Thus the greedy strategy does not always yield the optimal solution to 0-1 knapsack.

Coin Changing Problem
As before, denominations \( d = (d_1, \ldots, d_n) \) and an amount \( N \) to be dispensed with the fewest coins. (Assume an unlimited supply of coins in each denomination.)

The greedy strategy to coin changing is as follows:

- From amongst all the denominations whose addition would not cause the sum to exceed \( N \), choose the largest.
- Stop when sum is \( N \).

Exercise (Hard)
Show that for \( d = (1, 5, 10, 25, 100) \) the greedy strategy yields an optimal solution for any \( N \geq 0 \).

Exercise (Easy)
Show that for \( d = (1, 10, 25, 100) \) the greedy strategy does not yield an optimal solution for some \( N \). (e.g., \( N = 30 \).)
Exercise (Hard)
Characterize all denomination sets \( d = \{d_1, \ldots, d_n\} \) such that the greedy strategy yields an optimal solution for all \( N \geq 0 \).

How can we tell if a greedy algorithm will solve a particular optimization problem? In general, this is a difficult question. The key ingredients to look for are:

- **Optimal Substructure**: All optimal solutions contain optimal subproblem solutions.

- **Greedy Choice Property**: A globally optimal solution can be obtained by making locally optimal (greedy) choices (with respect to some selection function).

Much of the hard work in designing a greedy algorithm is in proving that the greedy choice property is satisfied.

The situation is complicated by the fact that the selection function may not coincide with the objective function.
MINIMUM WEIGHT SPANNING TREE

We recall several definitions related to an (undirected) graph \( G = (V, E) \).

- \( G \) is called **connected** if \( G \) contains a \( u-v \) path for all \( u, v \in V \).

- A **cycle** in \( G \) is a closed path in \( G \), i.e., a path whose initial and terminal vertices are identical.

- \( G \) is called **acyclic** if it contains no cycles.

- A graph which is connected and acyclic is called a **tree**.

**Ex.**

![Graph](attachment:image.png)

\[ |V| = 8, \quad |E| = 7 \]
**Theorem**

The following are equivalent.

(1) \( G \) is a tree

(2) \( G \) is connected and \( |E| = |V| - 1 \)

(3) \( G \) is acyclic and \( |E| = |V| - 1 \)

(4) \( G \) is acyclic, but if a new edge is added, a unique cycle is created.

(5) \( G \) is connected, but the removal of any edge disconnects \( G \).

(6) There is a unique \( u-v \) path in \( G \) for all \( u, v \in V \).

For a proof see Appendix B, p. 1085, or take CSE 177.

- A subgraph of \( G \) is called a **spanning tree** if it is a tree, and it includes all vertices of \( G \).

**Ex:**
Suppose $G = (V, E)$ is equipped with a non-negative weight function on edges $W: E \rightarrow \mathbb{R}_+$. 

Example:

- The weight of a spanning tree is the sum of the weights of its edges.

**Problem**

Determine a spanning tree of minimum weight in $G$.

We examine two famous greedy algorithms which solve this problem.

In what follows let $G = (V, E), |V| = n$. 
Prim's Algorithm (23.2)

- Choose an initial vertex (which is a tree).
- Amongst all edges incident with the current tree whose addition would not create a cycle, choose one of minimum weight.
- Stop when n-1 edges have been selected.

Ex

\[ W(T) = 18 \]

Observe that at each stage of execution, Prim's algorithm maintains a tree since no cycles are created and only incident edges are added.

When this tree contains n-1 edges it must have n vertices (by previous theorem), hence it is a spanning tree.

Theorem

This spanning tree has minimum possible weight.

(See book or TURC CE 177 for proof.)
**Kruskal's Algorithm** (23.2)

- Choose an edge of minimum weight.
- Amongst all edges which do not create a cycle with previously selected edges, choose one of minimum weight.
- Stop when \( n - 1 \) edges have been selected.

*Ex.*

![Diagram of a graph](attachment:image.png)

\[ W(T) = 18 \]

Observe that at each stage of execution, Kruskal's algorithm has created a forest (union of disjoint subforests) since no cycle is created.

When this forest contains \( n - 1 \) edges it must also have \( n \) vertices. (Any graph with \( n - 1 \) edges has at least \( n \) vertices. This forest can contain no more than \( n \) vertices since it is a subgraph of \( G \).)

Thus the resulting forest is connected (by previous theorem) and is a spanning tree in \( G \).
**Theorem**

This spanning tree has minimum weight among all spanning trees in G.

**Proof.**

Let T be the spanning tree in G created by Kruskal's algorithm, and let S be any other spanning tree. We must show

\[ w(T) \leq w(S) \]

Let \( e_1, e_2, \ldots, e_{n-1} \) be the edges of T in the order selected by Kruskal's algorithm. Since \( S \neq T \) there is a first edge \( e_k \) which is not in S, i.e.

\[ \{e_1, \ldots, e_{k-1}\} \subseteq E(S) \]
\[ e_k \notin E(S) \]

Let \( H \) be the subgraph obtained by adding \( e_k \) to \( S \): \( H = S + e_k \). By the uniqueness of T, H contains a unique cycle which includes \( e_k \), call it \( C \). Note C must contain an edge e of S which is not in T, for otherwise C is contained in the acyclic T.