

CMPSCI 611: Advanced Algorithms

Lecture 4: Greedy Algorithms and Matroids

Andrew McGregor

Last Compiled: September 18, 2009

Outline

Greedy Algorithms and Matroids

- Minimum Spanning Tree and Kruskal's Algorithm

- Subset Systems

- Matroids

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

Outline

Greedy Algorithms and Matroids

Minimum Spanning Tree and Kruskal's Algorithm

Subset Systems

Matroids

Minimum Spanning Tree and Kruskal's Algorithm

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

Minimum Spanning Tree and Kruskal's Algorithm

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

Algorithm (Kruskal)

1. *Sort edges by non-decreasing 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 $G = (V, E)$ with positive edge weights, find the minimum-weight subset $E' \subset E$ such that the graph $G = (V, E')$ is acyclic and connected, i.e., a minimum spanning tree (MST).

Algorithm (Kruskal)

1. Sort edges by non-decreasing 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 $F = F + e$

Easy to implement with $O(|E| \log |E| + |V|^2)$ running time. Later, we'll improve to $O(|E| \log |E|)$ via the union-find data structure.

Correctness of Kruskal's Algorithm (1/2)

Given an unconnected forest F , let $S(F)$ be the set of spanning trees that extend F .

Lemma

Suppose $S(F)$ contains an MST of G and e is the minimum weight edge not in F and not causing a cycle in F . Then $S(F + e)$ also contains a minimum spanning tree for G .

Correctness of Kruskal's Algorithm (1/2)

Given an unconnected forest F , let $S(F)$ be the set of spanning trees that extend F .

Lemma

Suppose $S(F)$ contains an MST of G and e is the minimum weight edge not in F and not causing a cycle in F . Then $S(F + e)$ also contains a minimum spanning tree for G .

Proof.

- ▶ Let $T' \in S(F)$ be an MST. Assume it doesn't include e (otherwise we're done)

Correctness of Kruskal's Algorithm (1/2)

Given an unconnected forest F , let $S(F)$ be the set of spanning trees that extend F .

Lemma

Suppose $S(F)$ contains an MST of G and e is the minimum weight edge not in F and not causing a cycle in F . Then $S(F + e)$ also contains a minimum spanning tree for G .

Proof.

- ▶ Let $T' \in S(F)$ be an MST. Assume it doesn't include e (otherwise we're done)
- ▶ Adding e to T' makes a cycle C

Correctness of Kruskal's Algorithm (1/2)

Given an unconnected forest F , let $S(F)$ be the set of spanning trees that extend F .

Lemma

Suppose $S(F)$ contains an MST of G and e is the minimum weight edge not in F and not causing a cycle in F . Then $S(F + e)$ also contains a minimum spanning tree for G .

Proof.

- ▶ Let $T' \in S(F)$ be an MST. Assume it doesn't include e (otherwise we're done)
- ▶ Adding e to T' makes a cycle C
- ▶ Since e doesn't make a cycle in F , there exists $e' \in C$ with $e' \notin F$. Note that $w(e) \leq w(e')$.

Correctness of Kruskal's Algorithm (1/2)

Given an unconnected forest F , let $S(F)$ be the set of spanning trees that extend F .

Lemma

Suppose $S(F)$ contains an MST of G and e is the minimum weight edge not in F and not causing a cycle in F . Then $S(F + e)$ also contains a minimum spanning tree for G .

Proof.

- ▶ Let $T' \in S(F)$ be an MST. Assume it doesn't include e (otherwise we're done)
- ▶ Adding e to T' makes a cycle C
- ▶ Since e doesn't make a cycle in F , there exists $e' \in C$ with $e' \notin F$. Note that $w(e) \leq w(e')$.
- ▶ Then $T = T' + e - e'$ is a MST in $S(F + e)$:

Correctness of Kruskal's Algorithm (1/2)

Given an unconnected forest F , let $S(F)$ be the set of spanning trees that extend F .

Lemma

Suppose $S(F)$ contains an MST of G and e is the minimum weight edge not in F and not causing a cycle in F . Then $S(F + e)$ also contains a minimum spanning tree for G .

Proof.

- ▶ Let $T' \in S(F)$ be an MST. Assume it doesn't include e (otherwise we're done)
- ▶ Adding e to T' makes a cycle C
- ▶ Since e doesn't make a cycle in F , there exists $e' \in C$ with $e' \notin F$. Note that $w(e) \leq w(e')$.
- ▶ Then $T = T' + e - e'$ is a MST in $S(F + e)$:
 - ▶ T is a spanning tree in $S(F + e)$

Correctness of Kruskal's Algorithm (1/2)

Given an unconnected forest F , let $S(F)$ be the set of spanning trees that extend F .

Lemma

Suppose $S(F)$ contains an MST of G and e is the minimum weight edge not in F and not causing a cycle in F . Then $S(F + e)$ also contains a minimum spanning tree for G .

Proof.

- ▶ Let $T' \in S(F)$ be an MST. Assume it doesn't include e (otherwise we're done)
- ▶ Adding e to T' makes a cycle C
- ▶ Since e doesn't make a cycle in F , there exists $e' \in C$ with $e' \notin F$. Note that $w(e) \leq w(e')$.
- ▶ Then $T = T' + e - e'$ is a MST in $S(F + e)$:
 - ▶ T is a spanning tree in $S(F + e)$
 - ▶ T is a MST because $w(T) = w(T') + w(e) - w(e') \leq w(T')$



Correctness of Kruskal's Algorithm (1/2)

Easy to see that the algorithm produces a spanning tree. . .

Correctness of Kruskal's Algorithm (1/2)

Easy to see that the algorithm produces a spanning tree. . . Why is the result acyclic and why is the result not disconnected?

Correctness of Kruskal's Algorithm (1/2)

Easy to see that the algorithm produces a spanning tree. . . Why is the result acyclic and why is the result not disconnected?

Theorem

Every forest F produced by Kruskal's algorithm is such that $S(F)$ contains a MST of G .

Correctness of Kruskal's Algorithm (1/2)

Easy to see that the algorithm produces a spanning tree. . . Why is the result acyclic and why is the result not disconnected?

Theorem

Every forest F produced by Kruskal's algorithm is such that $S(F)$ contains a MST of G .

Proof.

- ▶ Base Case: If $F = \emptyset$, $S(F)$ contains all the spanning trees including the minimum spanning trees.

Correctness of Kruskal's Algorithm (1/2)

Easy to see that the algorithm produces a spanning tree. . . Why is the result acyclic and why is the result not disconnected?

Theorem

Every forest F produced by Kruskal's algorithm is such that $S(F)$ contains a MST of G .

Proof.

- ▶ Base Case: If $F = \emptyset$, $S(F)$ contains all the spanning trees including the minimum spanning trees.
- ▶ Induction step: By the previous lemma, if $S(F)$ contains a MST for G , then so does $S(F + e)$



Correctness of Kruskal's Algorithm (1/2)

Easy to see that the algorithm produces a spanning tree. . . Why is the result acyclic and why is the result not disconnected?

Theorem

Every forest F produced by Kruskal's algorithm is such that $S(F)$ contains a MST of G .

Proof.

- ▶ Base Case: If $F = \emptyset$, $S(F)$ contains all the spanning trees including the minimum spanning trees.
- ▶ Induction step: By the previous lemma, if $S(F)$ contains a MST for G , then so does $S(F + e)$



Since the final forest is actually a tree T , then $S(F) = T$ is a MST.

Outline

Greedy Algorithms and Matroids

Minimum Spanning Tree and Kruskal's Algorithm

Subset Systems

Matroids

Subset Systems

Definition

A *subset system* $S = (E, \mathcal{I})$ is a finite set E with a collection \mathcal{I} of subsets E such that:

$$\text{if } i \in \mathcal{I} \text{ and } i' \subset i \text{ then } i' \in \mathcal{I}$$

i.e., “ \mathcal{I} is closed under inclusion”

Subset Systems

Definition

A *subset system* $S = (E, \mathcal{I})$ is a finite set E with a collection \mathcal{I} of subsets E such that:

$$\text{if } i \in \mathcal{I} \text{ and } i' \subset i \text{ then } i' \in \mathcal{I}$$

i.e., “ \mathcal{I} is closed under inclusion”

Example

1. $E = \{e_1, e_2, e_3\}$, $\mathcal{I} = \{\{e_1, e_2\}, \{e_2, e_3\}, \{e_1\}, \{e_2\}, \{e_3\}, \{\}\}$

Subset Systems

Definition

A *subset system* $S = (E, \mathcal{I})$ is a finite set E with a collection \mathcal{I} of subsets E such that:

$$\text{if } i \in \mathcal{I} \text{ and } i' \subset i \text{ then } i' \in \mathcal{I}$$

i.e., “ \mathcal{I} is closed under inclusion”

Example

1. $E = \{e_1, e_2, e_3\}$, $\mathcal{I} = \{\{e_1, e_2\}, \{e_2, e_3\}, \{e_1\}, \{e_2\}, \{e_3\}, \{\}\}$
2. E is the edges of a graph and \mathcal{I} is the acyclic subsets of edges

Subset Systems

Definition

A *subset system* $S = (E, \mathcal{I})$ is a finite set E with a collection \mathcal{I} of subsets E such that:

$$\text{if } i \in \mathcal{I} \text{ and } i' \subset i \text{ then } i' \in \mathcal{I}$$

i.e., “ \mathcal{I} is closed under inclusion”

Example

1. $E = \{e_1, e_2, e_3\}$, $\mathcal{I} = \{\{e_1, e_2\}, \{e_2, e_3\}, \{e_1\}, \{e_2\}, \{e_3\}, \{\}\}$
2. E is the edges of a graph and \mathcal{I} is the acyclic subsets of edges
3. E is the edges of a graph and \mathcal{I} are the **matchings**, i.e., subsets of edges such that no two edges share a vertex

Generic Problem and Greedy Algorithms

Problem Given a subset system $S = (E, \mathcal{I})$ and weight function $w : E \rightarrow \mathbb{R}^+$, find $i \in \mathcal{I}$ such that $w(i) = \sum_{e \in i} w(e)$ is maximized.

Generic Problem and Greedy Algorithms

Problem Given a subset system $S = (E, \mathcal{I})$ and weight function $w : E \rightarrow \mathbb{R}^+$, find $i \in \mathcal{I}$ such that $w(i) = \sum_{e \in i} w(e)$ is maximized.

Algorithm (Greedy)

1. $i = \emptyset$
2. Sort elements of E by non-increasing weight
3. For each $e \in E$: If $i + e \in \mathcal{I}$ then $i = i + e$

Generic Problem and Greedy Algorithms

Problem Given a subset system $S = (E, \mathcal{I})$ and weight function $w : E \rightarrow \mathbb{R}^+$, find $i \in \mathcal{I}$ such that $w(i) = \sum_{e \in i} w(e)$ is maximized.

Algorithm (Greedy)

1. $i = \emptyset$
2. Sort elements of E by non-increasing weight
3. For each $e \in E$: If $i + e \in \mathcal{I}$ then $i = i + e$

For what subset systems does this give optimal results?

Examples

Example

Let $E = \{e_1, e_2, e_3\}$, $\mathcal{I} = \{\{e_1, e_2\}, \{e_2, e_3\}, \{e_1\}, \{e_2\}, \{e_3\}, \{\}\}$, and $w(e_1) = 3$, $w(e_2) = 1$, and $w(e_3) = 4$. The greedy algorithm returns. . .

Examples

Example

Let $E = \{e_1, e_2, e_3\}$, $\mathcal{I} = \{\{e_1, e_2\}, \{e_2, e_3\}, \{e_1\}, \{e_2\}, \{e_3\}, \{\}\}$, and $w(e_1) = 3$, $w(e_2) = 1$, and $w(e_3) = 4$. The greedy algorithm returns. . . $\{e_2, e_3\}$ and this is optimal.

Examples

Example

Let $E = \{e_1, e_2, e_3\}$, $\mathcal{I} = \{\{e_1, e_2\}, \{e_2, e_3\}, \{e_1\}, \{e_2\}, \{e_3\}, \{\}\}$, and $w(e_1) = 3$, $w(e_2) = 1$, and $w(e_3) = 4$. The greedy algorithm returns... $\{e_2, e_3\}$ and this is optimal.

Example (Maximum Weight Forest)

E is the edges of a graph and \mathcal{I} is the acyclic subsets of edges. This is essentially the same as the MST and greedy does work.

Examples

Example

Let $E = \{e_1, e_2, e_3\}$, $\mathcal{I} = \{\{e_1, e_2\}, \{e_2, e_3\}, \{e_1\}, \{e_2\}, \{e_3\}, \{\}\}$, and $w(e_1) = 3$, $w(e_2) = 1$, and $w(e_3) = 4$. The greedy algorithm returns... $\{e_2, e_3\}$ and this is optimal.

Example (Maximum Weight Forest)

E is the edges of a graph and \mathcal{I} is the acyclic subsets of edges. This is essentially the same as the MST and greedy does work.

Example (Maximum Weight Matching)

E is the edges of a graph and \mathcal{I} are the **matchings**. Greedy does not work.

Outline

Greedy Algorithms and Matroids

Minimum Spanning Tree and Kruskal's Algorithm

Subset Systems

Matroids

Matroid Definition and Theorem

Definition

A matroid is a subset system $M = (E, \mathcal{I})$ that satisfies the **exchange property**: if $i, i' \in \mathcal{I}$ such that $|i| < |i'|$, then there exists $e \in i' - i$ with $i + e \in \mathcal{I}$

Matroid Definition and Theorem

Definition

A matroid is a subset system $M = (E, \mathcal{I})$ that satisfies the **exchange property**: if $i, i' \in \mathcal{I}$ such that $|i| < |i'|$, then there exists $e \in i' - i$ with $i + e \in \mathcal{I}$

Theorem

For any subset system (E, \mathcal{I}) , the greedy algorithm solves the optimization problem for (E, \mathcal{I}) if and only if (E, \mathcal{I}) is a matroid.

Matroid Definition and Theorem

Definition

A matroid is a subset system $M = (E, \mathcal{I})$ that satisfies the **exchange property**: if $i, i' \in \mathcal{I}$ such that $|i| < |i'|$, then there exists $e \in i' - i$ with $i + e \in \mathcal{I}$

Theorem

For any subset system (E, \mathcal{I}) , the greedy algorithm solves the optimization problem for (E, \mathcal{I}) if and only if (E, \mathcal{I}) is a matroid.

Definition

$i \in \mathcal{I}$ is **maximal** if there doesn't exist e with $i + e \in \mathcal{I}$. It is **maximum** if there doesn't exist $j \in \mathcal{I}$ with $w(j) > w(i)$.

Matroid Definition and Theorem

Definition

A matroid is a subset system $M = (E, \mathcal{I})$ that satisfies the **exchange property**: if $i, i' \in \mathcal{I}$ such that $|i| < |i'|$, then there exists $e \in i' - i$ with $i + e \in \mathcal{I}$

Theorem

For any subset system (E, \mathcal{I}) , the greedy algorithm solves the optimization problem for (E, \mathcal{I}) if and only if (E, \mathcal{I}) is a matroid.

Definition

$i \in \mathcal{I}$ is **maximal** if there doesn't exist e with $i + e \in \mathcal{I}$. It is **maximum** if there doesn't exist $j \in \mathcal{I}$ with $w(j) > w(i)$.

Note that the greedy algorithm produces a solution that is always maximal

Matroid Definition and Theorem

Definition

A matroid is a subset system $M = (E, \mathcal{I})$ that satisfies the **exchange property**: if $i, i' \in \mathcal{I}$ such that $|i| < |i'|$, then there exists $e \in i' - i$ with $i + e \in \mathcal{I}$

Theorem

For any subset system (E, \mathcal{I}) , the greedy algorithm solves the optimization problem for (E, \mathcal{I}) if and only if (E, \mathcal{I}) is a matroid.

Definition

$i \in \mathcal{I}$ is **maximal** if there doesn't exist e with $i + e \in \mathcal{I}$. It is **maximum** if there doesn't exist $j \in \mathcal{I}$ with $w(j) > w(i)$.

Note that the greedy algorithm produces a solution that is always maximal and a maximum solution is always maximal.

For Next Time...

- ▶ Read sections 3.1, 3.2, 3.3, and 3.4.
- ▶ (And homework will be posted this evening.)