

CSC 301: Analysis of Algorithms

Week 9: Graphs, Union Find, Shortest Path, Topological Sort

William Rebelsky
[rebelskyw]

Grinnell College

Monday 8th April, 2024

Table of Contents

1 Monday: Graphs

- Graphs
- Shortest Paths

2 Wednesday: MST

- Union-Find
- Union-Find
- Minimum Spanning Trees
- Topological Sort
- Strongly Connected Components

3 Friday: Discussion

- Chapter 20
- Chapter 21

CSC 301: Analysis of Algorithms

Class 28: Graphs, Shortest Paths

William Rebelsky
[rebelskyw]

Grinnell College

Monday 8th April, 2024

Reminders

- Essay is due Wednesday
- Homework 6 is due Wednesday
- Quiz 6 is due tonight
- Token opportunities: CS extras, CS tables, Convocation, Any talk hosted by the CS department, etc

Graph Basics (Review?)

- In the text: $G=(V,E)$ where G is the Graph, V is the set of vertices, and E is the set of edges
- Runtime is normally in terms of both V and E
- E is $O(|V|^2)$
- Stored in either Adjacency Lists (more efficient on sparse graphs) or an Adjacency Matrix
- Graphs may be directed or undirected and weighted or unweighted
- $v.d$ is attribute d of vertex v
- $(u,v).f$ is attribute f of edge (u,v) which is the edge from vertex u to vertex v

Breadth First Search

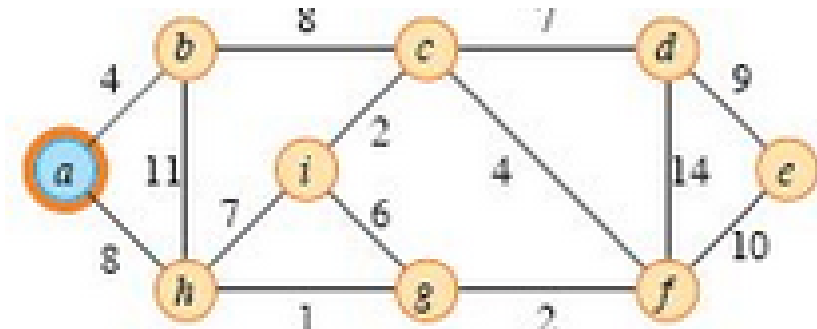
- Find all vertices reachable from starting vertex s
- Start with those of distance 1, then 2, then...
- Vertices are colored White, Gray, or Black
 - White: vertices not yet reached
 - Gray: vertices currently on the boundary
 - Black: vertices where all edges have been explored
- Use a FOFO Queue
- Output: BFS tree
- If the edge (u,v) is added to the tree, u is the predecessor or parent of v
- Three vertex attributes:
 - $v.color$
 - $v.d$ (distance from s to v)
 - $v.\pi$, v 's parent
- $\Theta(V + E)$

Depth First Search

- Find all vertices reachable from source vertex s
- Follow paths to conclusion before backtracking
- Same coloring as BFS
- Also stores two time stamps. $v.d$ when first discovered (and grayed), $v.f$ when finished (and blackened)
- Edge classification:
 - Tree edges: edges in the forest. Edge (u,v) is a tree edge if v was first discovered by exploring edge (u,v) . Vertex v was white
 - Back edges: edges (u,v) connecting vertex u to an ancestor v . If there are no back edges, then a directed graph is acyclic (Lemma 20.11). Vertex v was gray
 - Forward edges: non tree edges (u,v) connecting a vertex u to a proper decendent v . Vertex v was black
 - Cross edges: all other edges. Vertex v was black
- Use a stack (see problem 20.3-6)
- $\Theta(V + E)$

Graph Basics Practice

- 20.1-2 Give an adjacency list representation for a complete binary tree on 7 vertices. Give an equivalent adjacency matrix representation. Assume edges are undirected and numbered 1 to 7 as in a binary heap
- 20.2-1 Show the d and π values that result from running BFS on the below image
- 20.3-2 Show how DFS works on the below image



Shortest Paths Variants and Basics

- Variants:
 - Single source shortest path: shortest path to all nodes from source s (see 20.2, All of 22)
 - Single destination shortest path: shortest path from all nodes from sink t . Just invert the above
 - Single pair shortest path: can be solved with Single source shortest path, and has same worst case runtime
 - All pairs shortest paths: Shortest path from u to v for all pairs (chapter 23)
- The algorithms tend to rely on the optimal substructure property and assume no negative weight cycles
- Solution returned is the predecessor subgraph G_π which is a shortest path tree
- Relaxation: (u, v, w) : if $v.d > u.d + w(u, v)$,
 $v.d = u.d + w(u, v)$, $v.\pi = u$

Single Source and Relaxation Properties

- Triangle inequality: $\delta(s, v) \leq \delta(s, u) + w(u, v)$
- Upper bound property: $v.d \geq \delta(s, v)$
- No path property: $v.d = \delta(s, v) = \infty$
- Convergence property: if $s - u - v$ is a shortest path for some u, v and if $\delta(s, u) = u.d$ prior to relaxing edge (u, v) then $v.d = \delta(s, v)$ at all times afterwards
- Path relaxation property: if $p = (v_0, v_1, \dots, v_k)$ is a shortest path from $s = v_0$ to v_k and the edges are relaxed in order, then $v_k.d = \delta(s, v_k)$
- Predecessor subgraph property: Once $v.d = \delta(s, v)$ for all v in V , the predecessor subgraph is a shortest paths tree rooted at s

General Single Source Shortest Path (20.2)

- $\delta(s, v)$: minimum number of edges from s to v
- Lemma 20.1 For any edge (u, v) , $\delta(s, v) \leq \delta(s, u) + 1$
- Lemma 20.2 If you run BFS on graph G with source s ,
 $v.d \geq \delta(s, v)$
- Overall: The distances and paths computed by a BFS are the shortest paths from s to v

Bellman-Ford (22.1)

- No negative weight cycles
- Steps:
 - initialize d and π values
 - For i in range of vertices-1: relax all edges
- $O(V^2 + VE)$

Dijkstra's Algorithm (22.1)

- No negative weight edges
- Faster than Bellman-Ford
- Extended BFS, uses min priority queue keyed by d values instead of FOFO
- Steps:
 - initialize d and π values
 - Insert into the priority Queue Q
 - Extract min value, u , of Q
 - add u to S
 - Relax every edge u,v,w
 - If the call decreased v,d , decrease $\text{key}(Q,v,v.d)$
- Greedy algorithm (choose the closest vertex)

All Paths Shortest Paths: Floyd-Warshall (23.2)

- We will not cover this in this course, but it is worth reading if you are curious

Shortest Path Practice

- 22.1-1 Run the Bellman-Ford Algorithm on the directed graph of Figure 22.4 using vertex z as the source. Show the d and π values after each pass. Then change the weight of edge(z,x) to 4 and do it again using s as the source.
- 22.3-1 Run Dijkstra's algorithm on the directed graph of Figure 22.2 first using s as the source then using z as the source.

Table of Contents

1 Monday: Graphs

- Graphs
- Shortest Paths

2 Wednesday: MST

- Union-Find
- Union-Find
- Minimum Spanning Trees
- Topological Sort
- Strongly Connected Components

3 Friday: Discussion

- Chapter 20
- Chapter 21

CSC 301: Analysis of Algorithms

Class 29: MST

William Rebelsky
[rebelskyw]

Grinnell College

Wednesday 10th April, 2024

Reminders

- Essay is due Tonight
- Homework 6 is due Tonight
- Homework 7 is released and due next Wednesday
- Quiz 7 will be released Friday

Union Find/Disjoint Sets (Chapter 19)

- ADT: Support two operations (three)
 - Make-Set(x): create a new set whose only member and thus representative is x
 - Union(x,y): Unite two disjoint sets S_x and S_y . Resulting representative can be any element but is generally the representative of either of the two sets. Destroys both smaller sets to ensure uniqueness, generally implemented by having one set absorb the other
 - Find-Set(x): returns a pointer to the representative of the unique set containing x

Representing Disjoint Sets

- Linked list representation: each set has attributes head (point to first object) and tail (point to last object). Each object in the list contains a set member, a pointer to the next object in the list, and a pointer back to the set object. Make set and Find set are both $O(1)$, Union takes more work.
- Rooted trees representation: Forest where each tree represents a set, and each member is a node. Each member points only to its parent. Set: Easy, Find: follow parents, Union: Point Root of S_x to root of S_y or vice versa
 - Improve running time: union by rank, have smaller tree point to bigger, increase the rank by 1. Path compression: When performing find-set, if the parent is not the root, change the parent to the result of Find-Set(x.parent)
 - Ends with a worse case running time of $O(m\alpha(n))$ where for all conceivable applications $\alpha(n) \leq 4$

Minimum Spanning Tree Basics

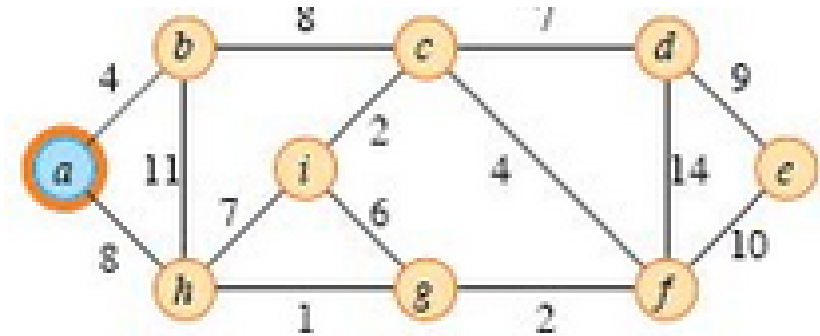
- Spanning Tree: Acyclic Tree that connects all Vertices of connected undirected Graph $G=(V,E)$
- Minimum Spanning Tree: Find the Spanning Tree of minimum weight
- Normally solved with greedy algorithms:
 - Kruskal's algorithm: $O(E \lg V)$
 - Prim's Algorithm: $O(E \lg V)$ if using a binary heap as a priority queue. $O(E + V \lg V)$ is possible using Fibonacci heaps instead
- Generic MST: While A is not a spanning tree, find an edge that is safe, add it to A
- The hard part is finding the safe edge
- A CUT $(S, V-S)$ is a partition of V . An Edge e crosses the cut if one end point is in S and one endpoint is in $V-S$. An edge is a light edge if it is the minimum of any edge crossing the cut. A light edge is safe (Theorem 21.1)

Kruskal's Algorithm vs Prim's Algorithm

- Kruskal:
 - Keep track of a forest, A , of all vertices in graph G
 - Sort edges in G by weight
 - In each step, choose the lowest weight edge in the graph that connects two distinct components in A
 - Uses Disjoint-set or union-find data structure
- Prim:
 - Keep track of a tree, A , which is the vertices found so far
 - Add light edge between sets $(A, V-A)$
 - Uses a min priority queue of vertices not in the tree

Minimum Spanning Tree Practice

- Run Kruskal's Algorithm on the Graph below
- Run Prim's Algorithm on the Graph below starting at node a



Topological Sort Basics

- Requires a DAG
- Steps:
 - 1 Call DFS to compute finish times $v.f$ for each vertex
 $\Theta(V + E)$
 - 2 As each vertex is finished, insert it onto the front of a linked list $O(1) * |V|$
 - 3 Return the linked list
- $\Theta(V + E)$
- Uses: determining which event must occur before another event

Topological Sort Practice

- 20.4-1 Show Topological Sort on Figure 20.8

Strongly Connected Components

- Use DFS to decompose a directed graph into strongly connected components. That is, for every pair of vertices (u,v) , there is a path from v to u and u to v
- The transpose of a graph $G = (V, E)$ is $G^T(V, E^T)$ where E^T is the set of edges (u,v) such that edge (v,u) is in E
- G^T can be computed in $\Theta(V + E)$
- Both G and G^T have the same Strongly Connected Components
- Steps:
 - 1 Call $\text{DFS}(G)$ to compute finish times $u.f$ for each vertex u
 - 2 Create G^T
 - 3 Call $\text{DFS}(G^T)$, but in the main loop consider the vertices in order of decreasing $u.f$
 - 4 Output the vertices of each tree in the Depth First Forest in line 3 as a separate SCC

Strongly Connected Components Practice

- 20.5-1 How can the number of SCC change if a new edge is added?

Table of Contents

1 Monday: Graphs

- Graphs
- Shortest Paths

2 Wednesday: MST

- Union-Find
- Union-Find
- Minimum Spanning Trees
- Topological Sort
- Strongly Connected Components

3 Friday: Discussion

- Chapter 20
- Chapter 21

CSC 301: Analysis of Algorithms

Class 30: Discussion

William Rebelsky
[rebelskyw]

Grinnell College

Friday 12th April, 2024

Reminders

- Homework 7 is due Wednesday
- Quiz 7 is released

Discussion Questions Chapter 20

- 20.3-7 Give a counterexample to the conjecture that if a directed graph G contains a path from u to v , and if $u.d < v.d$ in a DFS of G , then v is a descendant of u in the dfs forest produced.
- 20.3-8 Give a counterexample to the conjecture that if a directed graph G contains a path from u to v , then any dfs must result in $v.d \leq u.f$
- 20.5-6 Given an $O(V+E)$ time algorithm that given a directed graph $G=(V,E)$ constructs another graph $G'=(V,E')$ such that both G and G' have the same strongly connected components but $|E'|$ is as small as possible

Challenge Discussion Question Chapter 20

- 20.3-11 Let $G=(V,E)$ be a connected undirected graph. Give an $O(V+E)$ time algorithm to compute a path in G that traverses each edge in E exactly once in each direction. Describe how you can find your way out of a maze if you are given a large supply of pennies

Discussion Questions Chapter 21

- 21.1-3 Show that if an edge (u,v) is contained in some MST then it is a light edge crossing some cut of the graph
- 21.1-5 Let e be a maximum weight edge on some cycle of connected graph $G=(V,E)$. Prove that there is an MST of $G' = (V, E - e)$ that is also an MST of G .
- 21.1-6 Show that a graph has a unique MST if for every cut of the graph there is a unique light edge crossing the cut. Give a counterexample to show the converse is not true
- 21.1-7 Argue that if all edge weights of a graph are positive, then any subset of edges that connects all vertices and has minimum total weight must be a tree. Give an example to show that this is not true if we allow nonpositive weights

Challenge Discussion Question Chapter 21

- 21-1 Second best MST. Suppose that we have an undirected, connected graph with all real edge weights that are distinct. (a) Show that the Mst is unique. (b) Let the second best MST be the spanning tree with the minimum weight that is not the MST. Show that it need not be unique. (c) Let T be the MST of G . Prove that G contains an edge (u,v) in T and an edge (x,y) not in T such that $T - (u,v) + (x,y)$ is a second best MST of G