Chained Hash Table Expected Time

when key is not present:

expected work ≈ 1 + expected number of nodes examined

= 1 + n/m = 1 + α

when key is present:

expected work ≈ sum over keys of P(search for that key) * (1 + expected nodes examined searching for key)

$$= \sum_{i=1}^{n} \frac{1}{n} * \left(1 + \left(1 + \frac{i-1}{m}\right)\right) = \Theta(1 + \alpha)$$

P(2nd thing collided) = $\frac{1}{m}$

P(2nd didn't collide) = $\frac{m-1}{m}$

E[nodes to find 2nd] = $1 \cdot \frac{m-1}{m} + 2 \cdot \frac{1}{m}$

$= \frac{m+1}{m}$

$= 1 + \frac{1}{m}$

Open Addressing Expected Time

when key is not present, uniform hashing

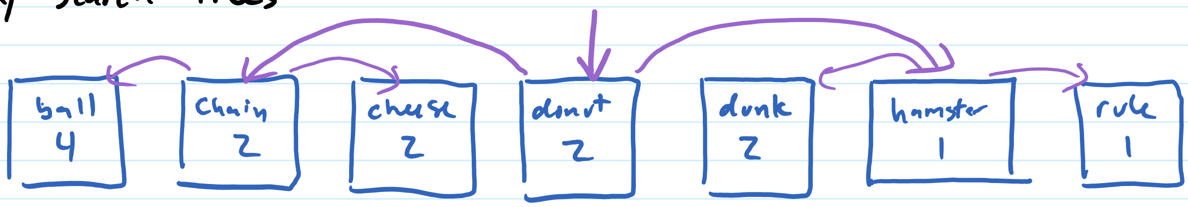
$$1 + \frac{1}{1 - \alpha}$$

Map/Set Time Complexity

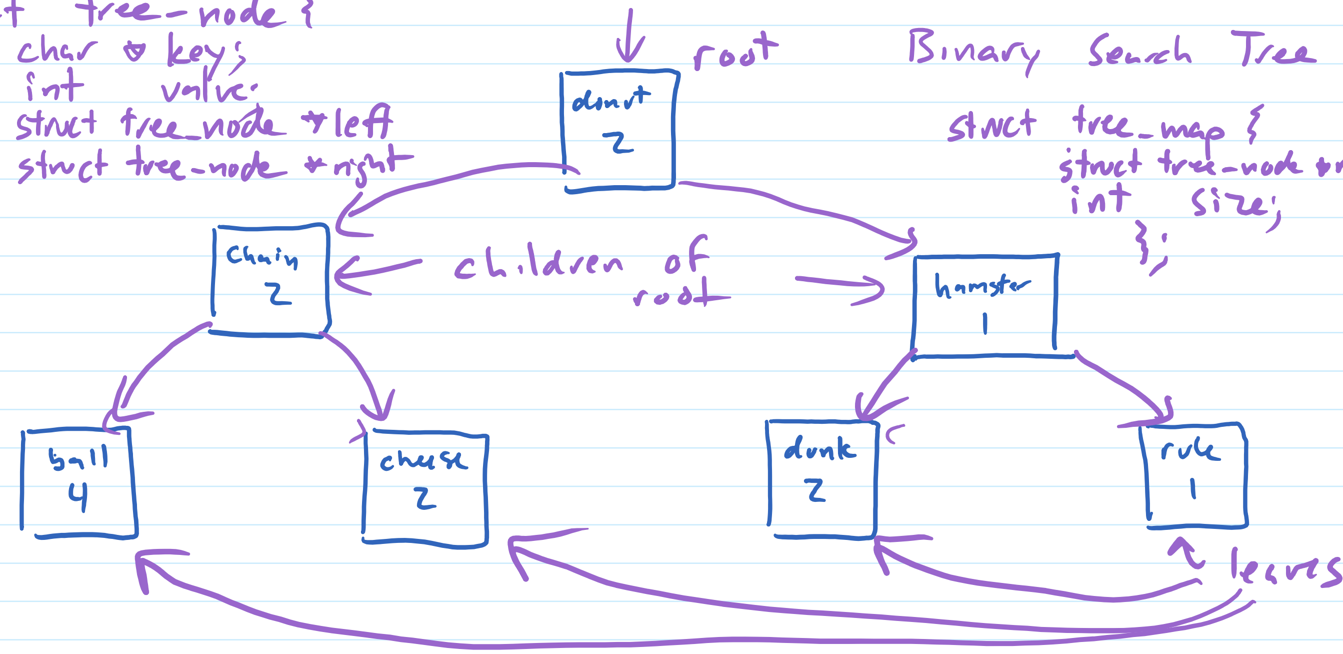
	unsorted array	hash table open addressing	hash table chaining	sorted array	balanced BST
contains_key/contains	**O(n)** worst case	**O(1) avg** **O(n) w.c.**	**O(1) avg** **O(n) w.c.**	**O(log n) w.c.**	**O(log n) w.c.**
put/add	**O(n) w.c.**	**O(1) avg** **O(n) w.c.**	**O(1) avg** **O(n) w.c.**	**O(n) w.c.**	**O(log n) w.c.**
get	**O(n) w.c.**	**O(1) avg** **O(n) w.c.**	**O(1) avg** **O(n) w.c.**	**O(log n) w.c.**	**O(log n) w.c.**
for_each	**O(n)**	**O(m)**	**O(m+n)**	**O(n)**	**O(n)**

n = number of (key, value) pairs
m = capacity of hash table
α = load factor = n / m

Binary Search Trees



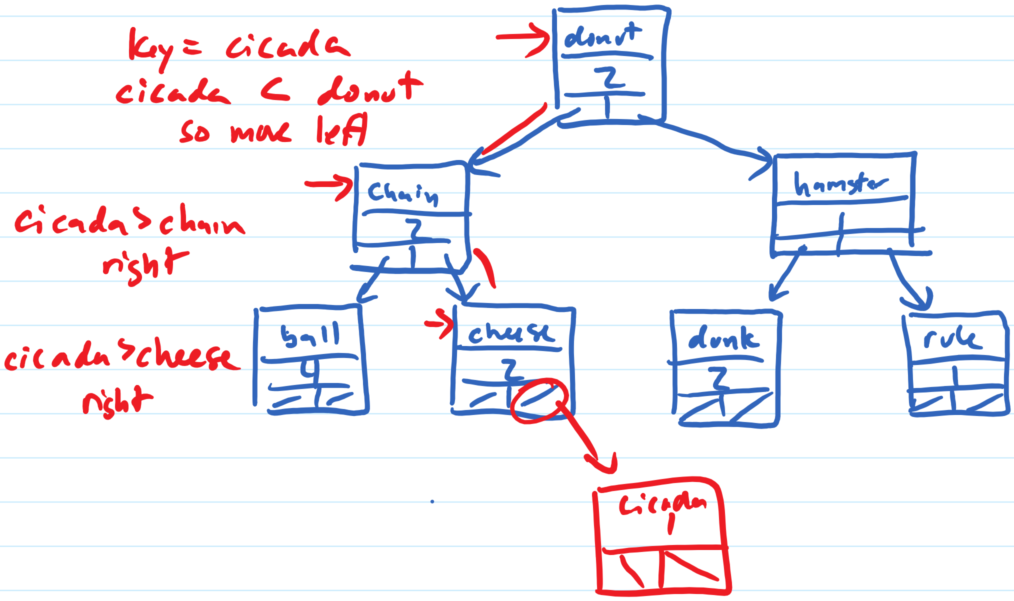

```
struct tree-node {
    char * key;
    int    value;
    struct tree_node *left
    struct tree-node *right
};
```



Binary Search Tree

```
struct tree_map {
  struct tree-node *root;
  int  size;
};
```

Adding to a BST




```
bool smap_contains_key(smap *m, const char *key)
{
   smap_node *curr = m->root;
   while (curr != NULL && strcmp(key, curr->key) != 0)
   {
      if (strcmp(key, curr->key) < 0)
      {
         curr = curr->left;
      }
      else
      {
         curr = curr->right;
      }//height = length of longest path

      root->leaf = O(log n) if tree is balanced

   }
   return (curr != NULL);
}
```

O(h)
↑
height of tree

Unshapely Trees

rule ball chain hamster cheese dunk donut