Computing Cliques

Don Kreher

Michigan Technological University

kreher@mtu.edu

Software for this talk:

`http://www.math.mtu.edu/~kreher/src/clique.zip`

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph.

A *clique* is a set of pairwise adjacent vertices.

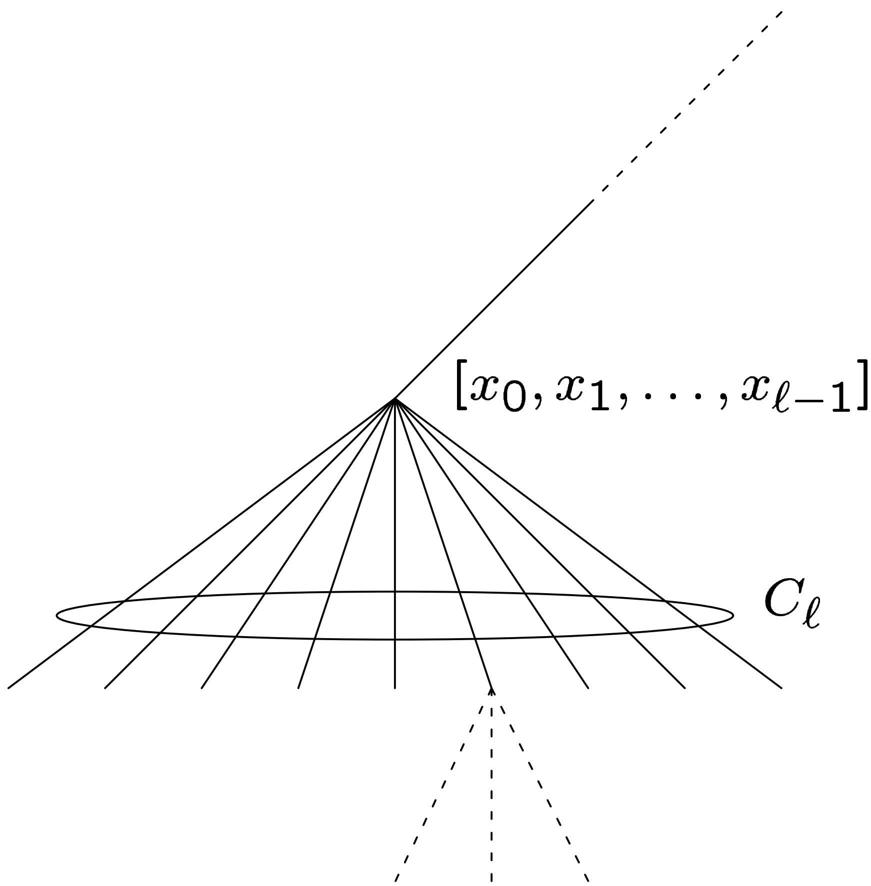
maximal - not contained in a larger clique.

maximum - has largest size.

$[x_0, x_1, \ldots, x_{\ell-1}]$ a clique $\Rightarrow \{x_i, x_j\} \in \mathcal{E} \ \ \forall\, i \neq j$.

Can we extend it to a larger clique?

Need $x \in \mathcal{V}$ so that $\{x_i, x\} \in \mathcal{E} \ \ \forall\, i$

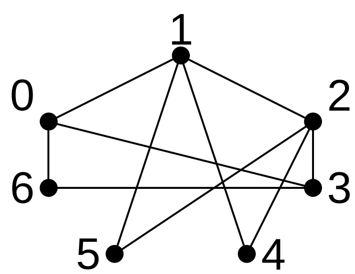

$[x_0, x_1, \ldots, x_{\ell-1}]$
C_ℓ

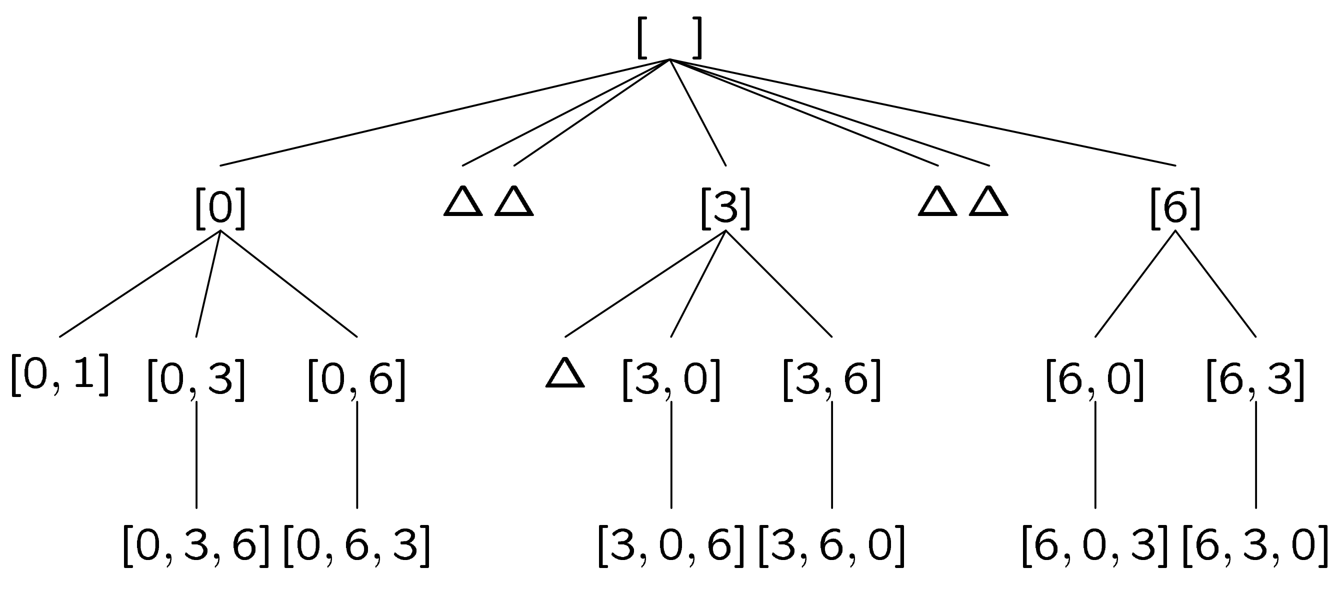
ALLCLIQUES1(ℓ)
output $[x_0, x_1, \ldots, x_{\ell-1}]$
if $\ell = 0$
then $\mathcal{C}_\ell \leftarrow \mathcal{V}$
else $\mathcal{C}_\ell \leftarrow \{v \in \mathcal{C}_{\ell-1} : \{v, x_{\ell-1}\} \in \mathcal{E}\}$
for each $x \in \mathcal{C}_\ell$
do $\begin{cases} x_\ell \leftarrow x \\ \text{ALLCLIQUES1}(\ell + 1) \end{cases}$

There's a problem?

– you get cliques more than once – each $k!$ times

Example:





Solution: Impose and ordering:

$$0 < 1 < 2 < 3 < \cdots < n-1$$

Thus

$$\mathcal{C}_\ell = \{v \in \mathcal{C}_{\ell-1} \setminus \{x_0, \ldots, x_{\ell-1}\} : \{v, x_{\ell-1}\} \in \mathcal{E}, v > x_{\ell-1}\}$$

Precompute for each $v \in \mathcal{V}$:

$$\begin{aligned} \mathcal{A}_v &= \{u \in \mathcal{V} : \{u, v\} \in \mathcal{E}\} \\ \mathcal{B}_v &= \{u \in \mathcal{V} : u > v\} \end{aligned}$$

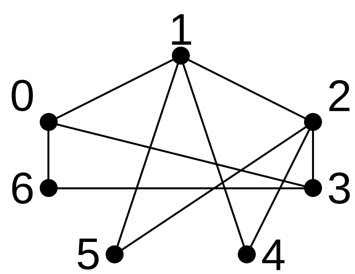
Then: $\mathcal{C}_\ell = \mathcal{A}_{x_{\ell-1}} \cap \mathcal{B}_{x_{\ell-1}} \cap \mathcal{C}_{\ell-1}$

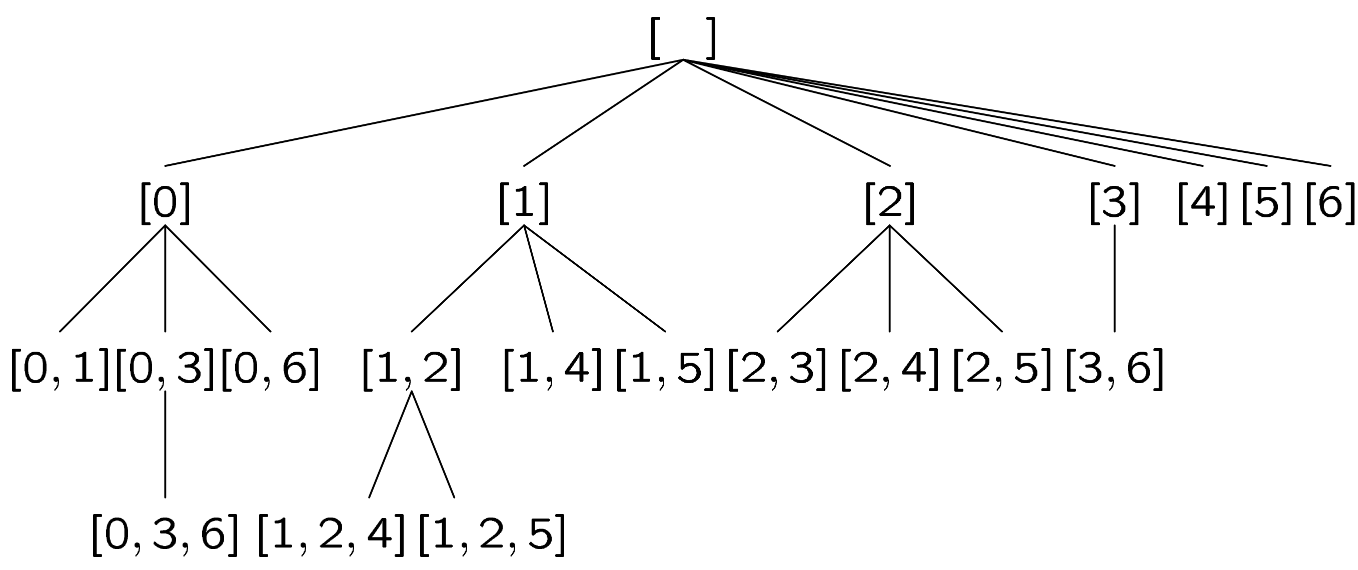
Now

```
ALLCLIQUES2(ℓ)
  output [x_0, x_1, ..., x_{ℓ-1}]
  if ℓ = 0
    then C_ℓ ← V
    else C_ℓ ← A_{x_{ℓ-1}} ∩ B_{x_{ℓ-1}} ∩ C_{ℓ-1}
  for each x ∈ C_ℓ
    do { x_ℓ ← x
       { ALLCLIQUES2(ℓ + 1)
```

Computes each clique exactly once.

Example:





Average-case Analysis:
For a given graph $\mathcal{G}$ the running time is $O(n\ c(\mathcal{G}))$ where

$$c(\mathcal{G}) = \#\text{ of cliques in } \mathcal{G} = \#\text{ of nodes in the tree}$$

Let $|\mathcal{V}| = n$.

$$\begin{array}{rcl} \mathcal{G} = (\mathcal{V}, \emptyset) & \Rightarrow & c(\mathcal{G}) = n + 1 \\ \mathcal{G} = K_n & \Rightarrow & c(\mathcal{G}) = 2^n \end{array}$$

Let

$$\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_{2^{\binom{n}{2}}}$$

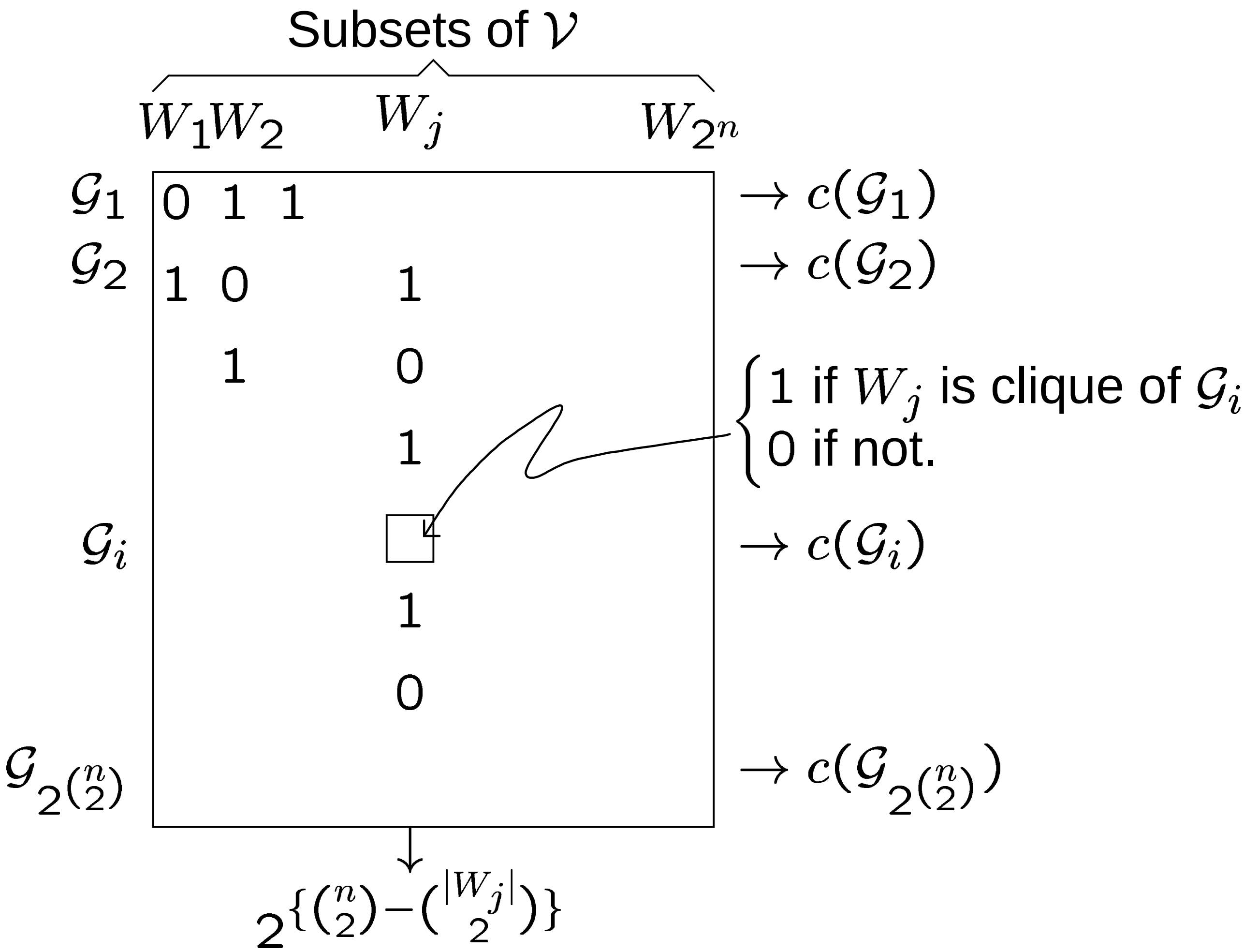
be the all the graphs on $\mathcal{V}$, where $|\mathcal{V}| = n$.

Then the average running time is:

$$O(n\bar{c}(n))$$

where

$$\bar{c}(n) = \frac{1}{2^{\binom{n}{2}}} \sum_{i=1}^{2^{\binom{n}{2}}} c(\mathcal{G}_i)$$



Therefore: $\sum_i c(\mathcal{G}_i) \;=\; \sum_j 2^{\{\binom{n}{2}-\binom{|W_j|}{2}\}}$

So: $\bar{c}(n) = \frac{1}{2^{\binom{n}{2}}} \sum_i c(G_i) \;=\; \frac{1}{2^{\binom{n}{2}}} \sum_j 2^{\{\binom{n}{2}-\binom{|W_j|}{2}\}}$

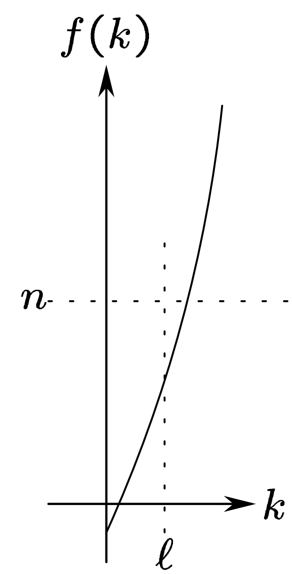
$$\bar{c}(n) = \frac{1}{2^{\binom{n}{2}}} \sum_{k=0}^{n} \binom{n}{k} 2^{\{\binom{n}{2} - \binom{k}{2}\}} = \sum_{k=0}^{n} \binom{n}{k} 2^{-\binom{k}{2}}$$

$$= \sum_{k=0}^{n} t_k; \qquad \text{where } t_k = \binom{n}{k} 2^{-\binom{k}{2}} = \frac{\binom{n}{k}}{2^{\binom{k}{2}}}$$

$$\frac{t_k}{t_{k-1}} = \frac{n-k+1}{k2^{k-1}}$$

$$t_k \geq t_{k-1} \Leftrightarrow n - k + 1 \geq k2^{k-1}$$

$$\Leftrightarrow n \geq \underbrace{k - 1 + k2^{k-1}}_{f(k)}$$



$f(k) = k - 1 + k2^{k-1}$ is strictly increasing.

So: $t_0 \leq t_1 \leq \cdots \leq t_\ell > t_{\ell+1} > \cdots > t_n \ \exists \ell$

Therefore: $\bar{c}(n) \leq (n+1)t_\ell$.

$$\begin{aligned} f(\log_2 n) &= \log_2 n - 1 + \log_2 n 2^{\log_2 n - 1} \\ &= \log_2 n - 1 + \frac{n \log_2 n}{2} \end{aligned}$$

$$n \geq 4 \Rightarrow \log_2 n \geq 2 \Rightarrow f(\log_2 n) > n$$

Therefor: $t_k < t_{k-1}$ when $k \geq \log_2 n$

So: $\ell \leq \log_2 n$ if $n \geq 4$.

Thus: $t_\ell = \binom{n}{\ell} 2^{-\binom{\ell}{2}} < \binom{n}{\ell} < n^\ell \leq n^{\log_2 n}$

Hence: $\bar{c}(n) \in O(n^{\log_2 n + 1})$

Consequently the average running time of AllCliques2 on a graph with n vertices is

$$O(n^{\log_2 n + 2}).$$

To find a maximum clique we can modify:

```
AllCliques2(ℓ)
  output [x_0, x_1, ..., x_{ℓ-1}]
  if ℓ = 0
    then C_ℓ ← V
    else C_ℓ ← A_{x_{ℓ-1}} ∩ B_{x_{ℓ-1}} ∩ C_{ℓ-1}
  for each x ∈ C_ℓ
    do { x_ℓ ← x
       { AllCliques2(ℓ + 1)
```

To:

```
MaxClique1(ℓ)
  if ℓ = 0
    then C_ℓ ← V
    else C_ℓ ← A_{x_{ℓ-1}} ∩ B_{x_{ℓ-1}} ∩ C_{ℓ-1}
  for each x ∈ C_ℓ
    do { x_ℓ ← x
       { if (ℓ + 1) > OptSize
       {   then { OptSize ← ℓ + 1
       {        { OptClique ← [x_0, x_1, ..., x_ℓ]
       { MaxClique1(ℓ + 1)
  main
   OptSize ← 0
   MaxClique1(0)
   output OptClique
```

Can we prune the search?

Given a partial clique

$$X = [x_0, x_1, \ldots, x_{\ell-1}]$$

found at level ℓ, we have:

$$\mathcal{C}_\ell = \{v \in \mathcal{C}_{\ell-1} \setminus X : v > x_{\ell-1} \ \& \ \{v, x_{\ell-1}\} \in \mathcal{E}\}$$

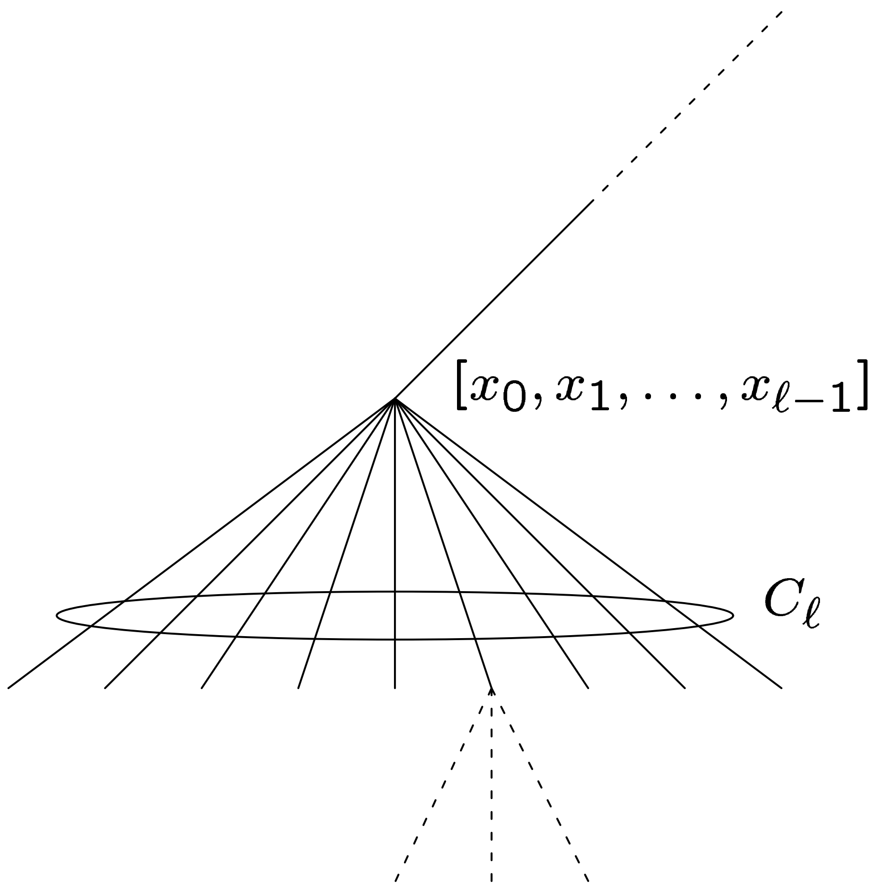
So if we define

$$B(\ell, X) = \ell + |\mathcal{C}_\ell|$$

Then no clique containing X can be extended to a clique of size larger than $B(\ell, X)$.

This gives the bounding function:

```
SIZEBOUND(ℓ, X)
  return ℓ + |C_ℓ|
```


$[x_0, x_1, \ldots, x_{\ell-1}]$
C_ℓ

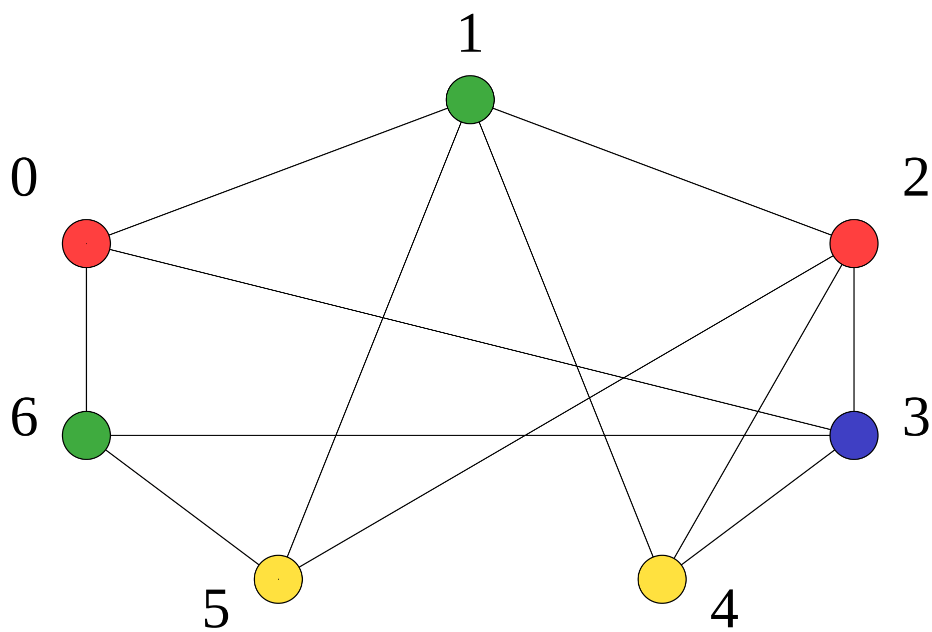
MaxClique2(ℓ)

if $\ell = 0$
  then $\mathcal{C}_\ell \leftarrow \mathcal{V}$
  else $\mathcal{C}_\ell \leftarrow A_{x_{\ell-1}} \cap B_{x_{\ell-1}} \cap \mathcal{C}_{\ell-1}$
$B \leftarrow B(\ell, X)$
for each $x \in \mathcal{C}_\ell$
  do
    if $B \leq OptSize$
      then return
    $x_\ell \leftarrow x$
    if $(\ell + 1) > OptSize$
      then $OptSize \leftarrow \ell + 1$; $OptClique \leftarrow [x_0, x_1, \ldots, x_\ell]$
    MaxClique2($\ell + 1$)

main
  $OptSize \leftarrow 0$
  $MaxClique2(0)$
  output $OptClique$

Are there better bounding functions?

Vertex Coloring. *If $\mathcal{G}$ has a k-vertex coloring, then the maximum clique size is at most K*



A priori to backtracking greedily color the vertices.

$$color[x] = \text{ the color assigned to vertex } x$$

Define

$$B(\ell, X) = \ell + |\{color[x] : x \in \mathcal{C}_\ell\}|$$

GREEDYCOLOR($\mathcal{G} = (\mathcal{V}, \mathcal{E})$)

global $color$

$k \leftarrow 0$

for each $i \in \mathcal{V}$

do $\begin{cases} h \leftarrow 0 \\ \textbf{while } h < k \textbf{ and } \mathcal{A}_i \cap ColorClass[h] \neq \emptyset \\ \quad \textbf{do } h \leftarrow h + 1 \\ \textbf{if } h = k \\ \quad \textbf{then } \begin{cases} k \leftarrow k + 1 \\ ColorClass[h] \leftarrow \emptyset \end{cases} \\ ColorClass[h] \leftarrow ColorClass[h] \cup \{i\} \\ color[i] \leftarrow h \end{cases}$

return (k)

SAMPLINGBOUND(ℓ, X)

return $\ell + |\{color[x] : x \in \mathcal{C}_\ell\}|$

Alternatively greedy color when the bound is computed.

GREEDYBOUND(ℓ, X)

$k \leftarrow$ GREEDYCOLOR($\mathcal{G}[\mathcal{C}_\ell]$)

return $\ell + k$

NoBound(ℓ, X)

return n

SizeBound(ℓ, X)

return $\ell + |\mathcal{C}_\ell|$

SamplingBound(ℓ, X)

return $\ell + |\{color[x] : x \in \mathcal{C}_\ell\}|$

GreedyBound(ℓ, X)

$k \leftarrow$ GreedyColor($\mathcal{G}[\mathcal{C}_\ell]$)
return $\ell + k$

Bounding Function	Complexity
No Bound	$O(1)$
SizeBound	$O(n)$
SamplingBound	$O(n)$
GreedyBound	$O(n^2)$

Size of state space trees on random graphs

Edge density .5

# of vertices	50	100	150	200	250
# of edges	607	2535	5602	9925	15566
size of maximum clique	7	9	10	11	11
bounding function					
none	8687	257145	1659016	7588328	26182672
size bound	3204	57225	350310	1434006	5008767
sampling bound	2268	44072	266246	1182514	4093535
greedy bound	430	5734	22599	91671	290788

Edge density .75

# of vertices	25	50	75	100	125
# of edges	236	959	2045	3720	5780
size of maximum clique	11	14	15	17	18
bounding function					
none	25570	2083770	12385596	186543706	1414266577
size bound	1840	91663	426279	5370268	35108264
sampling bound	794	37218	195567	2225982	15615755
greedy bound	91	2843	10476	70404	413421