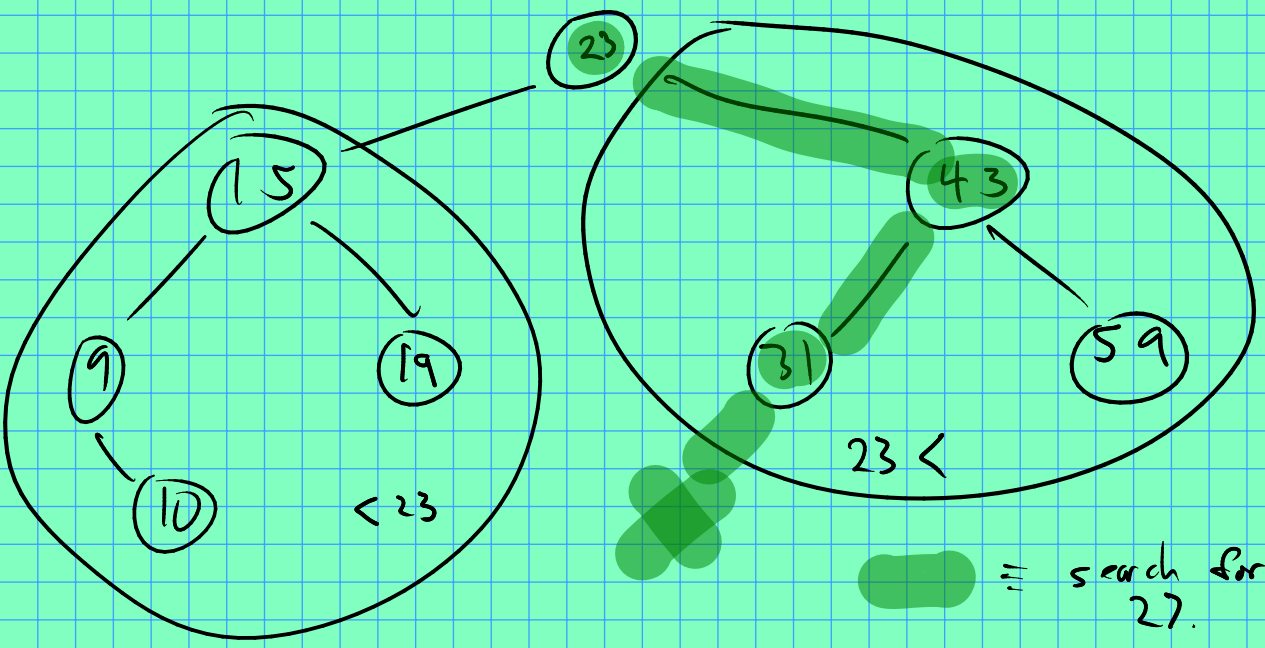
Sets & Maps

Implemented as binary search trees:



(Quick note on implementation: use pointers!)

Advantages over arrays/vectors:

Say tree has n values.

Time for a search is $\log_2 n$

$$1 = n/2^k$$ solve for k:

$2^k = n$ so $k = \log_2 n$.

$n/2^k \simeq$ # nodes left in subtree after k-steps (step ≡ comparison)

$\log_2 n$ is much faster than searching in a vector (n steps)

Disadvantages vs vectors:

Random access (setting the i^{th} element) takes $\approx i$ steps in a set, but ≈ 1 for a vector (use V[i])

Features they share:

easy to add new elements at will:

V.push_back(x) $\approx$ S.insert(x)

Note: they are modelling mathematical sets, so they don't store duplicates.

```
s.insert(23);
cout << s.size(); // prints 10
s.insert(23);
cout << s.size(); // still 10.
```

Aside: boolean functions as sets:

Say universe is U.

how does each function $f_S : U \to \{0,1\}$ define (uniquely) a subset $S \subseteq U$?

$$\{S \subseteq U\} \longleftrightarrow \{f_S : U \to \{0,1\}\}$$

The bijection is as follows:

For a set $S \subseteq U$, define its "characteristic

function" $f_S : U \to \{0,1\}$ by the rule

$$f_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{else} \end{cases}.$$

For a function $f : U \to \{0,1\}$ could recover the set as $S_f = f^{-1}(\{1\}) \subseteq U$

$$\nearrow$$

$$\{x \in U \mid f(x) \in \{1\}\}$$

(i.e, $f(x) = 1$)

$$f^{-1} : \mathcal{P}(Y) \to \mathcal{P}(X)$$

for $f : X \to Y$.