Dynamic Programming

Seminar on Exact Exponential Algorithms
of Prof. Haim Kaplan, Semester B 2020

Presented by Guy Korol
Department of CS, Tel-Aviv University

Based on the book: “Exact exponential algorithms”
by Fomin, Kratsch, 2010 (FK)

Dynamic Programming - intro

- Start by solving small or trivial instances of the problem
- Gradually resolving larger and harder sub problems by composing solution from the smaller sub problems
- Having great importance for designing polynomial and exponential time algorithms
- An (unwanted) property of exponential time dynamic prog. Is that they usually require exponential space (the relation between time and space will be covered later on the seminar)

What we will cover today?

- Dynamic programming algorithms to solve several problems
  - **Permutation problems**
    - TSP on Graphs of bounded degrees (time improvements over general graphs)
  - **Partition problems**
    - Minimum set cover
    - Minimum dominating set (reduce)

  - If we'll have time...
    - Optimal Coloring – find the chromatic number of a graph (k coloring) – improved time complexity

TSP on Graphs of Bounded degrees

Recall the analysis from last week:

- $OPT[S, c_i]$ is the shortest path that starts at c_1, goes through S, and finishes at c_i
- If $S = \{c_i\}$ then $OPT[S, c_i] = d(c_1, c_i)$
- For $S, |S| > 1$:

$$OPT[S, c_i] = \min\{\, OPT[S \setminus \{c_i\}, c_k] + d(c_k, c_i) \mid c_k \in S \setminus \{c_i\}\,\}$$

- Finally $OPT = \min\{\, OPT[\{c_2, \dots, c_n\}, c_i] + d(c_i, c_1) \mid i \in \{2,3, \dots, n\}\,\}$

- For each $OPT[S, c_i]$ we spend $O(n)$ time ➔
- $O^*(2^n)$ overall

TSP on Graphs of Bounded degrees

- We'll revisit the dynamic programming algorithm presented before

- We associate a graph $G = (V, E)$ on vertex set $V = \{c_1, c_2, \dots, c_n\}$.
  Two vertices c_i and c_j are adjacent in G iff $d(c_i, c_j) < \infty$.
- We show that for graphs of **bounded maximum degree**, the above algorithm for TSP run in time $O^*(c^n)$ for some $c < 2$.

- The proof is based on the observation that the running time of the algorithm is proportional to the number of **connected vetex sets** in a graph.

TSP on Graphs of Bounded degrees

- We call a **vertex set** $C \subseteq V$ of a graph G **connected** if $G[C]$ (the subgraph of G induced by C, is connected)
  - Notice that for a graph on n vertices, the number of connected vertex sets might be as large as 2^n (clique).

- For graphs of bounded degree, it is possible to show that the maximum is **significantly smaller** than 2^n.

- The proof of this fact is based on the folllownig lemma of **Shearer**

Shearer's Lemma

- Let U be a finite set of elements with a collection $S = \{S_1, S_2, \dots, S_r\}$ of subsets of U such that every element $u \in U$ is contained in at least δ subsets of S.

- Let T be another family of subsets of U.
- For each $1 \leq i \leq r$, we define the projection $\mathrm{T}_i = \{\mathrm{t} \cap \mathrm{S_i} : \mathrm{t} \in \mathrm{T}\}$

- Then:

$$|\mathrm{T}|^{\delta} \leq \prod_{i=1}^{r} |T_i|$$

- Proof is long, using induction on δ and some other lemmas as showed in: "The Probabilistic Method (2008)" by Alon N., Spencer, J.

Shearer's Lemma

- **Example:**

  $|\mathbf{U}| = 8$, $|\boldsymbol{S}| = 6$, $|\mathbf{T}| = 2$, $\boldsymbol{\delta} = 2$

$\boldsymbol{T_1} = \{\boldsymbol{S_1}, \emptyset\}$

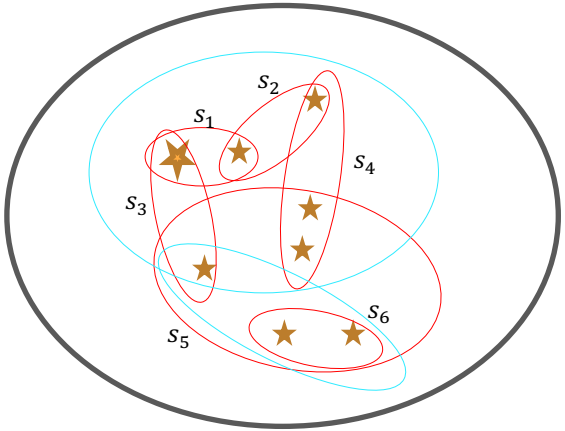
$\boldsymbol{T_2} = \{\boldsymbol{S_2}, \emptyset\}$

$\boldsymbol{T_3} = \{\qquad\}$

$\boldsymbol{T_4} = \{\boldsymbol{S_4}, \emptyset\}$

$\boldsymbol{T_5} = \{\qquad\}$

$\boldsymbol{T_6} = \{\boldsymbol{S_6}, \emptyset\}$



$|T|^2 \leq \prod_{i=1}^{6} |T_i| \quad \rightarrow \quad 2^2 \leq 2 * 2 * 2 * 2 * 2 * 2$

Intuition: As there are at least δ subsets that contain each element, the projection of T on all of the S_i will have at least δ T_i with "enough" elemets on T

Jensen's inequality

- For the next lemma, we'll use Jensen's inequality, a classical inequality for convex functions

- Let I be an interval of R. A real-valued function f defined on I is a **convex function** on I, if:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$$

for all $x, y \in I$ and $0 \leq \lambda \leq 1$.

Jensen's inequality – cont.

- Let f be a convex function on the interval I of R^1.
  Then:

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i)$$

- The above holds by activating Jensen's inequality n times

- Using Shearer's lemma and Jensen's inequality we can upper-bound the maximum number of connected vertex subsets in graphs of bounded maximum degree

Back to TSP on Graphs of Bounded degrees

- Let $G = (V, E)$ be a graph on n vertices of **maximum degree Δ**.
  Then, the number of connected vertex subsets in G is at most:

$$\left(2^{\Delta+1} - 1\right)^{\frac{n}{\Delta+1}} + n$$

Proof:

- The closed neighborhood of a vertex v, denoted by $N[v]$, consists of v and all of its neighbors: $N[v] = \{u \in V : \{u, v\} \in E) \cup \{v\}$.

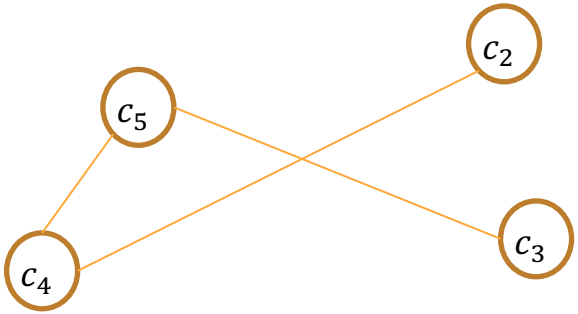
- We start by defining sets S_v for all $v \in V$. Initially, we assign $S_v \coloneqq N[v]$.

- For every vertex v which is **not** of maximum degree Δ, we add v to (arbitrary) $\Delta - d(v)$ sets S_u which **do not already contain v**.

Back to TSP on Graphs of Bounded degrees

Example:

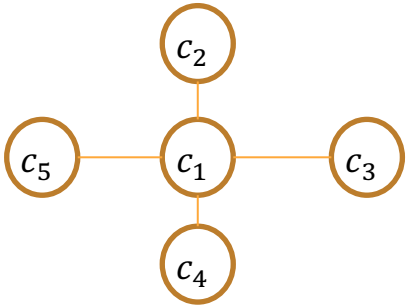
- In this case, for $\Delta = 2$, we need to “artificially" add c_2 and c_3 to an additional neighbour as $N[c_2] = \{c_4\}$ and $N[c_3] = \{c_5\}$ which is smaller than Δ.



Back to TSP on Graphs of Bounded degrees

- After completing the previous construction, every vertex v is contained in exactly $\Delta + 1$ sets S_u and: $\sum_{v \in V} |S_v| = n(\Delta + 1)$

- Let C be the set of all connected vertex sets in G of size at least two. For every $v \in V$, the number of subsets in the projection $C_v = \{c \cap S_v : c \in C\}$ is at most $2^{|S_v|} - 1$.
  - This is because for any connected set C of size at least two, $C \cap N[v] \neq \{v\}$, otherwise $v's$ neighbors would be attached as well.
  - Thus, $\{v\}$ doesn't belong to C_v.



Back to TSP on Graphs of Bounded degrees

- By Shearer's Lemma:

$$|C|^{\Delta+1} \leq \prod_{v \in V}\left(2^{|S_v|} - 1\right)$$

- Let us define $f(x) = 2^x - 1$.

  The function $\log\big(f(x)\big)$ is convex on the interval $[1, +\infty)$, and by Jensen's inequality we have that:

$$-\log f\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) \leq -\frac{1}{n}\log f\left(\sum_{i=1}^{n} x_i\right)$$

  for any $x_i \geq 1$.

- In particular:

$$\frac{1}{n}\sum_{v\in V}\log f(|S_V|) \leq \log f\left(\frac{1}{n}\sum_{v\in V}|S_v|\right) = \log f(\Delta + 1)$$

Back to TSP on Graphs of Bounded degrees

- By taking exponentials and combining with the above result by Shearer's lemma, we can obtain:

$$\prod_{v \in V} (2^{|S_v|} - 1) \leq (2^{\Delta+1} - 1)^n$$

- Therefore:

$$|C| \leq \left(2^{\Delta+1} - 1\right)^{\frac{n}{\Delta+1}}$$

- Finally, the number of connected sets of side one is at most n, which concludes the last final case and the proof of the lemma.

Minimum Set Cover

- In the MSC problem we're given a universe U of n elements and a collection $S = \{S_1, S_2, \dots, S_m\}$ of (non-empty) subsets of U. The goal is to find the minimum cardinality of a subset $S' \subseteq S$ which **covers** U, i.e. satisfies:

$$\bigcup_{S \in S'} S = U$$

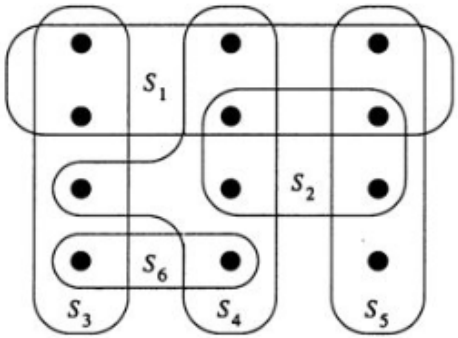
- We say that a subset $S' \subseteq S$ **covers** a subset $S \subseteq U$, if every element in S belongs to at least one member of S'.
  - Note that a minimum set cover of (U, S) can trivially be found in time $O(n2^{\boldsymbol{m}})$ by checking all subsets of S.

- We're going to proove that there is an $O(nm2^{\boldsymbol{n}})$ algorithm to solve the MSC, to be able to use it later on, for a better solution of the **Minimum dominating set** problem.

Minimum Set Cover – Example

- $\{S_1, S_4, S_5, S_6\}$ is a cover

- $\{S_3, S_4, S_5\}$ is a **minimum** cover



Minimum Set Cover - Proof

- For every nonempty subset $u \subseteq U$, and for every $j = 1,2, \dots, m$ we define: $OPT[u; j]$ as the **minimum cardinality of a subset** of $\{S_1, S_2, \dots, S_j\}$ that **covers** u.

- If $\{S_1, \dots, S_j\}$ doesn't cover u then we set $OPT[u; j] \coloneqq \infty$.
- For every subset $u \subseteq U$, we set $OPT[u; 1] = 1$ if $u \subseteq S_1$, and $OPT[u; 1] = \infty$ otherwise.

- Then, in step $j + 1,\ j \in \{1, 2, \dots, m - 1\},\ OPT[u; j + 1]$ is computed for all $u \in U$ in $O(n)$ time as follows:

$$OPT[u; j + 1] = \min\{OPT[u; j], OPT\left[u \backslash S_{j+1}; j\right] + 1\}$$

Minimum Set Cover – Summary

- The proof yields an algorithm to compute $OPT[u; j]$ for all $u \in U$ and all $j \in \{1,2, \dots, m\}$ of overall running time $O(nm2^{\boldsymbol{n}})$.

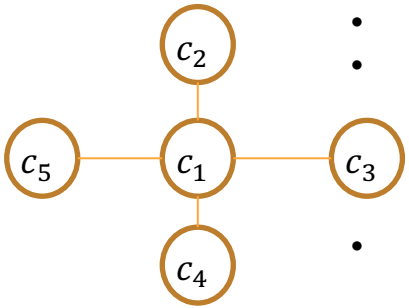
- Therefore, $OPT[U; m]$ will have the cardinality of a minimum set cover of (U, S).

- Now, we'll see how the above algorithm can be used to break the 2^n barrier for the Minimum Dominating Set problem

Minimum Dominating Set - Intro

- We are given an undirected graph $G = (V, E)$. The task is to find the minimum cardinality of a **dominating** set in G.

- **Dominating set:** A vertex subset $D \subseteq V$ is a **dominating set** for G if every vertex of G is either:
  - In D
  - Adjacent to some vertex in D



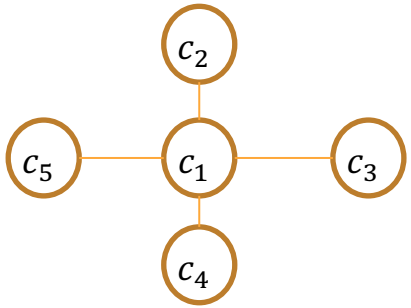
- $\{c_1\}$ is dominating
- $\{c_2\}$ isn't dominating

- $\gamma(G) = 1$

- The **domination** number $\gamma(G)$ of a graph G is the minimum cardinality of a dominating set of G.
- The MDS problem asks us to compute $\gamma(G)$.

Minimum Dominating Set – Reduction

- The MDS problem can be **reduced** to the MSC problem by imposing:
  - $U = V$, $S = \{N[v]: v \in V\}$
- Note that $N[v]$ is the set of vertices **dominated by** v, thus, D is a dominating set of G iff $\{N[v]: v \in D\}$ is a set cover of $\{N[v]: v \in V\}$.
  - Every **minimum** set cover of $\{N[v]: v \in V\}$ corresponds to a minimum dominating set of G.



- $U = V = \{c_1, \dots, c_5\}$
- $S_1 = \{c_1, c_2, c_3, c_4, c_5\}$
- $S_2 = \{c_1, c_2\}$
- $S_3 = \{c_1, c_3\}$
- $S_4 = \{c_1, c_4\}$
- $S_5 = \{c_1, c_5\}$

- S_1 is a set cover of U

Minimum Dominating Set – Problem intro

- Let $G = (\mathrm{V}, \mathrm{E})$ be a graph on n vertices given with an independent set I. Then, a **minimum dominating set** of G can be computed in time $O^*\left(2^{n-|I|}\right)$.
  - In particular, a MDS of a **bipartite** graph on n vertices can be computed in time $O^*\left(2^{\frac{n}{2}}\right)$.

Independent Set - Recall from last week

- Given a graph G = (V,E) find a largest subset $I \subset V$ such that:
  (u,v) ≠ E for every u,v ∈ I

Minimum Dominating Set – Proof

- Let $J = V \setminus \mathrm{I}$ be the set of vertices outside of the independent set I.

- Instead of trying all possible subsets D of V as potential dominating sets, we try all possible **projections** of D on J.

- For each such projection $J_D = J \cap D$, we decide wether J_D can be extended to a dominating set of G by adding only vertices of I.

- In fact, for every $J_D \subseteq J$ the smallest possible number of vertices of I should be added to J_D to obtain a dominating set of G.

Minimum Dominating Set – Proof

- For every subset $J_D \subseteq J,$ we show how to construct **a set $\boldsymbol{D}$** such that:

$$|D| = \min\{|D'|: D' \; is \; a \; dominating \; set \; \boldsymbol{and} \; J \cap D' = J_D\}$$

- Then the result will be:

$$\gamma(G) = \min_{J_D \subseteq J} |D|$$

- The set $I_D = I \backslash \mathrm{N}[J_D]$ is a subset of D since I is an independent set, and the vertices of I_D cannot be domindated by J_D (as we reduced its neighbors).

- Then, the only vertices that are not dominated by $I_D \cup J_D$ are the vertices:

$$J_X = J \; \backslash (\mathrm{N}[J_D] \cup N[I_D]).$$

Minimum Dominating Set – Proof

- Therefore, to find D we have to add to $I_D \cup J_D$ the minimum number of vertices from $I \backslash \mathrm{I}_D$ that **domindate all vertices of** J_X.

- We reduce this problem to MSC by imposing $U = J_X$ and $S = \{N[v] : v \in I \backslash \mathrm{I}_D\}$.
- By the MSC proof, such a problem is solvable in time $2^{|J_X|} * n^{O(1)}$.
  Thus, the running time of the algorithm is (up to polynomial factor):

$$\sum_{J_D \subseteq J} \binom{|J|}{|J_D|} * 2^{|J_X|} = \sum_{J_D \subseteq J} \binom{|J|}{|J_D|} * 2^{|J(N[J_D] \cup N[I_D])|} \leq \sum_{J_D \subseteq J} \binom{|J|}{|J_D|} * 2^{|J \backslash \mathrm{J}_D|} = \sum_{J_D \subseteq J} \binom{n - |I|}{|J_D|} * 2^{n - |I| - |J_D|}$$

$$= 3^{n - |I|}$$

Minimum Dominating Set – Improvement

- In the remaining of the proof we'll show how to improve the running time $3^{n-|I|}$ to the claimed $2^{n-|I|}$.

- We want to show that: $\sum_{J_D \subseteq J} \binom{|J|}{|J_D|} * 2^{|J \setminus J_D|}$ can be evaluated in time $2^{n-|I|} * n^{O(1)}$.

- Instead of trying **all** subsets of J and then for each subset construct a dominating set D, we'll do the following.

Minimum Dominating Set – Improvement

- For every subset $X \subseteq J$, we compute a minimum subset of I which dominates X. We can compute this by performing the following dynammic programming.

- Let's fix an ordering $\{v_1, v_2, \dots, v_k\}$ of I. We define $D_{X,i}$ a subset of $\{v_1, \dots, v_i\}$ of the **minimum** size which **dominates** X.

- $D_{X,k}$ is a subset of I, dominating X, of minimum size.

Minimum Dominating Set – Improvement

- We initialize $D_{\emptyset,k} = \emptyset$ and for $X \neq \emptyset$ we initialize:

$$D_{X,1} = \begin{cases} v_1 & if\ X \subseteq N[v_1] \\ \{v_1, v_2, \dots, v_k\} & otherwise \end{cases}$$

- To compute the values $D_{X,i}$ for $i > 1$, we consider two cases. Either the optimum set is a subset of $\{v_1, \dots, v_{i-1}\}$, or it contains v_i, so:

$$D_{X,i} = \begin{cases} D_{X,i-1} & if\ \left|D_{X,i-1}\right| < \left|D_{X\setminus \mathrm{N}[v_i],i-1}\right| + 1, \\ D_{X\setminus \mathrm{N}[v_i],i-1} \cup \{v_i\} & otherwise \end{cases}$$

Minimum Dominating Set – Improvement

- The computation of all sets $D_{X,k}$, $X \subseteq J$, takes time $2^{|J|} * n^{O(1)}$

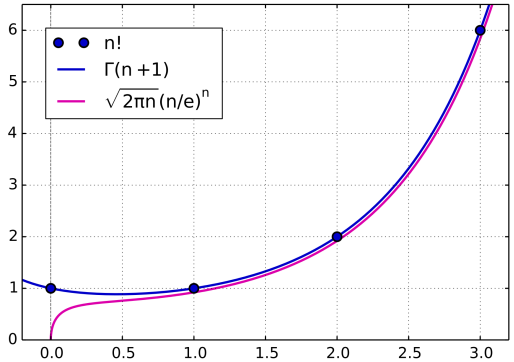
- Once these sets are computed, construction of a set that will be with a **minimum** cardinality for **every subset** $J_D \subseteq J$ can be done in polynomial time by computing $D = I_D \cup J_D \cup D_{J_X,k}$ (given all artifacts).

- In total, the running time needed to compute them, is the time required to compute $D_{X,k}$ plus the time required to compute for every given subset J_D the subsets I_D and $D_{J_X,\,k}$. So, up to a polynomial factor:

$$2^{|J|} + \sum_{J_D \subseteq J} \binom{|J|}{|J_D|} = 2^{|J|+1} = \mathbf{2^{n-|I|+1}}$$

Binary Eentropy Function

- The following binary entropy function is very helpful in computations involving binomial coefficeients.



Stirling's Formula

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \leq n! \leq 2\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

The binary entropy function h is defined by:

$$h(\alpha) = -\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha)$$

for $\alpha \in (0, 1)$

Binary Eentropy Function

- By the above we're getting the following:

for $\alpha \in (0,1)$

$$\frac{1}{\sqrt{8n\alpha(1-\alpha)}} * 2^{h(\alpha)n} \leq \sum_{i=1}^{\alpha n} \binom{n}{i} \leq 2^{h(\alpha)n} = \left(\frac{1}{\alpha}\right)^{\alpha n} * \left(\frac{1}{1-\alpha}\right)^{(1-\alpha)n}$$

Minimum Dominating Set – algorithm summary

By making use of the above proof and the binary entropy function, it is possible to construct an algorithm solving the MDS problem **faster than by trivial brute-force** in $O^*(2^n)$

Proof

- Every **maximal** independent set of a graph G is also a dominating (not necessary minimum) set of G.

- First, we compute any maximal independent set of G, which can be done by a greedy algorithm in polynomial time

Minimum Dominating Set – algorithm summary

- If the size of the maximal independent set found is larger than αn, for: $0.2271 < \alpha < 0.227711$, by the previous proof we can compute $\gamma(G)$ in time: $2^{n-\alpha n} * n^{O(1)} = O^*(2^{0.7729n}) = O^*(1.7088^n)$.

- If the size of the maximal independent set is at most αn, then we know that $\gamma(G) \leq \alpha n$, and by trying all subsets of size at most αn, we can find a minimum dominating set in time:

$$\binom{n}{\alpha n} * n^{O(1)} = O^*\left(\binom{n}{0.22711n}\right)$$

By making use of the formula for the entropy function above, we estimate:

$$O^*\left(\binom{n}{0.22711n}\right) = O^*(2^{0.7729n}) = O^*(1.7088^n)$$

Optimal Coloring Problem

- We are given an undirected graph $G = (V, E)$. The task is to compute the chromatic number of G.

- The computation of the chromatic number of a graph is a typical partition problem, as we saw on the first lecture, and the trivial brute-force solution would be:
  - For every vertex, try every possible color – will take $O^*(n^n)$ co cliques

- We will prove that the chromatic number of an n-vertex graph can be computed in time $O^*\left(\left(1 + \sqrt[3]{3}\right)^n\right) = O^*(2.4423^n)$ by a dynammic programming algorithm.

Optimal Coloring Problem - Proof

- For every $X \subseteq V$, we define $OPT[X] = \chi(G[X])$, the chromatic number of the **subgraph** of G, induced by X.

- For every X, in the order of increasing cardinality, the following recurrence is used:
  - $OPT[\emptyset] = 0,$
  - $OPT[X] = 1 + \min\{OPT[X \backslash \mathrm{I}]: I \text{ is a maximal independent set of } G[X]\}$

- We claim that $\chi(G) = OPT[V]$.

Optimal Coloring Problem – Explanation

- Every k-coloring of a graph G is a partition of the vertex set into k independent sets (each set has a different color)

- We may always modify the k-coloring such that one independent set is **maximal**, as we can always use the same color for the rest of the vertices

- Therefore, an optimal coloring of G is obtained by **removing a maximal independent** set I from G, and **adding** an optimal coloring of $G \backslash \mathrm{I}$.

Optimal Coloring Problem – Time Improvements

- Let $n = |V|$, the algorithm runs on all subsets $X \subseteq V$, and for every subset X, it runs on all its subsets I, which are maximal independent sets in $G[X]$.

- The number of such sets is at most $2^{|X|}$, for every subset X.
- Thus, the number of steps of the algorithm is up to a polynomial factor of:

$$\sum_{i=1}^{n} \binom{n}{i} * 2^i = 3^n$$

- But, in this one, we **don't take** into account that the algorithm doesn't run on all subsets of a subset X, but only on **maximal independents** sets of $G[X]$.

Optimal Coloring Problem – Time Improvements

- As we saw in the first lecture, the number of maximal independent sets in a graph of i vertices is at most $3^{\frac{i}{3}}$ and these sets can be enumerated in time $O^*\left(3^{\frac{i}{3}}\right)$.

- Thus, up to a polynomial factor, the running time of the algorithm can be bounded by:

$$\sum_{i=1}^{n} \binom{n}{i} * 3^{\frac{i}{3}} = \left(1 + \sqrt[3]{3}\right)^n < 2.4423^n$$

- The running time obtained here can be improved to $O^*(2^n)$ by combining dynamic programming with **inclusion-exclusion** (another topic in our seminar).