1. Basic Definitions and Examples

1.8. The greedy algorithm

Kyeongin Hong

Based on *Matroid Theory* by James Oxley

January 19, 2026

Minimum-Weight Spanning Tree

Let G be a connected graph and let w be a function from $E(G)$ into $\mathbb{R}$. We call w a *weight function* on G, and for all $X \subseteq E(G)$ we define the weight $w(X)$ of X to be

$$w(X) = \sum_{x \in X} w(x).$$

For the given G and w, the problem is to find a spanning tree of G of minimum weight.

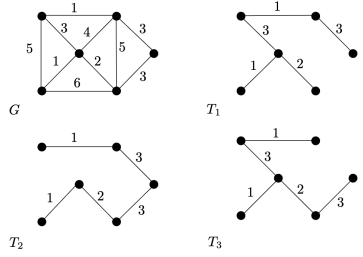


FIG. 1.32. G and its minimum-weight spanning trees.

Kruskal's algorithm

Kruskal's algorithm (1956) for solving this problem chooses edges one at a time. At each stage, the next edge e to be chosen is one of minimum weight such that

- e has not been previously chosen; and
- e does not form a cycle with the previously chosen edges.

When no further edges can be chosen subject to these restrictions, the algorithm stops. At this stage, the set of chosen edges is the edge set of a spanning tree of minimum weight.

Formulation

Let E be finite and $\mathcal{I} \subseteq 2^E$ satisfy

(I1) $\varnothing \in \mathcal{I}$.

(I2) If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$.

For a weight function $w : E \to \mathbb{R}$, define

$$w(X) = \sum_{x \in X} w(x),$$

and let $w(\varnothing) = 0$.

The optimization problem for the pair $(\mathcal{I}, w)$ is to find a maximal member B of $\mathcal{I}$ of maximum weight.

Greedy Algorithm

The greedy algorithm for the pair $(\mathcal{I}, w)$ proceeds as follows:

(i) Set $X_0 = \varnothing$ and $j = 0$.

(ii) If $E - X_j$ contains an element e such that $X_j \cup \{e\} \in \mathcal{I}$, choose such an element e_{j+1} of maximum weight, let

$$X_{j+1} = X_j \cup \{e_{j+1}\},$$

and go to (iii); otherwise, let $B_G = X_j$ and go to (iv).

(iii) Increase j by 1 and go to (ii).

(iv) Stop.

Lemma 1.8.3

If $(E, \mathcal{I})$ is a matroid M, then B_G is a solution to the optimization problem on the preceding slide.

Greedy Algorithm

Proof If $r(M) = r$, then $B_G = \{e_1, e_2, \ldots, e_r\}$ and B_G is a basis of M. Let B be another basis of M, say $B = \{f_1, f_2, \ldots, f_r\}$, where the elements are ordered so that $w(f_1) \geq w(f_2) \geq \cdots \geq w(f_r)$. We claim that $w(e_j) \geq w(f_j)$ for all $1 \leq j \leq r$.

Suppose, to the contrary, that this is not the case, and let k be the smallest integer such that

$$w(e_k) < w(f_k).$$

Let $I_1 = \{e_1, e_2, \ldots, e_{k-1}\}$ and $I_2 = \{f_1, f_2, \ldots, f_k\}$. Then $|I_1| < |I_2|$, so by the augmentation axiom (I3), we can extend I_1 by an element of I_2: there exists $f_t \in I_2 \setminus I_1$ such that $I_1 \cup \{f_t\} \in \mathcal{I}$. Since $w(f_t) \geq w(f_k) > w(e_k)$, the greedy algorithm would have chosen f_t instead of e_k. This contradiction completes the proof. □

Greedy Algorithm

The greedy algorithm works for every weight function on a matroid. Conversely, for a non-matroid satisfying (I1) and (I2), some weight function makes it fail.

Theorem 1.8.5

Let $\mathcal{I}$ be a collection of subsets of a set E. Then $(E, \mathcal{I})$ is a matroid if and only if $\mathcal{I}$ has the following properties:

(I1) $\varnothing \in \mathcal{I}$.

(I2) If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$.

(G) For all weight functions $w : E \to \mathbb{R}$, the greedy algorithm produces a maximal member of $\mathcal{I}$ of maximum weight.

Greedy Algorithm

Proof If $(E, \mathcal{I})$ is a matroid, then (I1) and (I2) certainly hold; by Lemma 1.8.3, so too does (G). Now suppose that $\mathcal{I}$ satisfies (I1), (I2), and (G). We want to show that $\mathcal{I}$ satisfies (I3). Assume the contrary, that is, suppose that I_1, I_2 are members of $\mathcal{I}$ with $|I_1| < |I_2|$ such that $I_1 \cup \{e\} \notin \mathcal{I}$ for all $e \in I_2 - I_1$.

Now $|I_1 - I_2| < |I_2 - I_1|$, so we can choose a real number ε such that

$$\frac{|I_1 - I_2|}{|I_2 - I_1|} < \varepsilon < 1.$$

Greedy Algorithm

Define $w : E \to \mathbb{R}$ by

$$w(e) = \begin{cases} 1, & \text{if } e \in I_1, \\ \varepsilon, & \text{if } e \in I_2 - I_1, \\ 0, & \text{otherwise.} \end{cases}$$

Then the greedy algorithm will first pick all the elements of I_1. By assumption, it cannot then pick any element of $I_2 - I_1$. Thus the remaining elements of B_G will be in $E - (I_1 \cup I_2)$. Hence

$$w(B_G) = |I_1|.$$

Since E is finite, the set I_2 is contained in a maximal member I_2' of $\mathcal{I}$, and

$$w(I_2') \geq w(I_2) = |I_1 \cap I_2| + |I_2 - I_1|\,\varepsilon > |I_1 \cap I_2| + |I_2 - I_1|\frac{|I_1 - I_2|}{|I_2 - I_1|} = |I_1|.$$

We deduce that $w(I_2') > w(B_G)$, that is, the greedy algorithm fails for this weight function. This contradiction completes the proof of the theorem. □

Job Assignment Problem

Let S be the set of jobs and Y the set of workers. For each $y \in Y$, let A_y be the set of jobs that worker y is qualified to perform, and let $\mathcal{A} = (A_y : y \in Y)$.

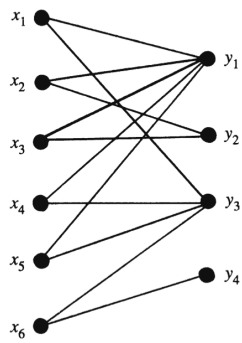
Evidently, the maximum number of jobs that can be done simultaneously is the size of a largest partial transversal of $\mathcal{A}$, or equivalently, the rank of the matroid $M[\mathcal{A}]$.

Now let $p : S \to \mathbb{R}$ be a function that gives the priority order on the jobs, where a lower p-value corresponds to a higher priority. We identify a maximum-size job assignment with its set of jobs, a basis of $M[\mathcal{A}]$.

Suppose that $B = \{x_1, x_2, \ldots, x_r\}$ is a job assignment with $p(x_1) \leq p(x_2) \leq \cdots \leq p(x_r)$. We call B *optimal* if, for any other job assignment $\{z_1, z_2, \ldots, z_r\}$ with $p(z_1) \leq p(z_2) \leq \cdots \leq p(z_r)$, we have $p(x_i) \leq p(z_i)$ for all $i \in \{1, 2, \ldots, r\}$.

Job Assignment Problem

If we define $\mathcal{I}$ to be the collection of independent sets of $M[\mathcal{A}]$ and let $w = -p$, then by Lemma 1.8.3 it follows that the greedy algorithm applied to the pair $(\mathcal{I}, w)$ will find an optimal job assignment.



Job Assignment Problem

We remark that, in this job assignment problem, the greedy algorithm finds an optimal basis of the corresponding transversal matroid without backtracking. However, assigning workers as jobs are added may require changing earlier worker–job assignments.

In general, once an optimal basis B has been found, the problem of assigning workers to the jobs in B is precisely the problem of finding a maximum-sized matching in the bipartite graph joining each worker y to the jobs in A_y, after deleting the vertices in $S - B$.