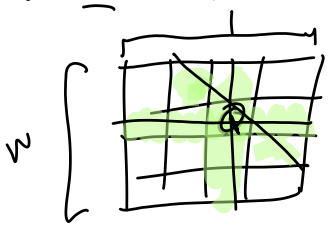
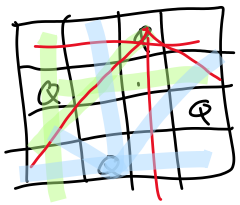
Backtracking

Sunday, October 4, 2020 4:39 PM

Ex: N-queens 8-queens



Place N queens in NxN board w/ no conflicts.



Brute force strategy

Solution space.

$\Sigma = \{\epsilon, Q\}$

Alt: Conf: $(n \times n)^n$

Conf: $\Sigma^{n \times n}$

Good: Conf $\rightarrow \{0,1\}$

Good(Conf) = 1. All n queens placed and exactly.

- 2. Check no conflicts diag.
- 3. Check no conflicts vert
- 4. Check no conflicts horizontal.

Analysis:

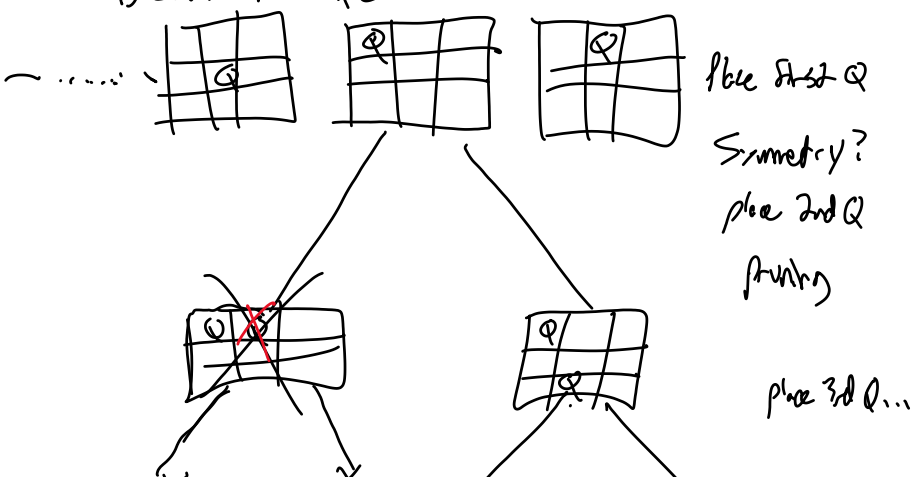
$T_{good}(n) = O(n^2)$ time to check 1 configuration.

Brute force: try all possible Confs,.. check good(Conf)?

$|Conf| = \binom{64}{8}$? 2^n

$T_{Bruteforce}(n) = O(2^{n^2})$ Exp.

Decision Tree



Analysis: ..... $O(2^{poly(n)})$

Subset Sum Problem

ex. Arr = [1, 3, 6, 10, 15, 20]

Target t: 22? 28?

Q. Is there a subset of A that adds up to t?

1+6+15? ✓ 10 3 15

You need to learn how to write recurrence solutions to problem

```
SubSum(A, t):
    True if t=0 if A=[]
    False otherwise
    SubSum(A', t-X) if A = X·A'   // include X
    OR
    SubSum(A', t)                 // do not include X
```

(X: one integer; A': rest of array)

Proof: Correct by induction
- If there is some solution then this finds it
- Any solutions found are correct

Analysis: $O(2^n)$

Fibonacci Numbers

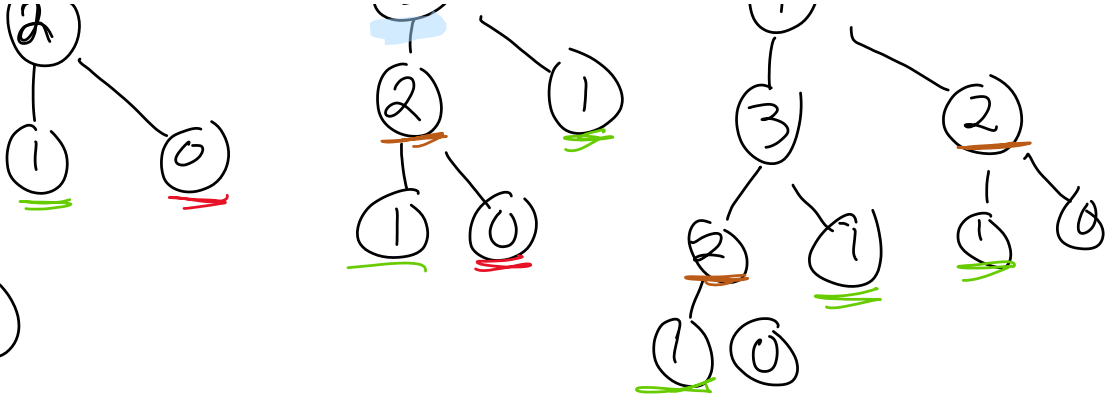
1 1 2 3 5 8 13

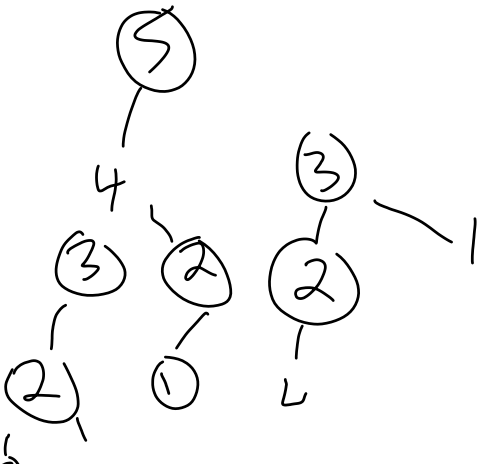
$fib(0) = 1$

$fib(1) = 1$

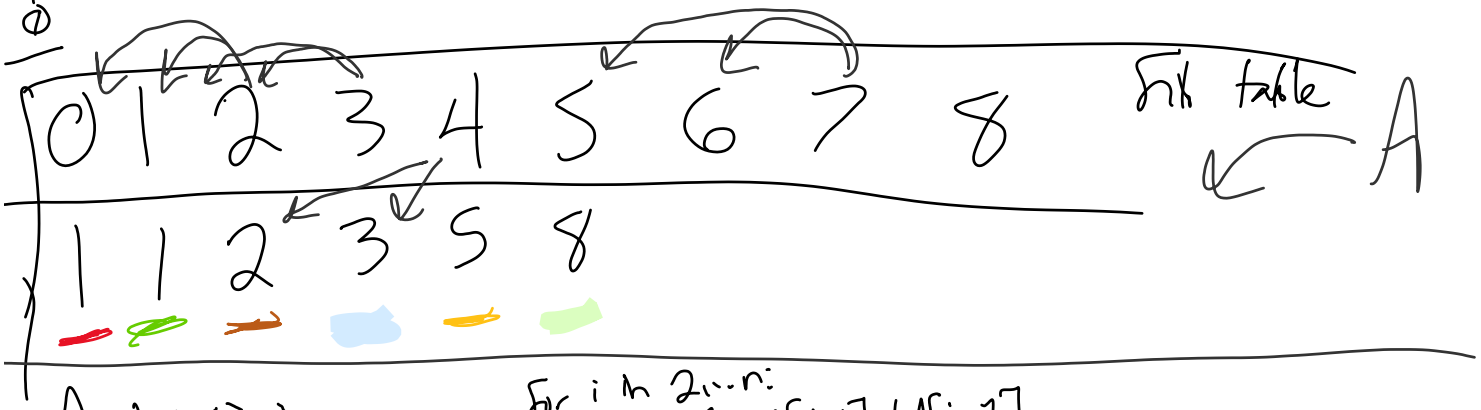
$fib(n) = fib(n-1) + fib(n-2)$

fib(2) (3) (4)





$O(2^n)$ or even worse



Analysis:

$T_{fib}(n) = O(n)$

```
for i in 2...n:
    A[i] = A[i-1] + A[i-2].
```

Storage: $|A| = ?$

Further optimizations:

- pipelining
  - can improve storage cost asymptotically
  - improve to $O(1)$
- don't get distracted by constant factor improvements.
- don't try to optimize storage in general.