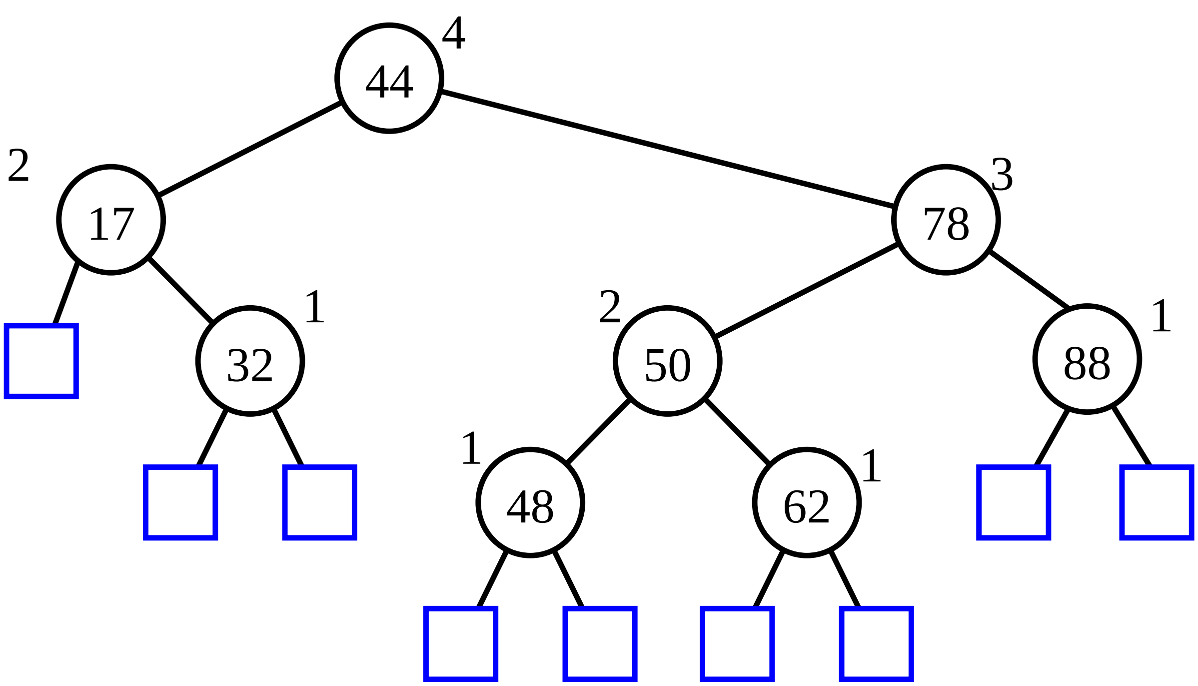
# SEARCHING

- the dictionary ADT
- binary search
- binary search trees

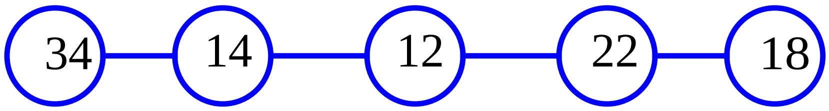


# The Dictionary ADT

- a dictionary is an abstract model of a database
- like a priority queue, a dictionary stores key-element pairs
- the main operation supported by a dictionary is searching by key
- simple container methods:
  - size()
  - isEmpty()
- query methods:
  - findElement(k)
  - findAllElements(k)
- update methods:
  - insertItem(k, e)
  - remove(k)
  - removeAll(k)
- special object
  - NO_SUCH_KEY, returned by an unsuccessful search

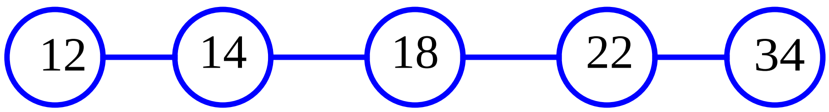
# Implementing a Dictionary with a Sequence

- unordered sequence:




  - searching takes O(n) time
  - inserting takes O(1) time

- ordered sequence

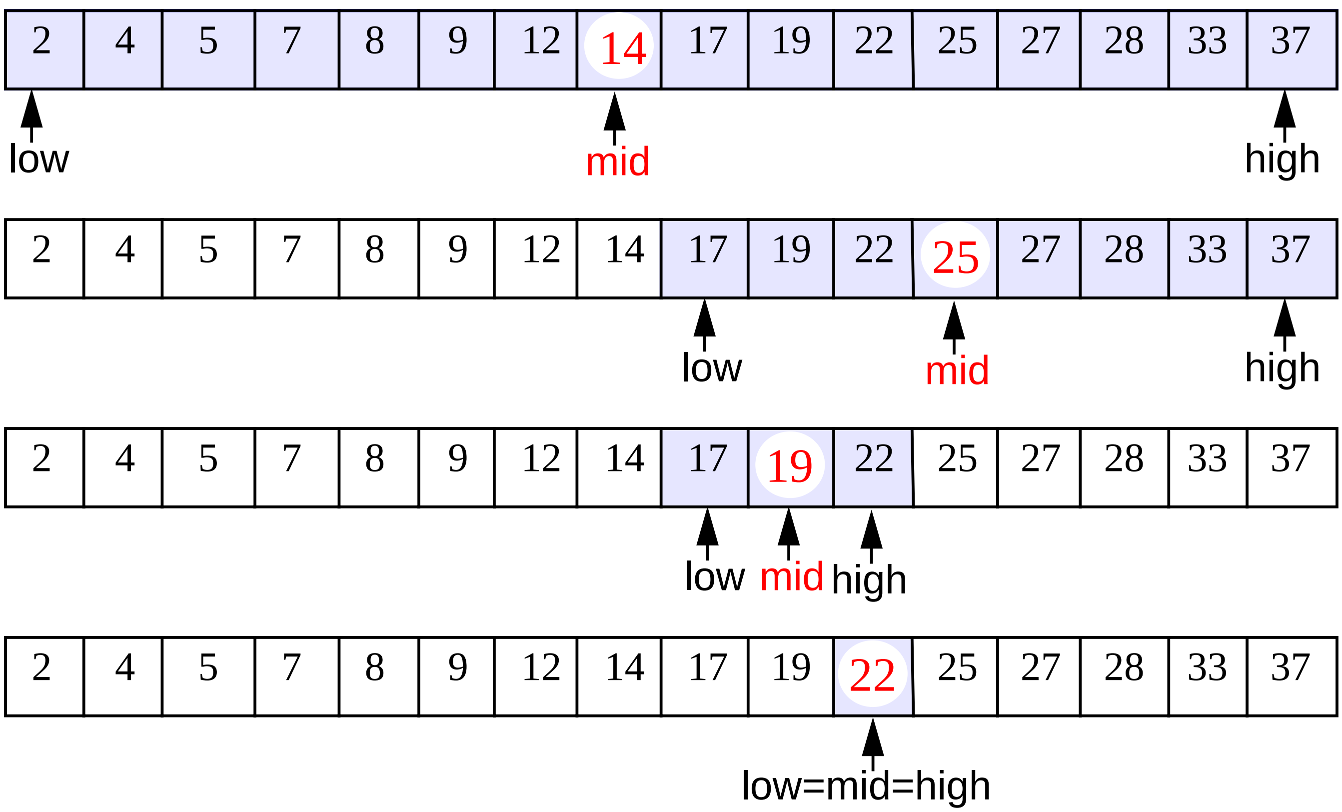



  - searching takes O(1) time
  - inserting takes O(n) time

- in the ordered sequence implementation, we can search faster if the sequence is array-based ...

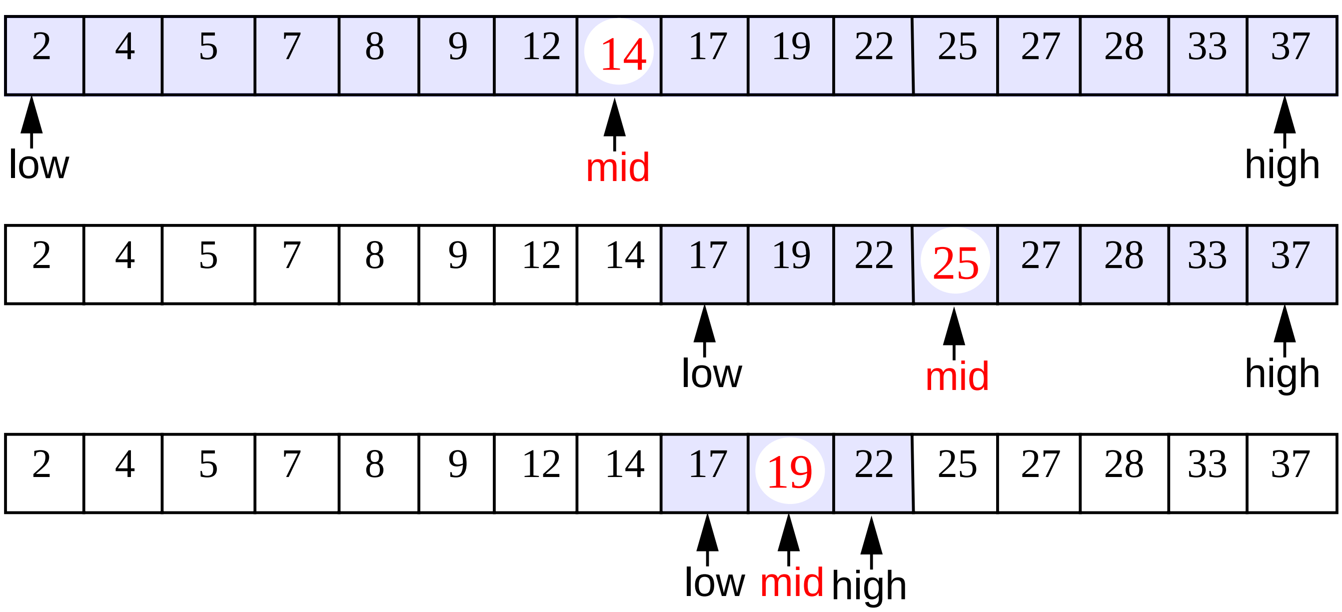
# Binary Search

- narrow down the search range in stages
- “high-low” game
- findElement(22)



# Pseudo-code for Binary Search

```
Algorithm BinarySearch(S, k, low, high)
if low > high then
   return NO_SUCH_KEY
else
   mid ← (low+high) / 2
   if k = key(mid) then
      return key(mid)
   else if k < key(mid) then
      return BinarySearch(S, k, low, mid–1)
   else
      return BinarySearch(S, k, mid+1, high)
```



# Running Time of Binary Search

- the range of candidate items to be searched is halved after comparing the key with the middle element

| comparison | search range |
|---|---|
| 0 | n |
| 1 | n/2 |
| 2 | n/4 |
| ... | ... |
| $2^i$ | $n/2^i$ |
| $\log_2 n$ | 1 |

- in the array-based implementation, access by rank takes O(1) time, thus binary search runs in O(log n) time