CMPSCI 611: Advanced Algorithms

Lecture 4: Greedy Algorithms and Matroids

Andrew McGregor

Last Compiled: February 13, 2024

Greedy Algorithms Overview

"An algorithm that finds a solution by adding elements one by one, where each element that is added is the best current choice without regard to the future consequences of this choice."

Greedy Algorithms Overview

"An algorithm that finds a solution by adding elements one by one, where each element that is added is the best current choice without regard to the future consequences of this choice."

- Minimum Spanning Tree and Kruskal's algorithm
- Matroids and Subset Systems
- Bipartite Matching and Intersections of Matroids
- Union-Find Data Structure

Minimum Spanning Tree and Kruskal's Algorithm

Problem: Given an undirected, connected graph (V, E) with edge weights find the min-weight subset $E' \subset E$ such that the graph (V, E') is acyclic and connected, i.e., a min-weight spanning tree (MST).

Throughout this class we'll assume all edge weights are distinct although everything generalizes to when some weights are the same.

Minimum Spanning Tree and Kruskal's Algorithm

Problem: Given an undirected, connected graph (V, E) with edge weights find the min-weight subset $E' \subset E$ such that the graph (V, E') is acyclic and connected, i.e., a min-weight spanning tree (MST).

Throughout this class we'll assume all edge weights are distinct although everything generalizes to when some weights are the same.

Algorithm (Kruskal)

1. *Sort edges by increasing weight*
2. $F = \emptyset$
3. *Until F is a spanning tree of G*
    - 3.1 *Get the next edge e*
    - 3.2 *If $F + e$ is acyclic then $F = F + e$*

Minimum Spanning Tree and Kruskal's Algorithm

Problem: Given an undirected, connected graph (V, E) with edge weights find the min-weight subset $E' \subset E$ such that the graph (V, E') is acyclic and connected, i.e., a min-weight spanning tree (MST).

Throughout this class we'll assume all edge weights are distinct although everything generalizes to when some weights are the same.

Algorithm (Kruskal)

1. *Sort edges by increasing weight*
2. $F = \emptyset$
3. *Until F is a spanning tree of G*
   3.1 *Get the next edge e*
   3.2 *If $F + e$ is acyclic then $F = F + e$*

The algorithm produces a tree because a) it never completes a cycle so end result is acyclic and b) it is connected since for any cut, algorithm adds at least the first edge it encounters across this cut.

Running Time of Kruskal's Algorithm

Implementation: Maintain an array A with an entry for each $v \in V$ that indicates which connected component it belongs to.

Running Time of Kruskal's Algorithm

Implementation: Maintain an array A with an entry for each $v \in V$ that indicates which connected component it belongs to.

- Initially $A[i] = i$ for $i = 1$ to $|V|$.

Running Time of Kruskal's Algorithm

Implementation: Maintain an array A with an entry for each $v \in V$ that indicates which connected component it belongs to.

- Initially $A[i] = i$ for $i = 1$ to $|V|$.
- When edge (v_i, v_j) is processed:
  - If $A[i] \neq A[j]$, add (v_i, v_j) to F
  - Replace array entries equal to $\max(A[i], A[j])$ by $\min(A[i], A[j])$

Running Time of Kruskal's Algorithm

Implementation: Maintain an array A with an entry for each $v \in V$ that indicates which connected component it belongs to.

- Initially $A[i] = i$ for $i = 1$ to $|V|$.
- When edge (v_i, v_j) is processed:
  - If $A[i] \neq A[j]$, add (v_i, v_j) to F
  - Replace array entries equal to $\max(A[i], A[j])$ by $\min(A[i], A[j])$

Running Time:

- Sorting: $O(|E| \log |E|)$

Running Time of Kruskal's Algorithm

Implementation: Maintain an array A with an entry for each $v \in V$ that indicates which connected component it belongs to.

- Initially $A[i] = i$ for $i = 1$ to $|V|$.
- When edge (v_i, v_j) is processed:
  - If $A[i] \neq A[j]$, add (v_i, v_j) to F
  - Replace array entries equal to $\max(A[i], A[j])$ by $\min(A[i], A[j])$

Running Time:

- Sorting: $O(|E| \log |E|)$
- Checking if acyclic: $|E|$ checks and each is $O(1)$ time.

Running Time of Kruskal's Algorithm

Implementation: Maintain an array A with an entry for each $v \in V$ that indicates which connected component it belongs to.

- Initially $A[i] = i$ for $i = 1$ to $|V|$.
- When edge (v_i, v_j) is processed:
  - If $A[i] \neq A[j]$, add (v_i, v_j) to F
  - Replace array entries equal to $\max(A[i], A[j])$ by $\min(A[i], A[j])$

Running Time:

- Sorting: $O(|E| \log |E|)$
- Checking if acyclic: $|E|$ checks and each is $O(1)$ time.
- Adding e to F: Updating array takes $O(|V|)$ time and array is updated exactly $|V| - 1$ times.
- Total running time $O(|E| \log |E| + |V|^2)$

Running Time of Kruskal's Algorithm

Implementation: Maintain an array A with an entry for each $v \in V$ that indicates which connected component it belongs to.

- Initially $A[i] = i$ for $i = 1$ to $|V|$.
- When edge (v_i, v_j) is processed:
  - If $A[i] \neq A[j]$, add (v_i, v_j) to F
  - Replace array entries equal to $\max(A[i], A[j])$ by $\min(A[i], A[j])$

Running Time:

- Sorting: $O(|E| \log |E|)$
- Checking if acyclic: $|E|$ checks and each is $O(1)$ time.
- Adding e to F: Updating array takes $O(|V|)$ time and array is updated exactly $|V| - 1$ times.
- Total running time $O(|E| \log |E| + |V|^2)$

Will make this $O(|E| \log |E|)$ later via the union-find data structure

Proof of Correctness: Part 1

Cut Lemma: Let $S \subset V$ and let $e = (u, v)$ be the lightest edge such that $u \in S$ and $v \notin S$. The MST contains edge e.

Proof of Correctness: Part 1

Cut Lemma: Let $S \subset V$ and let $e = (u, v)$ be the lightest edge such that $u \in S$ and $v \notin S$. The MST contains edge e.

Proof:

Proof of Correctness: Part 1

Cut Lemma: Let $S \subset V$ and let $e = (u, v)$ be the lightest edge such that $u \in S$ and $v \notin S$. The MST contains edge e.

Proof:

- Suppose there exists a minimum spanning tree T that doesn't include e. We'll construct a different spanning tree T' such that $w(T') < w(T)$ and hence T can't be the MST.

Proof of Correctness: Part 1

Cut Lemma: Let $S \subset V$ and let $e = (u, v)$ be the lightest edge such that $u \in S$ and $v \notin S$. The MST contains edge e.

Proof:

- Suppose there exists a minimum spanning tree T that doesn't include e. We'll construct a different spanning tree T' such that $w(T') < w(T)$ and hence T can't be the MST.
- Since T is a spanning tree, there's a $u \rightsquigarrow v$ path P in T. Since the path starts in S and ends up outside S, there must be an edge $e' = (u', v')$ on this path where $u' \in S, v' \notin S$.

Proof of Correctness: Part 1

Cut Lemma: Let $S \subset V$ and let $e = (u, v)$ be the lightest edge such that $u \in S$ and $v \notin S$. The MST contains edge e.

Proof:

- Suppose there exists a minimum spanning tree T that doesn't include e. We'll construct a different spanning tree T' such that $w(T') < w(T)$ and hence T can't be the MST.
- Since T is a spanning tree, there's a $u \rightsquigarrow v$ path P in T. Since the path starts in S and ends up outside S, there must be an edge $e' = (u', v')$ on this path where $u' \in S, v' \notin S$.
- Let $T' = T - \{e'\} + \{e\}$. This is still spanning tree, since any path in T that needed e' can be routed via e instead. But since e was the lightest edge between S and $V \setminus S$,

$$w(T') = w(T) - w(e') + w(e) < w(T) - w(e') + w(e') = w(T)$$

Proof of Correctness: Part 2

Kruskal's Algorithm: Sort the edges by increasing weight and keep on add the next edge that doesn't complete a cycle.

Proof of Correctness:

Proof of Correctness: Part 2

Kruskal's Algorithm: Sort the edges by increasing weight and keep on add the next edge that doesn't complete a cycle.

Proof of Correctness:

- Suppose $e = (u, v)$ is the next edge added.

Proof of Correctness: Part 2

Kruskal's Algorithm: Sort the edges by increasing weight and keep on add the next edge that doesn't complete a cycle.

Proof of Correctness:

- Suppose $e = (u, v)$ is the next edge added.
- Let S be the set of nodes that can be reached from u before e was added. Note that $v \notin S$ since otherwise adding e would have completed a cycle.

Proof of Correctness: Part 2

Kruskal's Algorithm: Sort the edges by increasing weight and keep on add the next edge that doesn't complete a cycle.

Proof of Correctness:

- Suppose $e = (u, v)$ is the next edge added.
- Let S be the set of nodes that can be reached from u before e was added. Note that $v \notin S$ since otherwise adding e would have completed a cycle.
- No other edge between S and $V \setminus S$ has been encountered before since if it had it would have been added since it doesn't complete a cycle. Hence e is the lightest edge between S and $V \setminus S$. Therefore, the cut lemma implies e must be in the MST.

A Different Greedy Algorithm: Prim's Algorithm

Prim's Algorithm:

A Different Greedy Algorithm: Prim's Algorithm

Prim's Algorithm:

- Sort the edges by increasing weight.

A Different Greedy Algorithm: Prim's Algorithm

Prim's Algorithm:

- Sort the edges by increasing weight.
- Let $S = \{s\}$.

A Different Greedy Algorithm: Prim's Algorithm

Prim's Algorithm:

- Sort the edges by increasing weight.
- Let $S = \{s\}$.
- While $S \neq V$: Add next edge (u, v) where $u \in S, v \notin S$ and add v to S.

Proof of Correctness:

A Different Greedy Algorithm: Prim's Algorithm

Prim's Algorithm:

- Sort the edges by increasing weight.
- Let $S = \{s\}$.
- While $S \neq V$: Add next edge (u, v) where $u \in S, v \notin S$ and add v to S.

Proof of Correctness:

- Let S be the set of nodes in the tree constructed so far.

A Different Greedy Algorithm: Prim's Algorithm

Prim's Algorithm:

- Sort the edges by increasing weight.
- Let $S = \{s\}$.
- While $S \neq V$: Add next edge (u, v) where $u \in S, v \notin S$ and add v to S.

Proof of Correctness:

- Let S be the set of nodes in the tree constructed so far.
- The next edge added to the tree is the lightest edge between S and $V \setminus S$. Hence, the cut lemma implies e must be in the MST.