Today

- Prim's Algorithm
- Huffman Encoding

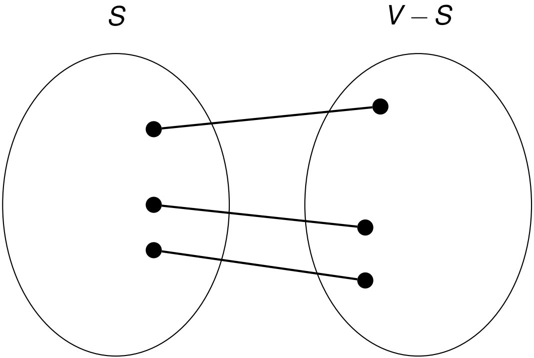
What is a Cut?

What is a cut in an undirected graph, $G = (V, E)$?

(A) A set of edges whose removal disconnects a graph.

(B) For partition of V, $(S, V - S)$, set of edges across it; $E \cap (S \times V - S)$.

(A) and (B).



Note:sometimes specified as $(S, V - S)$

..... sometimes explicitly as subset of edges E'.

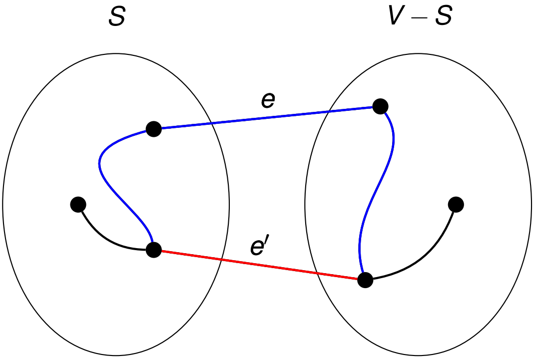
What is the cut property for MSTs?

(A) Any edge in a cut is in some mst.

(B) The smallest edge in a cut is in some mst.

(C) The largest edge in a cut cannot be in an mst.

B.



Cut Property: Any edge of minimal weight in a cut is in some MST. (If unique, it must be in MST.)
Proof: replace, $n-1$ edges still connected and as cheap.

Prim's algorithm

Cut Property: Any edge of minimal weight in a cut is in some MST. (If unique, it must be in MST.)

Generic MST:
While not connected.
  Add smallest edge across a cut.

Correctness: cut property with unique weight edges.

Break ties for smallest edge according to lowest neighbors.

Prim's Algorithm.

Generic MST:
Start with $S = v$:
  Find smallest edge (x, y) in crossing $(S, V - S)$
    let $S = S \cup \{y\}$

Implementation?

Find smallest edge (x, y)? $O(m)$ time.

$O(nm)$ time.

Look at same edges over and over again!

Use a priority queue to keep edges!

Actually... use a priority queue to keep "closest" node.

Prim's Algorithm

Prim(G,s)
  foreach $v \in V$: $c(v) = \infty, prev(v) = nil$
  c(s) = 0, prev(s) = s

  H = make_pqueue(V,c)
  while (v = deletemin(H)):
    Add $(v, prev(v))$ to T
    foreach (v,w):
      if $(w(v,w) < c(w))$:
        $c(w) = w(v,w), prev(w) = v$
        decreaseKey(H,w)

Dijkstras:

      if $(c(v)+w(v,w) < c(w))$:
        $c(w) = c(v) + w(v,w), prev(w) = v$

Runtime? $\Theta(mn)$? $\Theta((m+n)\log n)$? $\Theta(m+n\log n)$? $O((m+n)\log n)$

With Fibonacci Heaps: $O(m+n\log n)$.

Compression.

Given a long file, make it shorter.

16 characters alphabet, four bits/character.
One Idea: shorter representation of frequent characters.

Morse code:

E . “dot”
T - “dash”
I .. “dot dot”
A .- “dot dash”
N -. “dash dot”
S ... “dot dot dot”
...

Common Characters are shorter...
Cool! Translate to 0 and 1? Sure.

What is **..** or “dot dot”?
“I” or “EE” ??

Separate using pauses in morse code.. for binary?

Prefix free codes.

A code is *prefix free* (or prefix) if no code for a letter is a prefix of another.

Letters: A,B,C,D.

Codewords: strings in $\{0, 1\}^*$.

Example:

$$A : 00 \quad B : 01 \quad C : 10 \quad D : 11$$

Fixed length codewords.
No codeword is prefix of another.

What is 100011?
First two: "C"
Next two: "A"
Third two: "D"

Can decode!

Prefix freeness.

Another prefix free code for A,B,C,D.

$(A:0),(B:10),(C:110),(D:111)$

"110010" ???

C A B

DNA example.

Consists of letters A, C, T, G with varying frequencies.

A: .4
C: .1
T: .2
G: .3

Expected length of fixed length encoding for N chars: $2N$ bits.

A: .4	0
C: .1	100
T: .2	101
G: .3	11

Prefix Free?

0 not prefix of 100, 101, or 11.

11 not prefix of 0, 100 or 101

...

Yes!

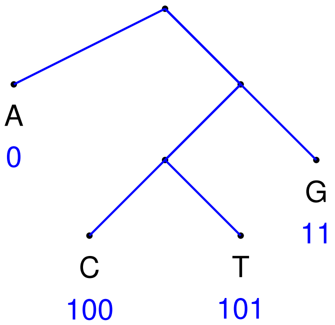
Expected length: N(.4*1 + .1*3 + .2 *3 + .3 * 2) = $1.9N$

Prefix free codes and binary trees.

Each prefix-free codes corresponds to a full binary trees:

- Each internal node has two children.
- Code words at the leaves.

Our Example:



011101100

0 11 101 100

A G C T

Huffman Coding.

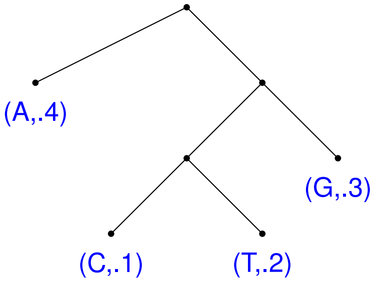
Given symbol frequencies $f_1, \ldots, f_n$, find "best" prefix code.

Smallest average length.

Cost of prefix tree with symbol leaves:

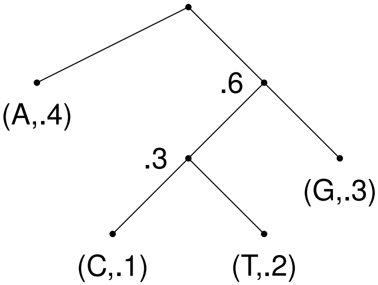
$$\sum_i f_i(\text{depth of symbol } i \text{ in tree.})$$

Example: (A,.4), (C,.1),(T,.2),(G,.3)



Cost: $.4 * 1 + .3 * 2 + .1 * 3 + .2 * 3 = 1.9$

Another view of cost.



Sum over all nodes, except root, of their frequency.

$.4 + .1 + .2 + .3 + .3 + .6 = 1.9$

Greedy Algorithm.

Recursive View: internal node has frequency
..."internal nodes" $\equiv$ "sort of symbols."

Idea: Merge two symbols to make new internal node (symbol).

Which ones?

Cost of prefix tree with symbol leaves:

$$\sum_i f_i(\text{depth of symbol } i \text{ in tree.})$$

Might as well merge two lowest frequency symbols...
to make low freq internal symbol.

Let's see method!

$(A, .4), (C, .1), (T, .2), (G, .3)$

$(A, .4), (\{C, T\}, .3), (G, .3)$

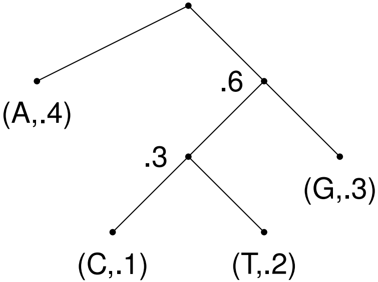
$(A, .4), (\{\{C, T\}, G\}, .6)$

$(\{A, \{\{C, T\}, G\}\}, 1)$

Algorithm.

Cost2: Sum over all nodes, except root, of their “frequency”.

Algorithm:
Make each symbol into single node tree.
While more than one tree:
    Merge two lowest frequency trees, into a new tree.



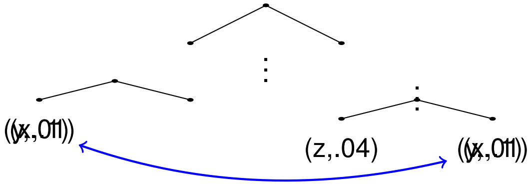
Implementation: priority queue to get lowest frequency trees.

Correctness..

Recall MST: added a “could have” edge in tree every time.
  “must have” if no ties.

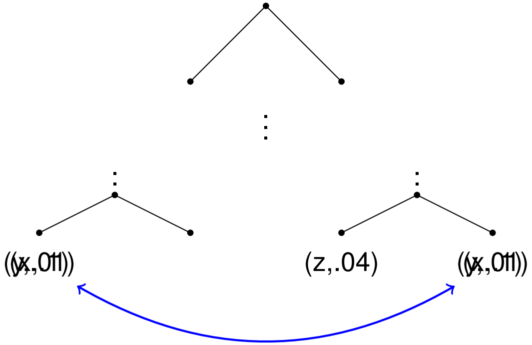
Huffman: An optimal tree
  “could have” subtree with lowest frequency symbols
  and optimal tree on new characters.

Consider optimal tree.
Consider lowest frequency two symbols (assume no ties).
If siblings $\rightarrow$ ok to merge in algorithm.
Otherwise ... switch each with deepest pair of siblings improves tree.



Cost gets better with this switch. Lowest frequency made siblings.

What about same depth?



Cost stays the same, but can make lowest into sibling.

Huffman codes wrap-up

Questions ??

Can the shape of a Huffman tree be a

1. Caterpillar? Yes / No ?
2. Full balanced binary tree on k levels? Yes/No?
3. Any shape of a full binary tree? Yes/No ?

Taint Analysis

$a = http.read\_response();$

$\vdots$

$b = a + c;$

$\vdots$

$d = sql\_command(b);$

a is input from web.

"a is tainted."

"if a is tainted b is tainted."

"b should not be tainted."

Logic representation:

A - "a is tainted"

B - "b is tainted"

$$\implies A\ ,\ A \implies B\ ,\ \overline{B}.$$

Satisfiable?

Not in this case.

Horn SAT

Implications:

And of positive literals imply **one** positive literal.

$$x \wedge y \rightarrow z$$

Negative clauses:

Or of negative literals.

$$\overline{u} \vee \overline{v}$$

Taint Example:

$$\implies A, A \implies B, \overline{B}.$$

Is this satisfiable?

Horn Sat: another view.

$$\begin{aligned} x_1 \wedge x_2 &\implies x_4 \\ x_3 &\implies x_2 \\ x_1 &\implies x_3 \\ x_5 \wedge x_1 &\implies x_3 \\ x_2 \wedge x_6 &\implies x_5 \\ &\implies x_1 \end{aligned}$$

Problem: Find consistent assignment with fewest "True" variables.

Greedy algorithm: Only set variables to true if you have to.

Example:
x_1 must be true so x_3 must be true
so x_2 must be true so x_4 must be true
Solution: $\{x_1, x_2, x_3, x_4\}$ are True

Could also set x_5 to true, or both x_5 and x_6 to true...but don't!

Same as horn sat!

Why same as HornSAT?

Horn SAT had negative clauses.

No negative clauses for above algorithm.

Algorithm: Set a variable true ..if you have to!

Property: any variable set to true must be true in *any* satisfying assignment.

By induction. First k set to true... must be!
The $k+1$ set variable set to true
    is set to true to satisfy a clause
        so it must be true.

Horn has negative clauses.

Negative clauses only problem for true variables.

Any variable that is true must be true.

So if a negative clause is false, it must be.

Efficient implementation

$$
\begin{aligned}
x_1 \wedge x_2 &\implies x_4 \\
x_3 &\implies x_2 \\
x_1 &\implies x_3 \\
x_5 \wedge x_1 &\implies x_3 \\
x_2 \wedge x_6 &\implies x_5 \\
&\implies x_1
\end{aligned}
$$

For each clause: keep count of true antecedents:

When all antecedents true, than make consequent true.

Data Structure:

Connect variable to clauses with var as antecendent.

When variable is set to true see if connected clauses are invoked.