February 26, 2019 10:59 AM

# Backtracking

recursion to try all sol'ns

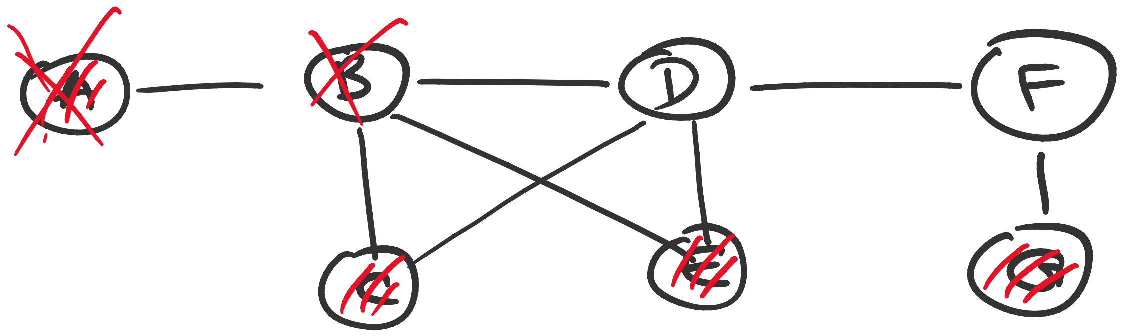
## Ex 1: Max Independent Set

Given undirected graph $G=(V,E)$, $|V|=n$, $|E|=m$

find subset $S \subseteq V$, maximizing $|S|$

s.t. $\forall u, v \in S \Rightarrow uv \notin E$

(an optimization problem)




e.g. $\{A, D, G\}$ of size 3

$\{A, C, E, G\}$ of size 4

## Alg'm 0: Brute force

try all subsets, test each one

$2^n$ subsets; $O(m)$ time each

$\Rightarrow \boxed{O(2^n \cdot m)}$ time

## Alg'm 1: Backtracking

## Alg'm 1: Backtracking

idea - consider a vertex v

Case 1. v is not in opt sol'n.

remove v & recurse.

Case 2. v is in opt sol'n.

add v to sol'n

remove v & all its neighbors

& recurse

MIS(G): // return size of max indep set

if G is empty return 0

pick vertex v in G

$$\text{return } \max\Big\{\text{MIS}(\underbrace{G-v}_{\text{graph formed by removing } v \text{ \& its incident edges}}),\ \text{MIS}(G-v-\underbrace{N(v)}_{\text{all vertices adj to } v})+1\Big\}$$

(recursion will automatically backtrack...)

$$\Rightarrow T(n) \le T(n-1) + T(n-1-\deg(v)) + O(m)$$

### Analysis 1:

$$\deg(v) \geq 0 \Rightarrow T(n) \le 2T(n-1) + O(m)$$

$$\Rightarrow \boxed{O(2^n m)} \text{ time}$$

## Analysis 2:

can remove deg-0 vertices (isolated) (& include in sol'n)

$\deg(v) \geq 1 \Rightarrow T(n) \leq T(n-1) + T(n-2) + O(m)$

( Fibonacci #s:

$$F_n = \begin{cases} F_{n-1} + F_{n-2} \\ \text{base case} \quad \ldots \end{cases}$$

guess $F_n \leq x^n$

want $x^{n-1} + x^{n-2} = x^n$

i.e. $x + 1 = x^2$ $\quad x^2 - x - 1 = 0$

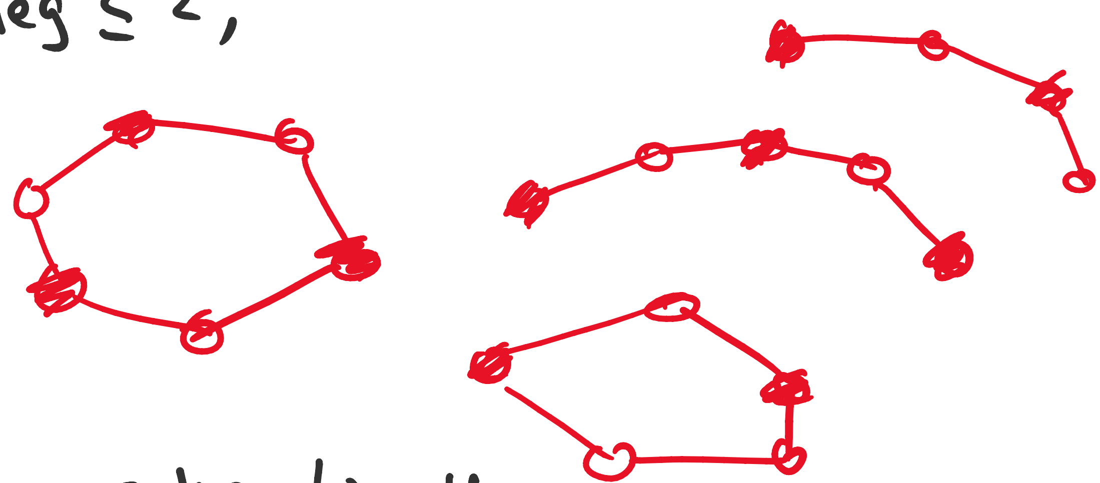
$\Rightarrow x = \frac{1+\sqrt{5}}{2} \sim 1.618$ )

$\Rightarrow \boxed{O(1.618^n \cdot m)}$ time

## Analysis 3:

pick vertex v of max deg.

if max deg $\leq 2$,



can solve directly

so may assume $\deg(v) \geq 3$

$\Rightarrow T(n) \leq T(n-1) + T(n-4) + O(m)$

$$\Rightarrow \boxed{O(1.381^n \cdot m)} \text{ time}$$

$x^4 = x^3 + 1$

**Rmk:** current record $\tilde{O}(1.221^n)$
(Fomin et al. '06)

---

## Ex 2: Longest Increasing Subsequence

Given numbers $a_1, \dots, a_n$,
find subsequence $a_{i_1}, \dots, a_{i_\ell}$ maximizing $\ell$
$i_1 < i_2 < \dots < i_\ell$
s.t. $a_{i_1} < a_{i_2} < \dots < a_{i_\ell}$

e.g. 8, 2, 3, 1, 10, 5, 17, 4, 9, 7, 12

### Alg'm 0: Brute force

try all subsequences

$O(2^n n)$ time

### Alg'm 1: Backtracking

idea – consider $a_n$

Case 1. if $a_n$ is not in opt sol'n
recurse on $a_1, \dots, a_{n-1}$

Case 2. if $a_n$ is in opt sol'n

add $a_n$

recurse on $a_1, \ldots, a_{n-1}$.

among all elems $< a_n$.

Extend problem:

$LIS(\langle a_1, \ldots, a_n \rangle, x)$:

// return length of longest increas. subseq whose largest elem $< x$.

if $n=0$ return 0

if $a_n < x$

return $\max\{LIS(\langle a_1, \ldots, a_{n-1} \rangle, x),$

$LIS(\langle a_1, \ldots, a_{n-1} \rangle, a_n) + 1\}$

else return $LIS(\langle a_1, \ldots, a_{n-1} \rangle, x)$

$LIS(\langle a_1, \ldots, a_n \rangle)$:

return $LIS(\langle a_1, \ldots, a_n \rangle, \infty)$

Naive Analysis:

$$T(n) \le 2\,T(n-1) + O(1)$$

$$\Rightarrow \boxed{O(2^n)} \text{ time}$$

Improvement?

Key observation:

# distinct subproblems

$( \ldots ) = O(n^2)$

# distinct subproblems

$$\le \underbrace{n}_{\text{\# prefixes of input seq.}} \cdot \underbrace{(n+1)}_{\text{\# choices for } x \text{ (including } \infty)} = O(n^2).$$

avoid solving same subproblem over & over again

by remembering answers

memoization $\Rightarrow$ dynamic programming

## Memoized Version 1:

$LIS(i,j)$: // $x = a_j$, input $\langle a_1, \ldots, a_i \rangle$

```
if i=0 return 0
if L[i,j] ≠ undef return L[i,j]
if a_i < a_j
    return L[i,j] = max { LIS(i-1,j),
                          LIS(i-1,i)+1 }
else return L[i,j] = LIS(i-1,j)
```

## Memoized Version 2:

idea - evaluate bottom-up! ($a_{n+1} = \infty$)

```
for j=1 to n+1 do L[0,j] = 0
for i=1 to n do
    for j=1 to n+1
        if a_i < a_j   L[i,j] = max{ L[i-1,j],
                                     L[i-1,i]+1 }
        else L[i,j] = L[i-1,j]
```

return $L[n, n+1]$

$\Rightarrow$ $\boxed{O(n^2)}$ time