```
20144987 4 4 4 CLT OAK X 17 1.00 X
20144756 2 4 4 CLT BOS X US 1.00 X
201441020 2 4 4 CLT MCO X US 1.00 X
201442094 3 4 4 CHS CLT - 16 1.00 X
20144214 3 4 4 IAH CLT - US 1.00 X
201441020 1 4 4 AVL CLT - 16 1.00 X
201441110 4 4 4 PHX OAK X US 1.00 X
20144794 3 4 4 CLE CLT - 16 1.00 X
201441578 4 5 4 BNA PHL - YX 1.00 X
201442030 4 4 4 DCA BHM X US 1.00 X
201441020 3 4 4 MCO CLT - US 1.00 X
```

CLT 64827 % 8 = 3
OAK 78009 % 8 = 1
BOS 65958 % 8 = 6
MCO 76153 % 8 = 1
CHS
IAH

key → int
hash fxn

AVL 65207 % 8 = 7
CHS 66702 % 8 = 6
IAH 72240 % 8 = 0
PHX 79200 % 8 = 0
↳ % 16 = 0

hash table

	key	value
0		
1	OAK	1
2		
3	CLT	2
4		
5		
6	BOS	1
7	X	

contains_key : compute hash(key) % size
if NULL at that location, return NO
if something there, do comparison

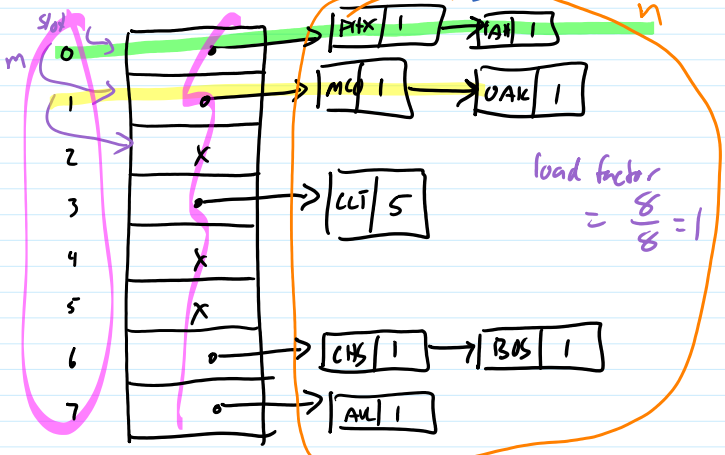
collision w/ BOS
↓
resolve collisions on put

```
20144987 4 4 4 CLT OAK X 17 1.00 X
20144756 2 4 4 CLT BOS X US 1.00 X
201441020 2 4 4 CLT MCO X US 1.00 X
201442094 3 4 4 CHS CLT - 16 1.00 X
20144214 3 4 4 IAH CLT - US 1.00 X
201441020 1 4 4 AVL CLT - 16 1.00 X
201441110 4 4 4 PHX OAK X US 1.00 X
20144794 3 4 4 CLE CLT - 16 1.00 X
201441578 4 5 4 BNA PHL - YX 1.00 X
201442030 4 4 4 DCA BHM X US 1.00 X
201441020 3 4 4 MCO CLT - US 1.00 X
```

CLT
AVL
IAH
PHX
OAK
CLE
BOS
MCO

hash table w/ chaining

colliding keys saved in same linked list



contains key : compute hash(key) % # slots
search linked list at that index (sequential search)

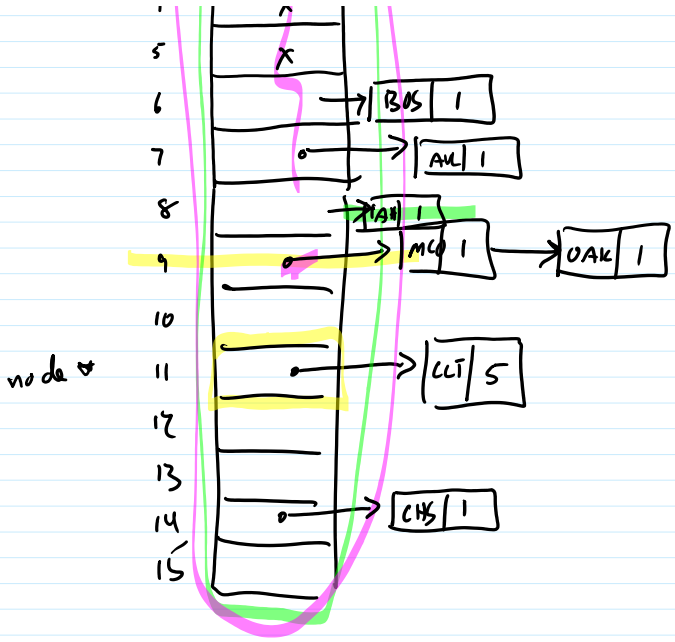
get : compute hash(key) % # slots
search linked list at that index (sequential search)
return value in node

put : compute hash(key) % # slots
search linked list at that index (sequential search)

hash fxn should distribute keys evenly

contains_key(OAK)
hash(OAK) = 78009
78009 % 16 = 9

(expected) running time = O(load factor)



expected running time = O(load factor)

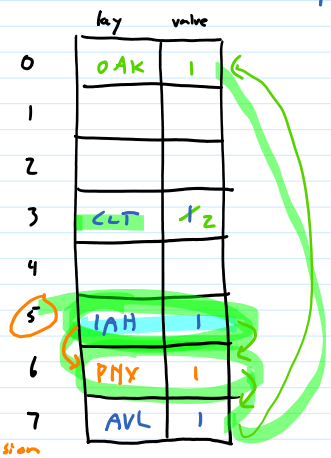
if $m \geq n$

$= O(1)$

worst case $O(n)$

(all keys collide in same slot)

hash table w/ open addressing
linear probing




```
20144987 4 4 4 CLT OAK X 17 1.00 X
20144756 2 4 4 CLT BOS X US 1.00 X
201441020 2 4 4 CLT MCO X US 1.00 X
201442094 3 4 4 CHS CLT - 16 1.00 X
20144214 3 4 4 IAH CLT - US 1.00 X
201441020 1 4 4 AVL CLT - 16 1.00 X
201441110 4 4 4 PHX OAK X US 1.00 X
20144794 3 4 4 CLE CLT - 16 1.00 X
201441578 4 5 4 BNA PHL - YX 1.00 X
201442030 4 4 4 DCA BHM X US 1.00 X
201441020 3 4 4 MCO CLT - US 1.00 X
```

	hash fxn
CLT	3
AVL	7
IAH	21
PHX	805
OAK	447
CLE	
BOS	
MCO	

21 % 8 = 5

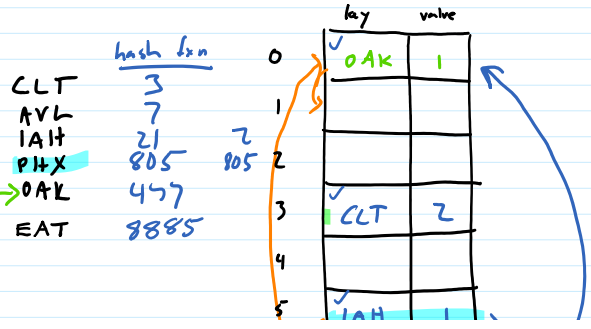
805 % 8 = 5 collision

contains_key
put
get

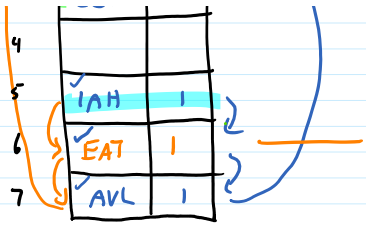
1) compute hash
2) compute %
3) seq. search through slots
until key found
or empty
or wrapped all the way around

O(n)

if $\alpha \leq \frac{1}{2}$ → O(1) expected time



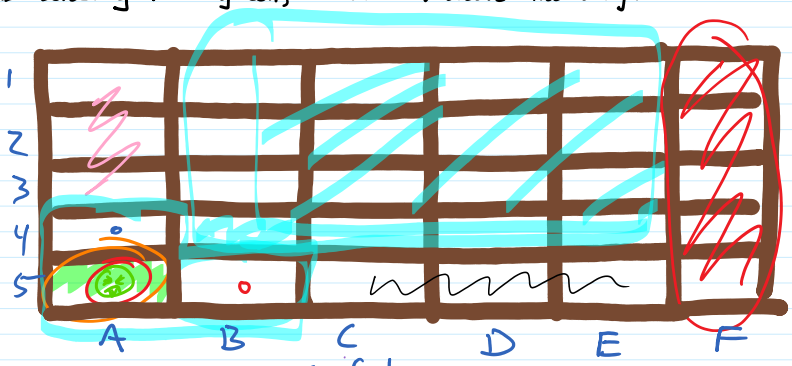
EAT 9885



Chomp

Play on m×n grid. Take turns selecting remaing cell, remove all above and to right

Last move loses.
(misere)



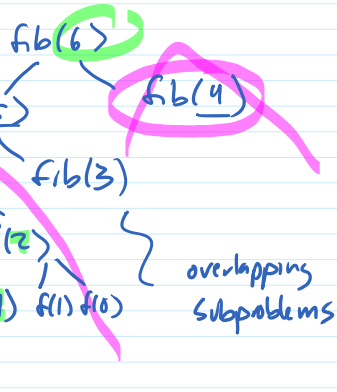
pos in finite impartial game

outcome_class(p)

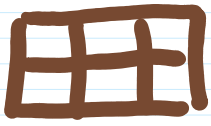
if p is end of game
  return value according to rules W/L
else
  S ← positions reachable in 1 move from p
  if S contains a losing position
    return WIN, and corresponding move
  else
    return LOSE

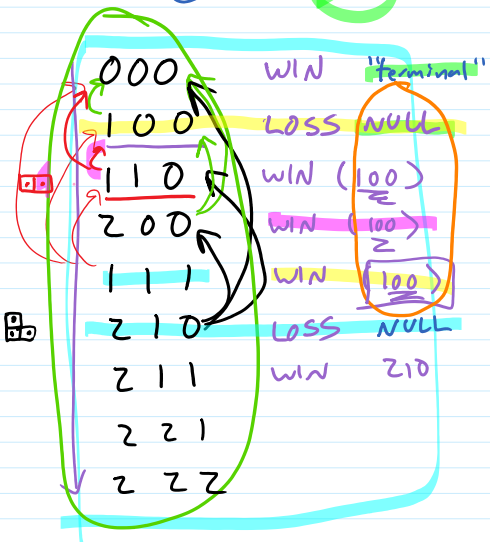
```
long fib_rec(int n)
{
  if (n < 2)
    {
      return n;
    }
  else
    {
      return fib_rec(n - 1) + fib_rec(n - 2);
    }
}
```

exponential running time!

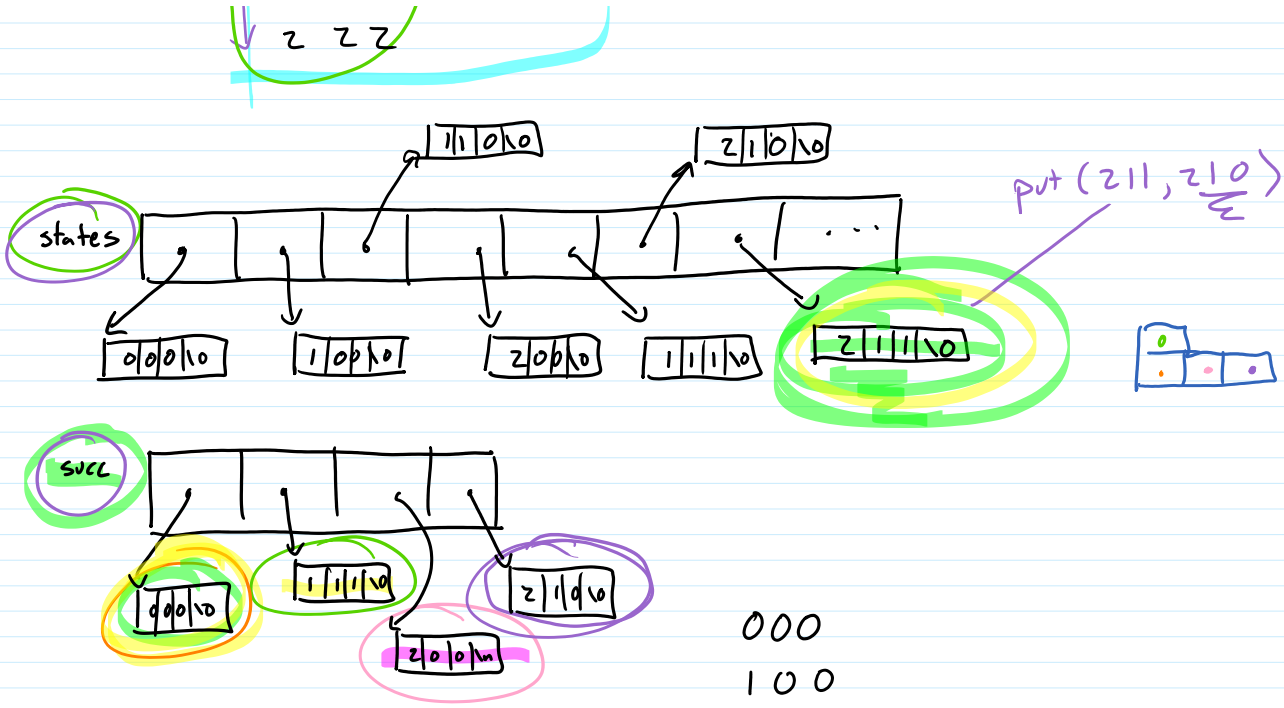


dynamic programming (replace recursion w/ array)





map keys: string representation of state
values: NULL (losing position)
string rep of state to move to for winning move



000
100
110
200
111
210
211
221
222