Announcements

- Exam 2 on Friday
  - Please let me know if you cannot take during class time (and didn't already tell me for exam 1)
- No class on Monday

Last Time

- Greedy Algorithms
  - Find decision Procedure
  - Repeatedly make best available choice
  - Repeat until done
- Exchange arguments
- Optimal Caching

Today

- Huffman Codes
- Minimum Spanning Trees

Huffman Codes

- Want to encode string of letters in binary.

Ex: ABCDACBDAD

- A = 00, B = 01, C = 10, D = 11

A	B	C	D	A	C	B	D	A	D
00	01	10	11	00	10	01	11	00	11

- Use two bits to encode each letter.

Question: Encoding Length

Using the coding scheme from the last slide, how many bits are needed to encode a string of n `A`s, `B`s, `C`s and `D`s?

1) 2
2) n
3) 2n
4) 4n
5) n^2

You need two bits per letter.

Non-Fix Length Encodings

- Suppose instead we had to decode:

  AAABAACBAABADAAA

- 16 Letters requires 32 bits.
- Note that there are a lot of A's here. If we could find a way to encode them with fewer bits, we could save a lot.

Unique Decoding

Cannot do any encoding we like.

Suppose we tried:

`A = 0, B = 1, C = 10, D = 01`

How do you decode `01`? Either `AB` or `D`.

Problem: The encoding for `A` is a prefix of the encoding for `D`. When you see it, you don't know if it's an `A`, or the start of a `D`.

Prefix Free Encodings

Definition: An encoding is prefix-free if the encoding of no letter is a prefix of the encoding of any other.

Example:

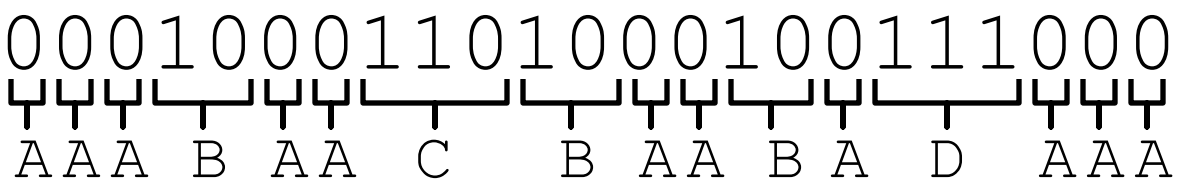
`A = 0, B = 10, C = 110, D = 111`

Lemma: Any prefix-free encoding can be uniquely decoded.

Example

A = 0, B = 10, C = 110, D = 111

Decode:



Only 23 bits instead of 32!

Optimal Encoding

Problem: Given a string, S, find a prefix-free encoding that encodes S using the fewest number of bits.

How Long is the Encoding?

If for each letter x in our string, x appears f(x) times and if we encode x as a string of length ℓ(x), the total encoding length is:

$$\Sigma\, f(x) \cdot \ell(x).$$

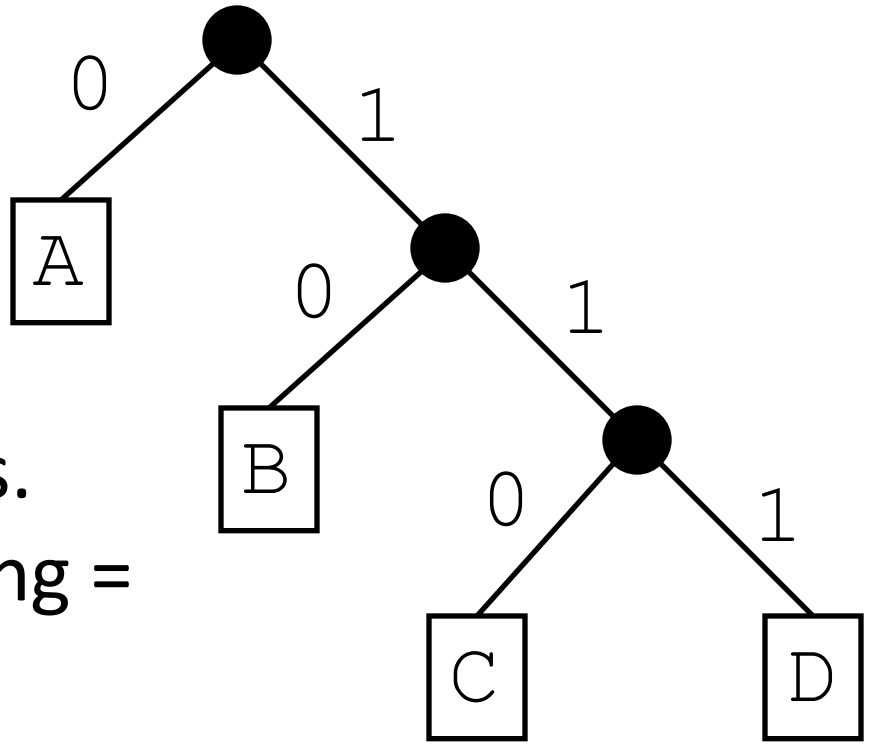
Our example has:

11 `A`s, 3 `B`s, 1 `C`, 1 `D`.

These are the frequencies. We need to find the best encoding.

Tree Representation

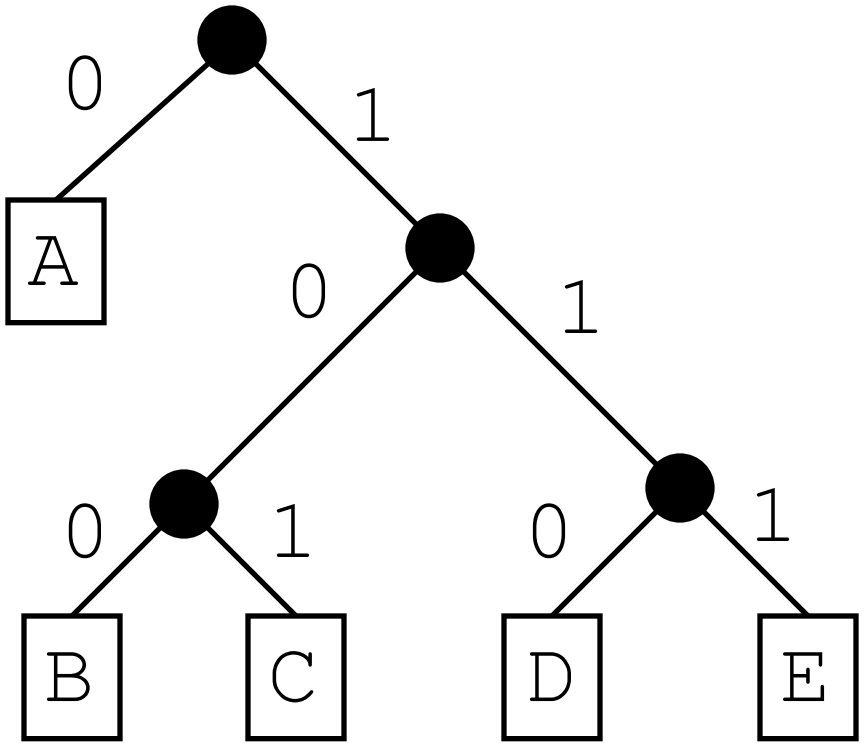
Can represent prefix-free encoding as a tree.



Letters are leaves.
Length of encoding =
Depth of leaf.

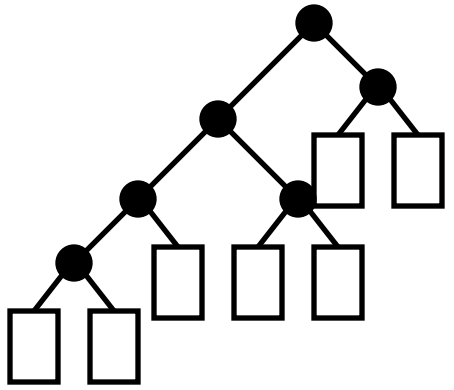
Question: Tree Decoding

What letter does the string `111` correspond to in this tree?



Placement of Leaves

Suppose we know the tree structure. Where do we put the letters?



Objective = Σ freq(x)depth(x)

Want least frequent letters at lowest depth.

Letter frequencies

```
Ax10, Bx15,
Cx4,  Dx22,
Ex31, Fx5,
Gx19
```

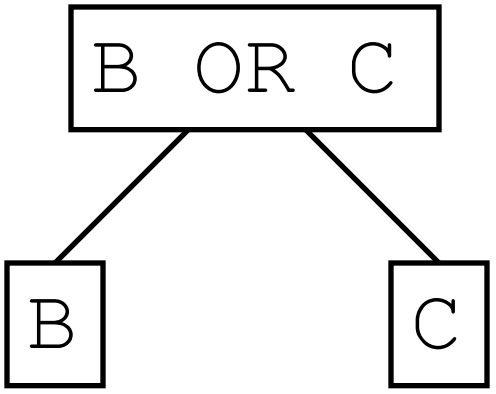
Siblings

- No matter what the tree structure, two of the deepest leaves are siblings.
- Can assume filled by two least frequent elements.
- Can assume that two least frequent elements are siblings!

Example

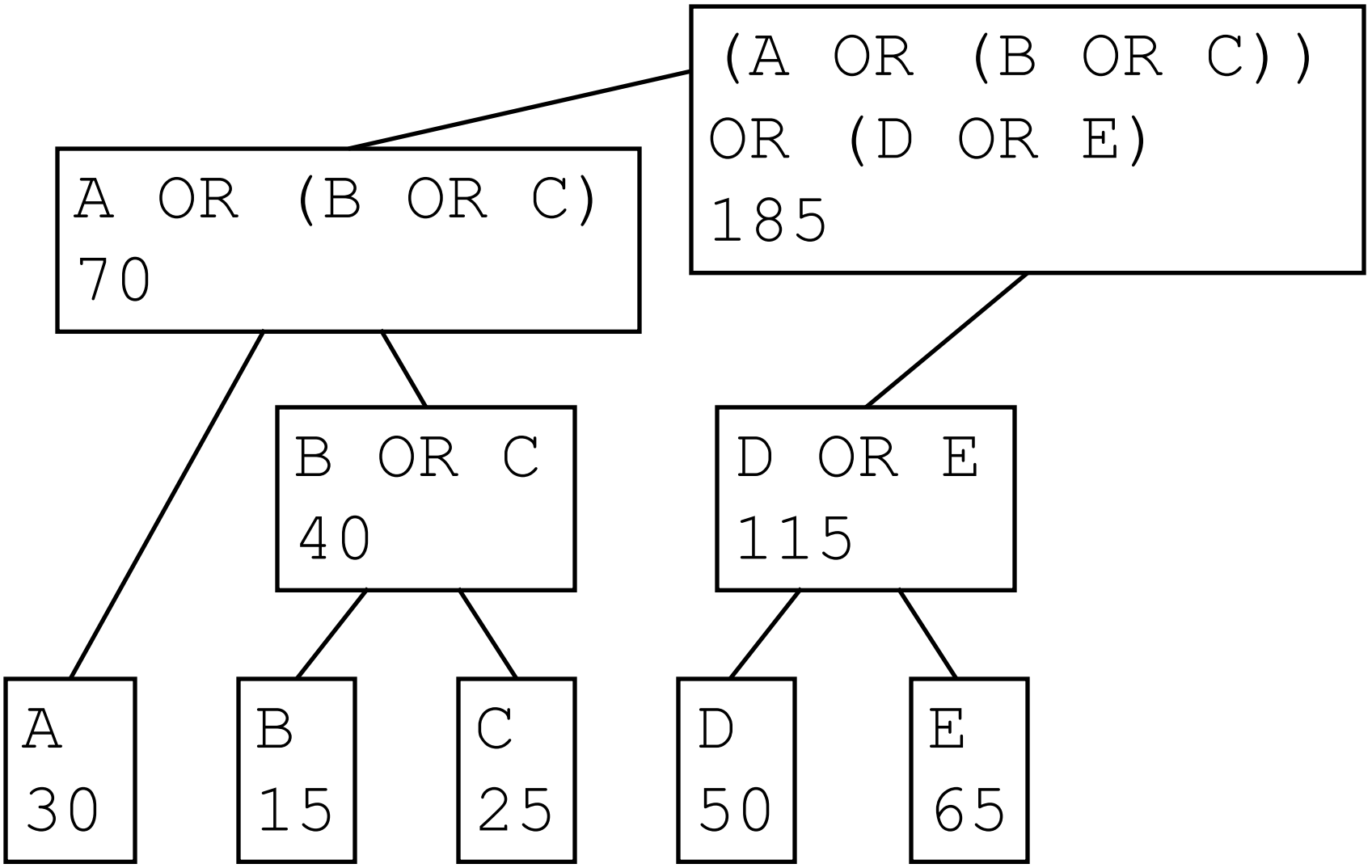
Frequencies:

`Ax30, Bx15, Cx25, Dx50, Ex65`



Think of as a new node of weight 15+25=40

Example


(A OR (B OR C))
OR (D OR E)
185
A OR (B OR C)
70
B OR C
40
D OR E
115
A
30
B
15
C
25
D
50
E
65

Algorithm

```
HaffmanTree(L)
  While(at least two left)
    x, y ← Two least frequent
    z new node f(z) ← f(x)+f(y)
    x and y children of z
    Replace x and y with z in L
  Return remaining elt of L
```

Better with priority queue.

Optimized Algorithm

```
HuffmanTree(L)
   Priority queue Q
   Insert all elements of L to Q
   While(|Q| ≥ 2)
      x ← Q.DeleteMin()
      y ← Q.DeleteMin()
      Create z, f(z) = f(x) + f(y)
      x and y children of z
      Q.Insert(z)
   Return Q.DeleteMin()
```

O(n log(n))

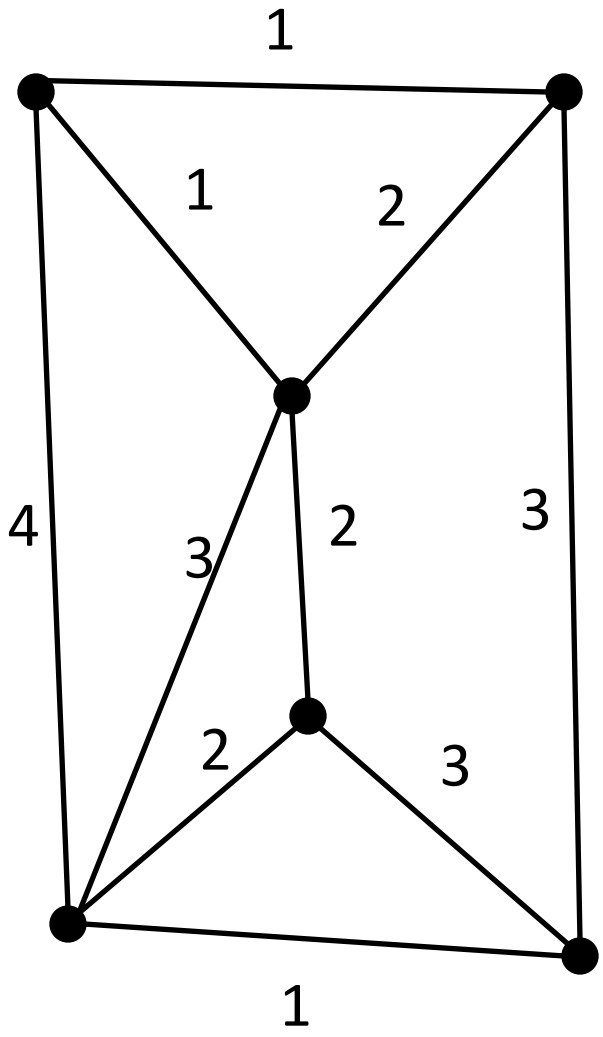
O(n) Iterations

O(log(n))

Runtime: O(n log(n))

Minimum Spanning Trees

- Suppose that you have a collection of cities that you would like to connect by roads.
- There are several potential roads you could build.
- Each has a cost.
- What is the cheapest way to connect them? 1+1+1+2+2=7



Trees

Note: In this problem, you will never want to build more roads than necessary. This means, you will never want to have a cycle.

Definition: A tree is a connected graph, with no cycles.

A spanning tree in a graph G, is a subset of the edges of G that connect all vertices and have no cycles.

If G has weights, a minimum spanning tree is a spanning tree whose total weight is as small as possible.

Question: MST

What is the weight of the minimum spanning tree of the graph below?

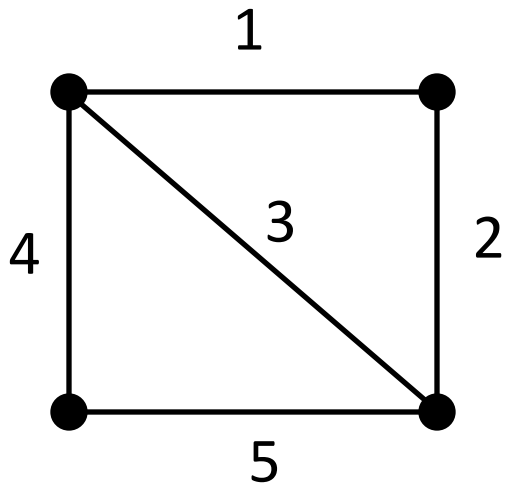
A) 5

B) 6

C) 7

D) 8

E) 9



Basic Facts about Trees

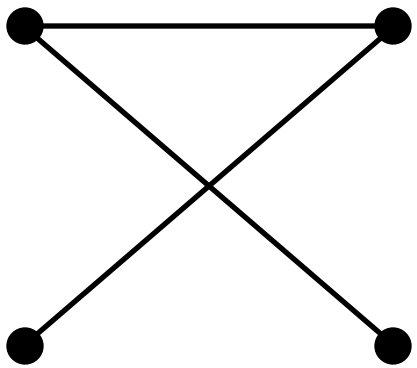
Lemma: For an undirected graph G, any two of the below imply the third:

1. $|E| = |V|-1$
2. G is connected
3. G has no cycles

Corollary: If G is a tree, then $|E| = |V|-1$.

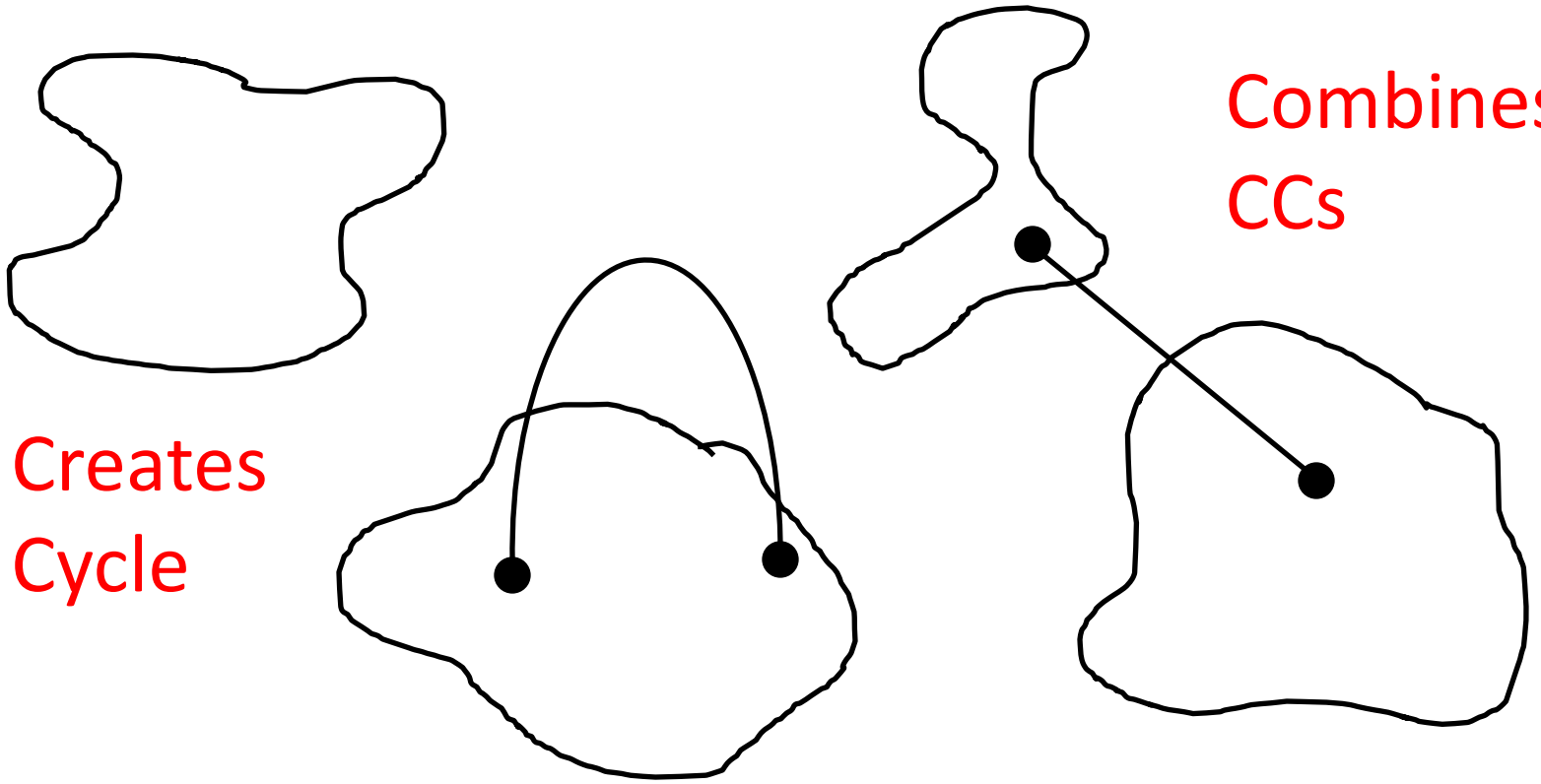
Proof Idea

- Start with a graph with no edges.
- Add edges one at a time.
- Count number of connected components.



Extra Edge

An extra edge decreases the number of CCs by 1 *unless* it creates a cycle.



Number of Components

- Starts with |V|.
- Each edge decreases by 1 unless a cycle.
- Final graph is connected if it reduces to 1.

If |E|=|V|-1 and no cycle, then only 1 CC left.

If |E|=|V|-1 and connected, each edge must decrease by 1, so no cycles.

If connected and no cycles, each edge decreases by 1, so must be |V|-1 edges.

Greedy Idea

How do you make an MST?

- Try using the cheapest edges.

Proposition: In a graph G, let e be an edge of lightest weight. Then there exists an MST of G containing e. Furthermore, if e is the unique lightest edge, then *all* MSTs contain e.

Proof Idea

- Suppose that we have an MST T that *does not* contain e.
- Modify T to get T’ that does contain e and has $wt(T') \leq wt(T)$.
- T’ will be a MST as well.
- Furthermore if e is the unique lightest edge, $wt(T') < wt(T)$, so T could not have been minimal.

Proof

- Consider tree T not containing e.
- With extra edge, no longer a tree, must contain a cycle.
- Remove edge e’ from cycle to get T’.
- |T’|=|V|-1, and connected, so T’ is a tree.
- wt(T’) = wt(T)+wt(e)-wt(e’)
  ≤ wt(T)

(because wt(e) is minimal).