Greedy Algorithms

1. Minimum Spanning Tree
2. Huffman Coding
3. Horn Formulas
4. Set Cover

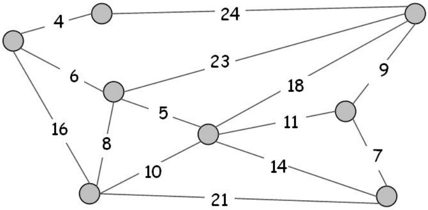
Greedy Approach

Idea. Greedy algorithms build up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit

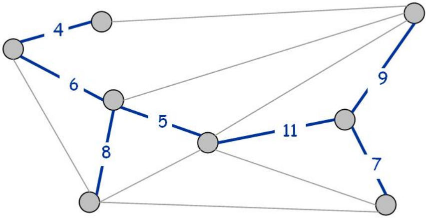
Minimum Spanning Tree

Definition

Minimum Spanning Tree (MST). Given a connected graph $G = (V, E)$ with real-valued edge weights c_e, an MST is a subset of the edges $T \subseteq E$ such that T is a spanning tree whose sum of edge weights is minimized



G = (V, E)



$T, \ \Sigma_{e \in T} c_e = 50$

from Wayne's slides on "Algorithm Design"

Greedy Algorithms

1. **Kruskal's algorithm**
   Start with $T = \emptyset$. Consider edges in ascending order of cost. Insert edge e in T unless doing so would create a cycle
2. **Reverse-Delete algorithm**
   Start with $T = E$. Consider edges in descending order of cost. Delete edge e from T unless doing so would disconnect T
3. **Prim's algorithm**
   Start with some root node s and greedily grow a tree T from s outward. At each step, add the cheapest edge e to T that has exactly one endpoint in T

Kruskal's Algorithm

Figure 5.1 The minimum spanning tree found by Kruskal's algorithm.

Kruskal's Algorithm

Figure 5.1 The minimum spanning tree found by Kruskal's algorithm.

1. `makeset(x)`: create a singleton set containing just x
2. `find(x)`: to which set does x belong?
3. `union(x, y)`: merge the set containing x and y

Kruskal's Algorithm

```
procedure kruskal (G, w)
Input:  A connected undirected graph G = (V, E) with edge weights w_e
output: A minimum spanning three defined by the edges X

for all u ∈ V:
   makeset (u)

X = {}
sort the edges E by weight
for all edges {u, v} ∈ E, in increasing order of weight:
    if find(u) ≠ find(v):
        add edge {u, v} to X
        union(u, v)
```

Running time $= |V|$ makeset $+ 2 \cdot |E|$ find $+ (|V| - 1)$ union

Correctness of Greedy Algorithm

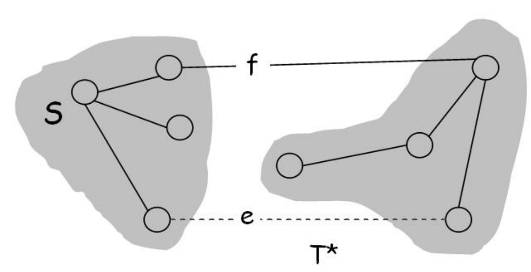
Definition

Cut. A *cut* is any partition of the vertices into two groups, S and $V - S$

Lemma

Let S be any subset of nodes, and let e be the min-*cost edge with exactly one endpoint in S. Then the MST contains e*

Proof.



□

from Wayne's slides on "Algorithm Design"

Correctness of Greedy Algorithm

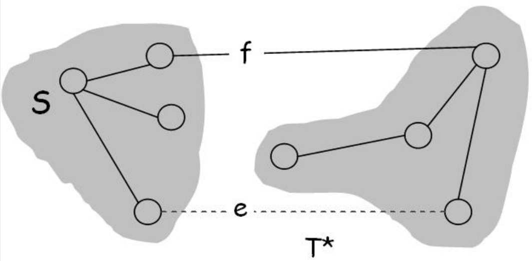
Definition

Cycle. Set of edges the form $(a, b), (b, c), (c, d), \dots, (y, z), (z, a)$

Lemma

Let C be any cycle in G, and let f be the max cost edge belonging to C. Then the MST does not contain f

Proof.



□

from Wayne's slides on "Algorithm Design"

Prim's Algorithm

1. Initialize S = any node
2. Apply cut property to S
3. Add min-cost edge in cut-set corresponding to S to T, and add one new explored node u to S

Figure 5.1 The minimum spanning tree found by Kruskal's algorithm.

Huffman Coding

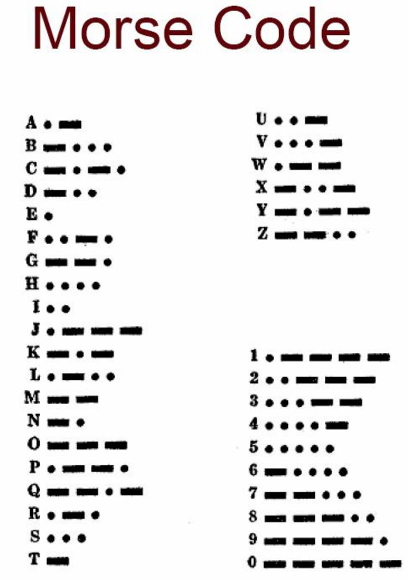


Image adapted from Wikipedia.

Huffman Coding

Definition

Prefix-free. No codeword can be a prefix of another codeword

$\{0,\ 01,\ 11,\ 001\}$?

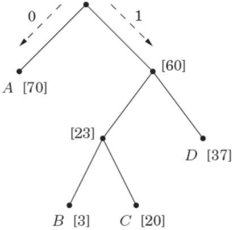
Remark

Any prefix-free encoding can be represented by a full *binary tree.*

$\{0,\ 100,\ 101,\ 11\}$?

Figure 5.10 A prefix-free encoding. Frequencies are shown in square brackets

Symbol	Codeword
A	0
B	100
C	101
D	11



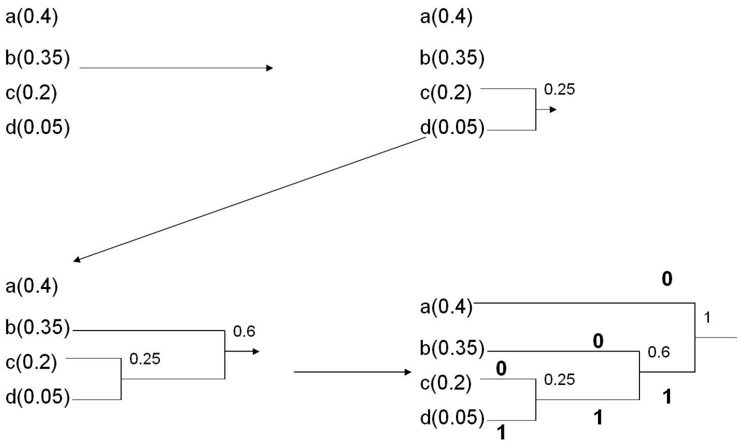
$$\text{cost of tree} = \sum_{i=1}^{n} f_i \cdot (\text{depth of the } i\text{th symbol in tree})$$

Huffman Coding

```
procedure Huffman(f)
Input:  An array f[1···n] of frequencies
Output: An encoding tree with n leaves

let H be a priority queue of integers, ordered by f
for i = 1 to n: insert(H, i)
for k = n+1 to 2n−1:
   i = deletemin(H), j = deletemin(H)
   create a node numbered k with children i, j
   f[k] = f[i] + f[j]
   insert(H, k)
```

Huffman Coding



http://rio.ecs.umass.edu/ gao/ece665_08/slides/Rance.ppt

Horn Formula

The most primitive object in a Horn formula is a *Boolean variable*, taking value either `true` or `false`

A *literal* is either a variable x or its negation $\bar{x}$

There are two kinds of *clauses* in Horn's formulas

1. *Implications*

$$(z \wedge w) \Rightarrow u$$

2. Pure *negative clauses*

$$\bar{u} \vee \bar{v} \vee \bar{y}$$

Questions. To determine whether there is a consistent explanation: an assignment of `true/false` values to the variables that satisfies all the clauses

Satisfying Assignment

Input: A Horn formula
Output: A satisfying assignment, if one exists

```
function horn

    set all variables to false;

    while (there is an implication that is not satisfied)
        set the right-hand variable of the implication to true;

    if (all pure negative clauses are satisfied)
        return the assignment;
    else
        return ``formula is not satisfiable'';
```

$$(w \wedge y \wedge z) \Rightarrow x, (x \wedge z) \Rightarrow w, x \Rightarrow y, \Rightarrow x, (x \wedge y) \Rightarrow w, (\bar{w} \vee \bar{x} \vee \bar{y}), \bar{z}$$