume $m \geq n-1$). Alternatively, we can construct the reversed graph G_R, then do a single DFS from t, in time $O(n+m)$ to find all those vertices in N that are reachable from t in the reversed graph G_R – these are the vertices in the original graph G from which t is reachable. In any case, step 7 takes time no more than $O(nm)$. In conclusion, the algorithm Enhanced Bellman-Ford solves the given problem in time $O(nm)$. $\square$

3. Based on the Floyd-Warshall algorithm, develop an algorithm of time $O(n^3)$ that on a weighted and directed graph G, constructs a matrix $C[1..n, 1..n]$ such that for each vertex pair (v, w):

1. $C[v, w] = +\infty$, if w is not reachable from v;
2. $C[v, w] =$ "meaningless",
   if the shortest path problem from v to w is meaningless; and
3. $C[v, w] = d$, if the weight of a shortest path from v to w is d.

Solution. The idea is similar to that of Question 2. For two vertices v and w, if there is a path from v to w that passes vertices in negative cycles, then the shortest path problem from v to w is meaningless. With the matrix $C^{(n)}$ constructed by the Floyd-Warshall algorithm, we can get the related information easily: (1) a vertex x is in a negative cycle if and only if $C^{(n)}[x,x] < 0$ (this can be proved by induction on the number of edges in the negative cycle); and (2) a path from v to w passes a vertex x in negative cycles if and only if there exist a path from v to x and a path from x to w, i.e., $C^{(n)}[v,x] \neq +\infty$ and $C^{(n)}[x,w] \neq +\infty$. These observations give the following modified Floyd-Warshall algorithm, where M is the $n \times n$ adjacency matrix of the graph G such that if $[v,w]$ is an edge in G then $M[v,w]$ is the weight of the edge, and if there is no edge from v to w then $M[v,w] = +\infty$:

```
Modified Floyd-Warshall(M)
1.   C^(0)[1..n, 1..n] = M[1..n, 1..n];
2.   for (k = 1; k ≤ n; k++)  \\ construct C^(k)[1..n, 1..n] from C^(k-1) [1..n, 1..n]
        for (v = 1; v ≤ n; v++)
          for (w = 1; w ≤ n; w++)
            if (C^(k-1)[v, w] < C^(k-1)[v, k] + C^(k-1)[k, w])
               C^(k)[v, w] = C^(k-1)[v, w];
            else C^(k)[v, w] = C^(k-1)[v, k] + C^(k-1)[k, w];
3.   N = ∅;
4.   for (x = 1; x ≤ n; x++)  \\ collect vertices in negative cycles
        if (C^(n)[x, x] < 0)  add x to N;
5.   for (v = 1; v ≤ n; v++)  \\ check meaningfulness of shortest path
        for (w = 1; w ≤ n; w++)
          for (each vertex x in N)
5.1          if ((C^(n)[v, x] ≠ +∞) & (C^(n)[x, w] ≠ +∞))
               C^(n)[v, w] = "meaningless";
6.   return the array C^(n)[1..n, 1..n].
```

Steps 1-2 are the original Floyd-Warshall algorithm that constructs the matrix $C^{(n)}[1..n, 1..n]$. Note that if vertex w is not reachable from vertex v then in the input matrix M, $M[v,w] = +\infty$, and in the matrix $C^{(n)}$, we also have $C^{(n)}[v,w] = +\infty$. Otherwise, $C^{(n)}[v,w] \neq +\infty$. Step 4 collects all vertices x in negative cycles in the graph G. In step 5, for each pair v and w of vertices, the condition

$$\text{x is in N, i.e., } (\mathrm{C}^{(n)}[\mathrm{x, x}] < 0) \text{ and } (\mathrm{C}^{(n)}[\mathrm{v, x}] \neq +\infty) \text{ and } (\mathrm{C}^{(n)}[\mathrm{x, w}] \neq +\infty)$$

tested in step 5.1 is necessary and sufficient for having a path from v to w that passes vertices in negative cycles, i.e., for the shortest path problem from v to w to be meaningless. Finally, it is easy to see that steps 3-6 also take time $O(n^3)$. In conclusion, the algorithm Modified Floyd-Warshall is an algorithm that solves the given problem and runs in time $O(n^3)$. $\square$