Priority algorithms — part 1: exact algorithms

Joan Boyar[1], Kim S. Larsen[1], and *Denis Pankratov*[2]

[1] University of Southern Denmark

[2] Concordia University

OLAWA 2020

What is a Greedy Algorithm?

- CLRS: “A ***greedy algorithm*** always makes the choice that looks best at the moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution.”
- Pros: conceptually simple, easy to design, describe and implement, fast running times.
- Cons: don’t always work well.
- To study power and limitations of greedy algorithms we need a formal model.
- Surprise: there is no universally agreed upon model!

Models of Greedy Algorithms

- *Matroid* – Whitney (1935), Edmonds (1971)
- *Greedoid* – Korte and Lovasz (1981)
- *Online Algorithms/Competitive Analysis* – Sleator and Tarjan (1985), Graham (1966), …
    - "In a somewhat vacuous sense, all online algorithms are greedy, since an online algorithm is defined by a function f of the current request, the current state, and the history." – Karlin and Irani
- **Priority Algorithms** – Borodin, Nielsen, Rackoff (2002)
    - Online algorithms on steroids

Priority Algorithms

- Many greedy algorithms have the structure:
  - Sort input items in some way
  - Run some online algorithm on the sorted input
  - E.g.: Kruskal’s algorithm for MST
  - Sometimes the algorithm resorts remaining input items in between online decisions, e.g., Prim’s algorithm for MST
- Modelling obstacle:
  - Many problems admit an input order for which trivial online algorithm is optimal (e.g., matching)
  - How do we restrict sorting functions?

Priority Algorithms

- Input is a collection $I = \langle x_1, \ldots, x_n \rangle$ of items from the universe U
- Overcoming modelling obstacle:
  - Sorting function is not allowed to depend on I
  - Sorting function has to order the entire universe U
  - Often specified by a priority function $P : U \to \mathbb{R}$

Priority Algorithms

Fixed Priority Algorithm:

Specify $P : U \to \mathbb{R}$

While I is not empty

$x \leftarrow argmax_{x \in I}\{P(x)\}$

make an irrevocable decision on x

$I \leftarrow I \setminus \{x\}$

Adaptive Priority Algorithm:

Specify initial $P : U \to \mathbb{R}$

While I is not empty

$x \leftarrow argmax_{x \in I}\{P(x)\}$

make an irrevocable decision on x

$I \leftarrow I \setminus \{x\}$

Update $P : U \to \mathbb{R}$

MST Example

- $U = \mathbb{N} \times \mathbb{N} \times (\mathbb{R} \cup \{-\infty, +\infty\})$
- Item (u, v, w): u, v – endpoints of an edge; w – weight of the edge
- Kruskal's algorithm:
  - fixed priority $P(u, v, w) = -w$, i.e., negative projection on the third coordinate
  - decision: accept the item if and only if it does not create a cycle
- Prim's algorithm:
  - Adaptive priority: let S be the set of encountered vertices so far
  - $P(u, v, w) = -w$ if u or v is in S and $-\infty$ otherwise

Properties of the model

- Information-theoretic
  - Similar to online algorithms
  - Makes lower bounds very strong
  - Upper bounds typically translate to polynomial time algorithms
- Does not capture all greedy algorithms
- In particular, greedy algorithms could precompute some functions providing useful side-information for the online phase
  - This could be modeled with advice

Adding Advice to Fixed Priority

Oracle

Binary advice tape ϕ

b_1	b_2	b_3	b_4	b_5	$\cdots$

Input $I = \langle x_1, \ldots, x_n \rangle$

Priority algorithm
$x_{i_1}, \ldots, x_{i_n}$

Output $O = \langle y_{i_1}, \ldots, y_{i_n} \rangle$

Adversary

Priority function P does not depend on advice

Borodin, Boyar, Larsen, *Pankratov*. Advice Complexity of Priority Algorithms. WAOA 2018

Adding Advice to Adaptive Priority

I is the input

$P_0 : U \to \mathbb{R}$ is the initial priority function
$i \leftarrow 1$

While $I \neq \emptyset$

$x_i \leftarrow argmax_{x \in I}\{P_{i-1}(x)\}$

read zero or more bits of advice from the tape

$s_i \leftarrow$ known contents of the advice tape

$d_i \leftarrow$ decision of ALG on x_i

$P_i \leftarrow$ updated priority function

$I \leftarrow I \setminus \{x_i\}$

$i \leftarrow i + 1$

Model 1: $P_i \coloneqq P_i(x_1, \ldots, x_i, s_i)$

Model 2: $P_i \coloneqq P_i(x_1, \ldots, x_i, d_1, \ldots, d_i)$

Rest of the Talk

- $G = (V, E)$ - simple undirected graph;
- $|V| = n, |E| = m$
- $S \subseteq V$ is a vertex cover if $\{u, v\} \in E$ implies either $u \in S$ or $v \in S$
- Input items $\left(v, N(v)\right) \in \mathbb{N} \times \bigcup_{k \in \mathbb{N}} \binom{\mathbb{N}}{k}$

Theorem. There is an adaptive priority algorithm, $RejectFirst$, that solves the Minimum Vertex Cover (VC) problem with at most $\left(\frac{7}{22}\right) n \approx 0.3182\, n$ bits of advice in Model 2 on triangle free graphs with maximum degree at most 3.

Theorem. There is an adaptive priority algorithm, $RejectFirst$, that solves the Minimum Vertex Cover (VC) problem with at most $\left(\frac{7}{22}\right)n \approx 0.3182\,n$ bits of advice in Model 2 on triangle free graphs with maximum degree at most 3.

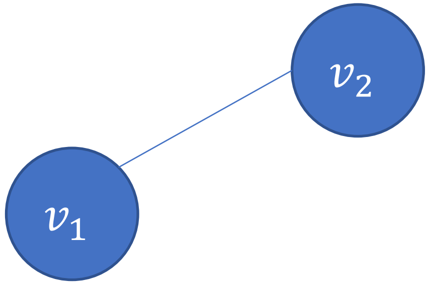
- Exact algorithm, so also solves MIS (Maximum Independent Set)
- Graph class might seem restrictive, but MIS/VC for this class or related classes has rich history:
  - Brooks (1941), Staton (1979), Jones (1990), Heckman and Thomas (2001), Harant et al (2008), Kanj and Zhang (2010), …
- No adaptive priority algorithm without advice can achieve approximation ratio better than $4/3$. (Borodin et al. 2010).
- No online algorithm with fewer than $\frac{n}{3} - c$ bits of advice can achieve optimality. (this work, not presented).

$RejectFirst$ Intuition

- Suppose vertices are revealed in order
$$v_1, v_2, \dots, v_n.$$
- **Current neighborhood** of v_i is
$$N(v_i) \setminus \{v_1, \dots, v_{i-1}\}.$$
- The size of the current neighborhood of v_i is its **current degree,** $\boldsymbol{cdeg(v_i)}$**.**
- An advice bit: accept v_i into a VC or not.
- Key idea: often we can infer whether to accept or reject v_i without advice.
- Specify priority functions so that we maximize the number of vertices that do not require advice.

$RejectFirst$ Ideas

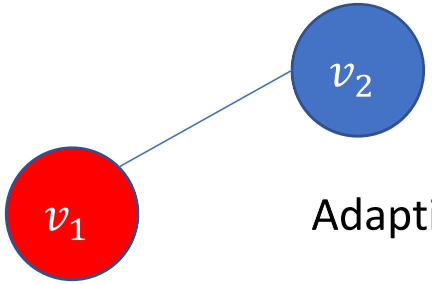
- If $cdeg(v_i) = 0$ then v_i can be rejected.
- If $cdeg(v_i) = 1$ then v_i can be rejected and its unique neighbor accepted.



$cdeg(v_1) = 1$
Has highest priority

$RejectFirst$ Ideas

- If $cdeg(v_i) = 0$ then v_i can be rejected.
- If $cdeg(v_i) = 1$ then v_i can be rejected and its unique neighbor accepted.

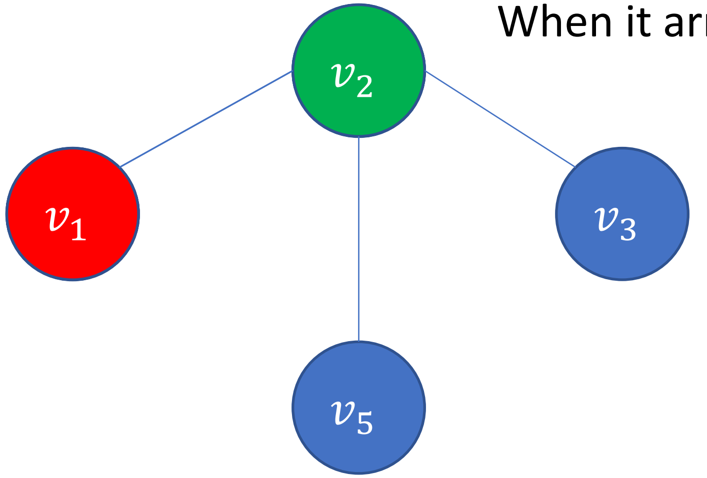


Adaptively give its unique neighbor highest priority

Reject without advice

$RejectFirst$ Ideas

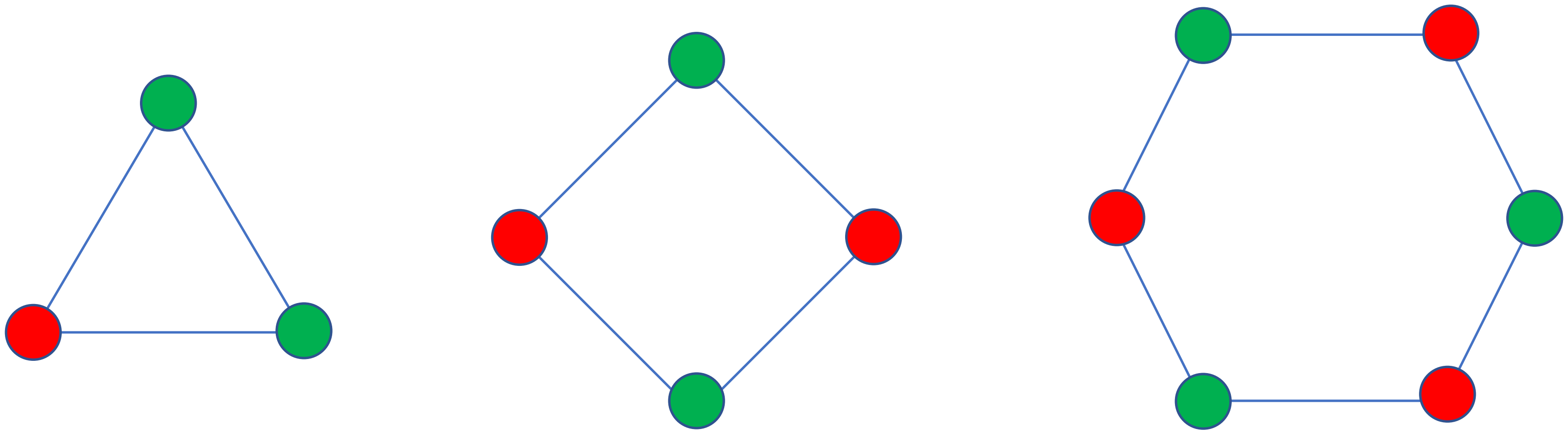
- If $cdeg(v_i) = 0$ then v_i can be rejected.
- If $cdeg(v_i) = 1$ then v_i can be rejected and its unique neighbor accepted.



When it arrives, accept it

$RejectFirst$ Ideas

- Suppose that $cdeg(v_i) = 2$ for **all remaining vertices**.
- This implies that the remaining graph is a set of vertex disjoint cycles.
- Can be processed without advice:



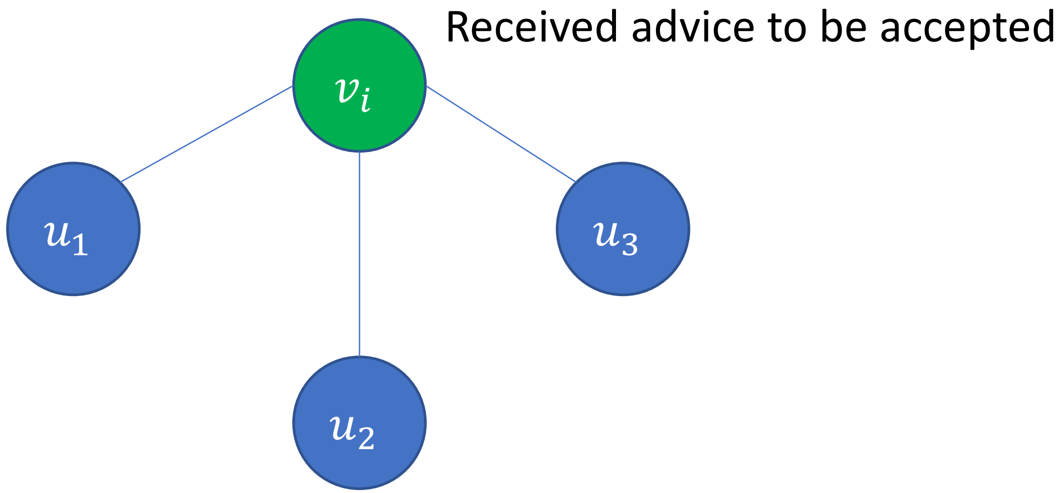
- Giving $cdeg = 2$ lowest priority allows us to detect this case.

- Therefore, we can handle $cdeg = 0, 1,$ and 2 vertices without advice.
- Need more tricks to avoid giving advice to many $cdeg = 3$ vertices.
- IDEA: **cooperation** between the oracle and the algorithm.

<u>**INTUITION**</u> (actual details more involved)

- Suppose $cdeg(v_i) = 3$ and v_i requires advice
- Moreover, suppose there is a minimum VC which includes v_i and there is another minimum VC which excludes v_i
- ORACLE prefers to give advice to **REJECT** v_i

Implies that at most one neighbor of v_i with accept advice is accepted

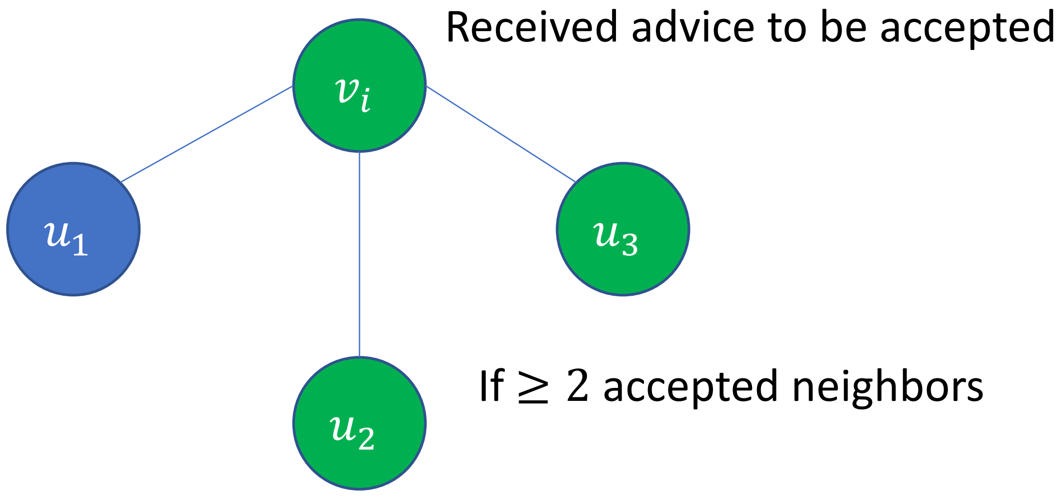


- Therefore, we can handle $cdeg = 0, 1,$ and 2 vertices without advice.
- Need more tricks to avoid giving advice to many $cdeg = 3$ vertices.
- IDEA: **cooperation** between the oracle and the algorithm.

INTUITION (actual details more involved)

- Suppose $cdeg(v_i) = 3$ and v_i requires advice
- Moreover, suppose there is a minimum VC which includes v_i and there is another minimum VC which excludes v_i
- ORACLE prefers to give advice to **REJECT** v_i

Implies that at most one neighbor of v_i with accept advice is accepted



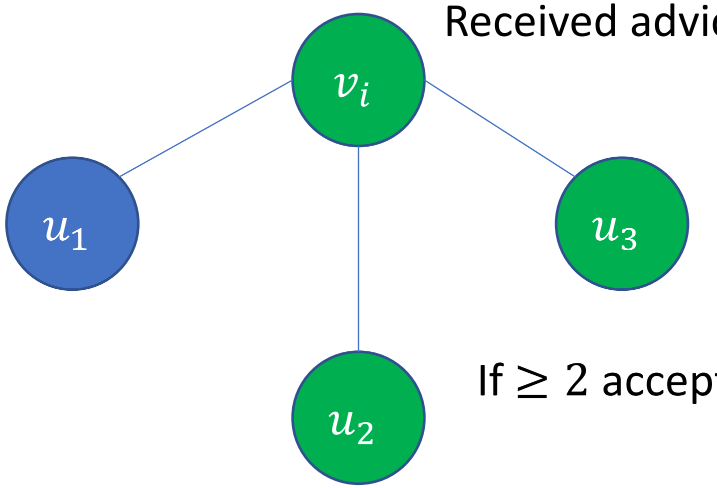
- Therefore, we can handle $cdeg = 0, 1,$ and 2 vertices without advice.
- Need more tricks to avoid giving advice to many $cdeg = 3$ vertices.
- IDEA: **cooperation** between the oracle and the algorithm.

<u>INTUITION</u> (actual details more involved)

- Suppose $cdeg(v_i) = 3$ and v_i requires advice
- Moreover, suppose there is a minimum VC which includes v_i and there is another minimum VC which excludes v_i
- ORACLE prefers to give advice to **REJECT** v_i

Implies that at most one neighbor of v_i with accept advice is accepted

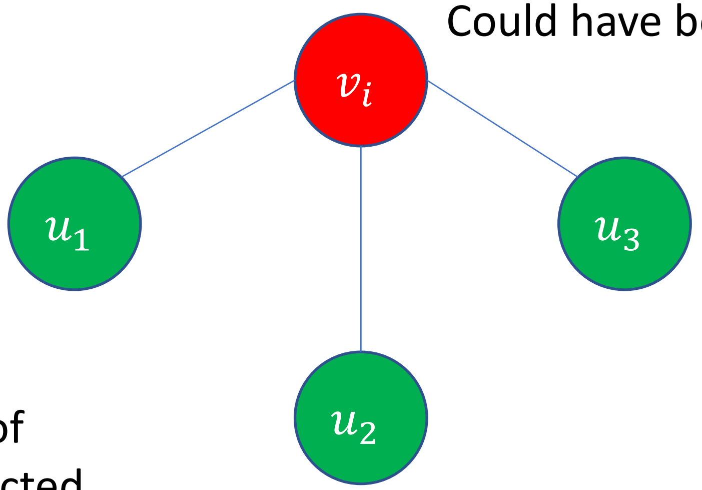
Received advice to be accepted



If ≥ 2 accepted neighbors

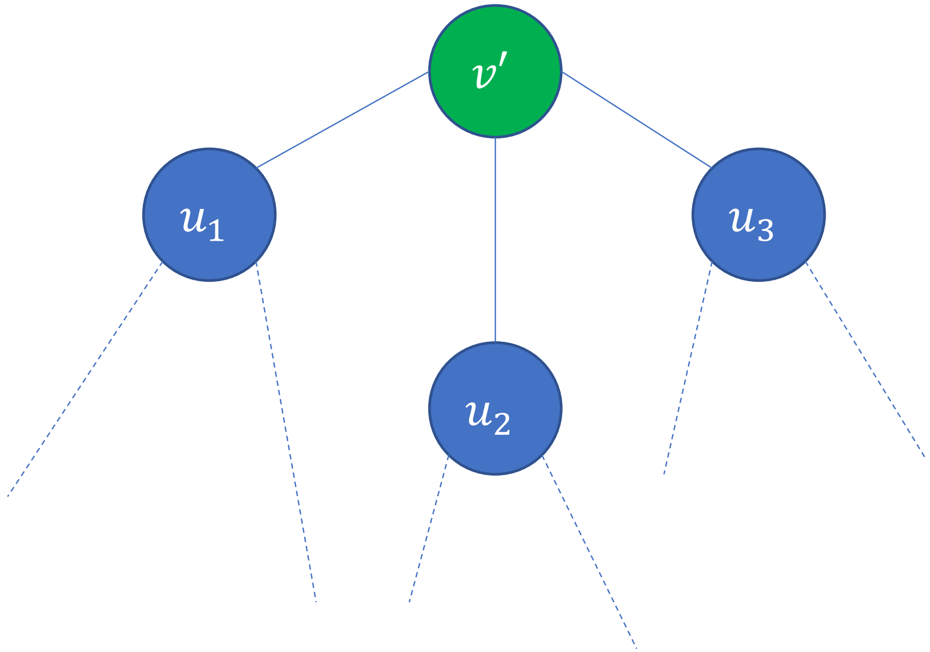
Then there is another VC of same size where v_i is rejected

Could have been rejected



$RejectFirst$ Ideas

Received advice to be accepted,



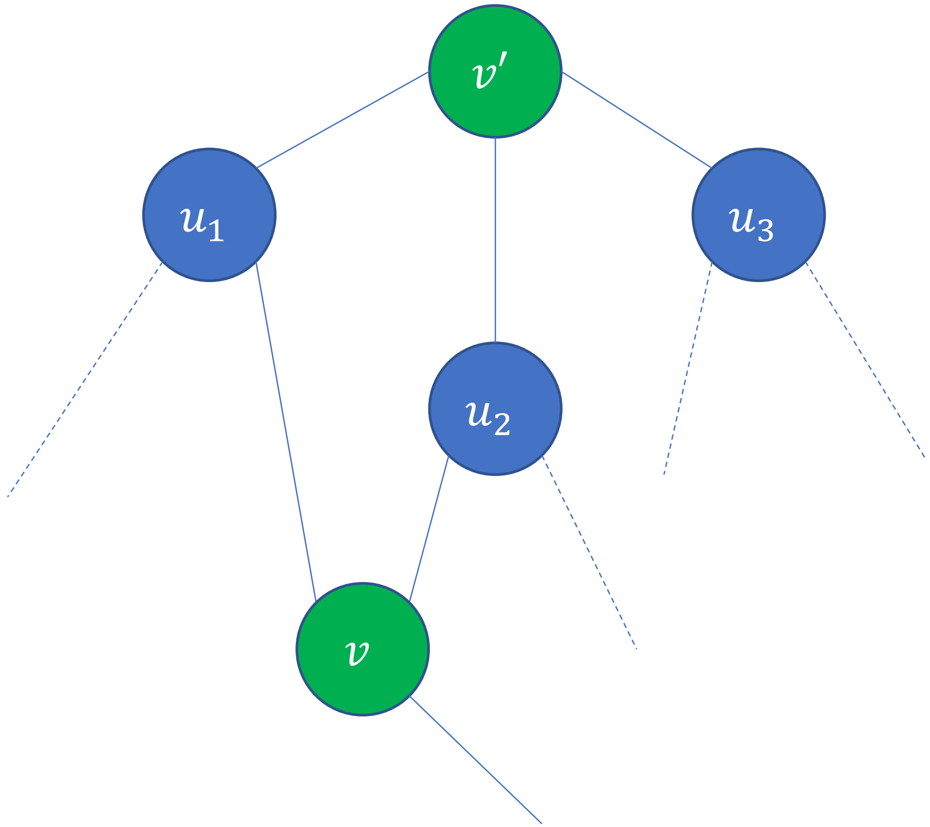
Dashed edges are not yet revealed

(*) At most 1 of u_1, u_2, u_3 can be accepted in the future

Use this to avoid giving advice to other degree 3 vertices
If there exists v such that $cdeg(v) = 3$ and $|N(v) \cap \{u_1, u_2, u_3\}| \geq 2$ then accept v
Why?
Property (*) guarantees v has a rejected neighbor

$RejectFirst$ Ideas

Received advice to be accepted,



(*) At most 1 of u_1, u_2, u_3 can be accepted in the future

Use this to avoid giving advice to other degree 3 vertices
If there exists v such that $cdeg(v) = 3$ and $|N(v) \cap \{u_1, u_2, u_3\}| \geq 2$ then accept v
Why?
Property (*) guarantees v has a rejected neighbor

$RejectFirst$ Priority Function

Define the priority function P as follows (listed in order from highest to lowest priorities):

P1: nodes with a rejected neighbor;
highest priority is given to those nodes whose neighbor was most recently rejected.

P2: nodes with current degree 0.

P3: nodes with current degree 1;
highest priority is given to those nodes with a most recently processed neighbor; among those, highest priority is given to those nodes that had two neighbors that became aa-vertices.

P4: nodes with current degree 2 that had a third neighbor in common with a previously rejected bad-vertex.

P5: a-siblings.

P6: nodes with current degree 3 with 2 or 3 neighbors in common with a single aa-vertex that was not a bad-vertex when it received advice.

P7: nodes with current degree 3 that share neighbors with a-vertices.

P8: other nodes with current degree 3.

P9: nodes with current degree 2

$RejectFirst$ Main Loop

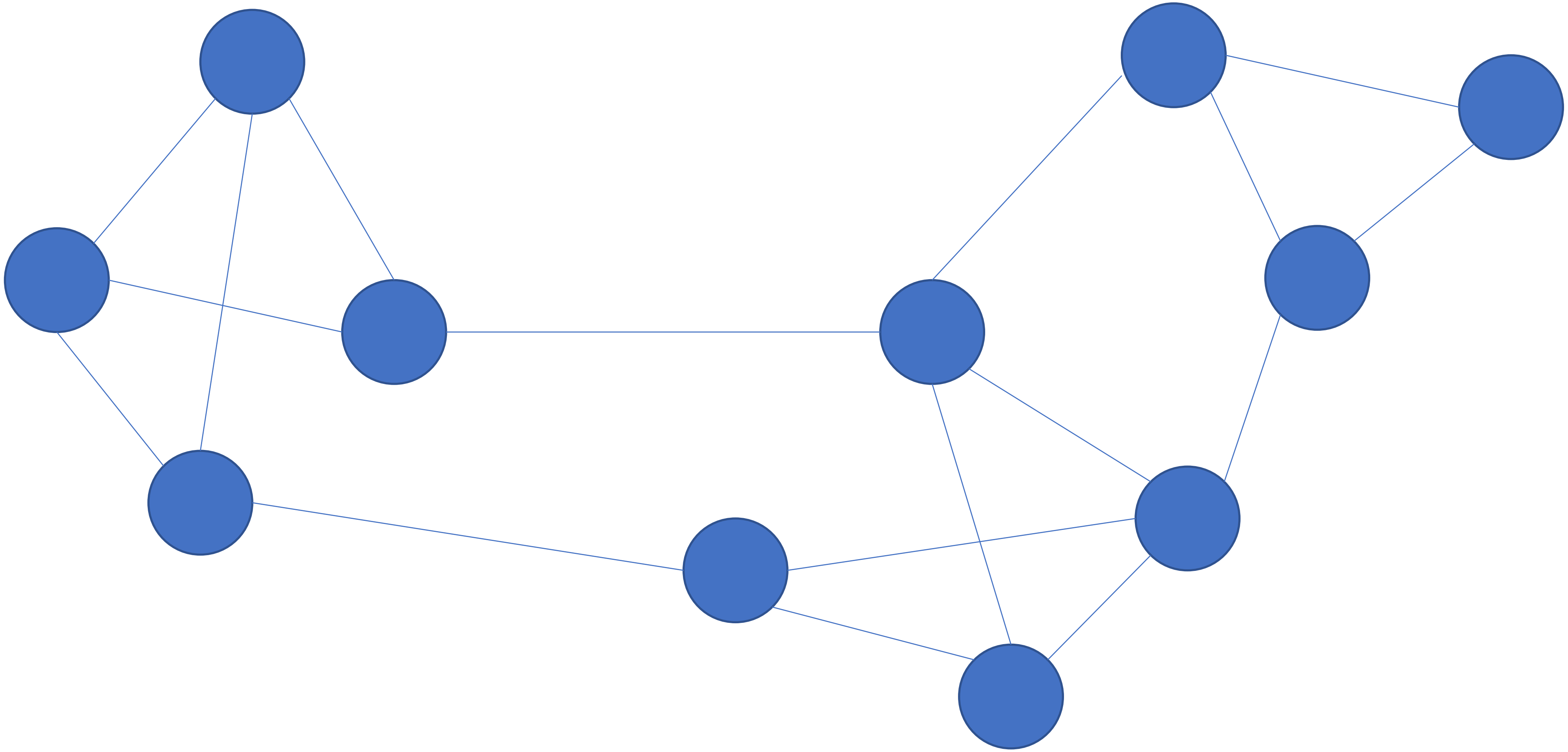
```
procedure RejectFirst
    while there exist unprocessed vertices do
        Receive the next vertex v according to P
        switch priority of v
            case P1 or P6:
                Accept v
            case P2, P3, P4, P5, or P9:
                Reject v
            case P7 or P8:
                Obtain advice to accept or reject and apply it to v
```

Key Notion in the Analysis

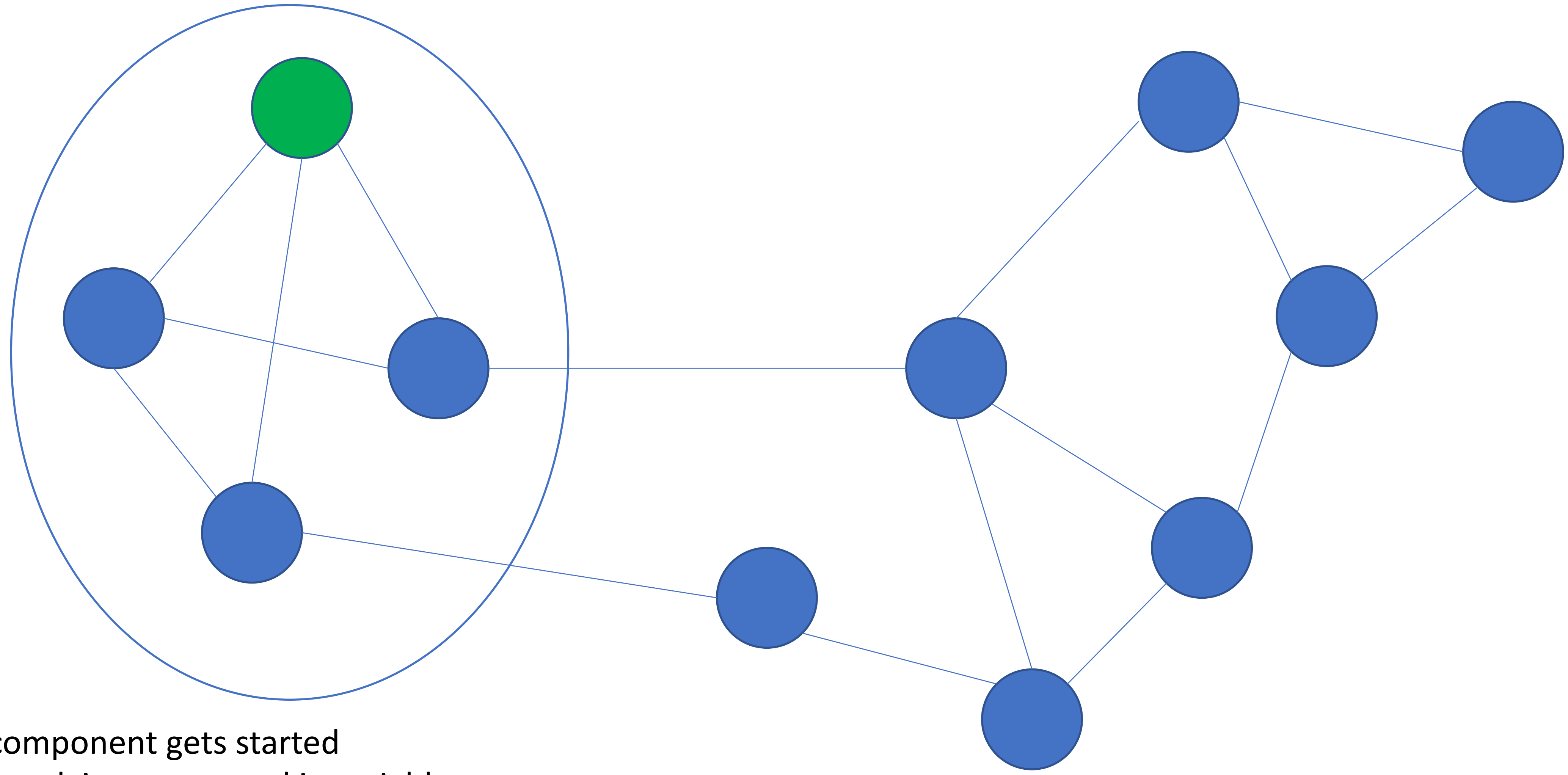
- Interactions of $cdeg = 3$ through common neighbors have effect on advice complexity.
- Component: captures related sequences of $cdeg = 3$ vertices with shared neighbors during runtime.

Example

Current degree 3
Receives advice to be accepted

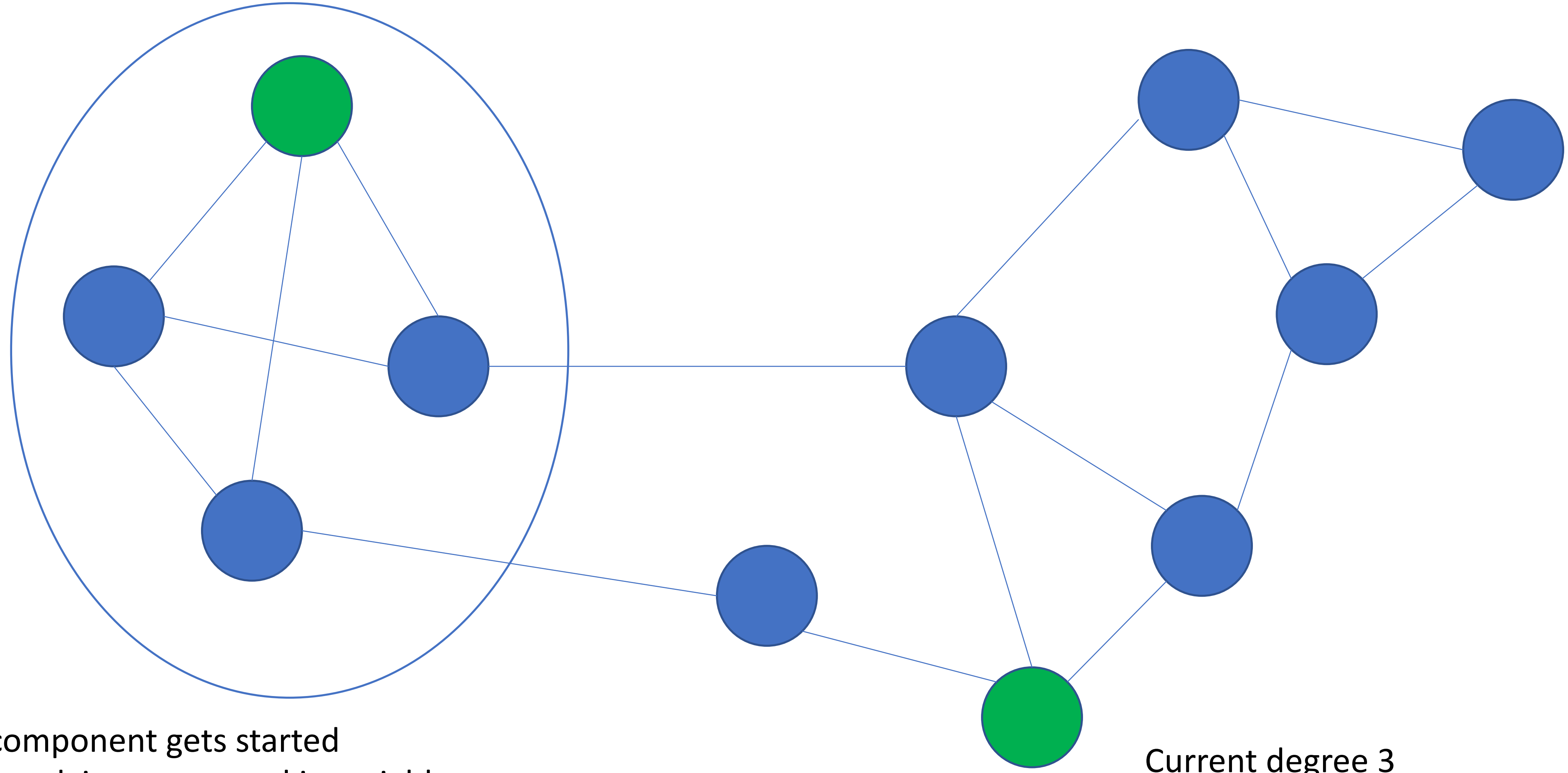


Example



New component gets started
Include advice-vertex and its neighbors

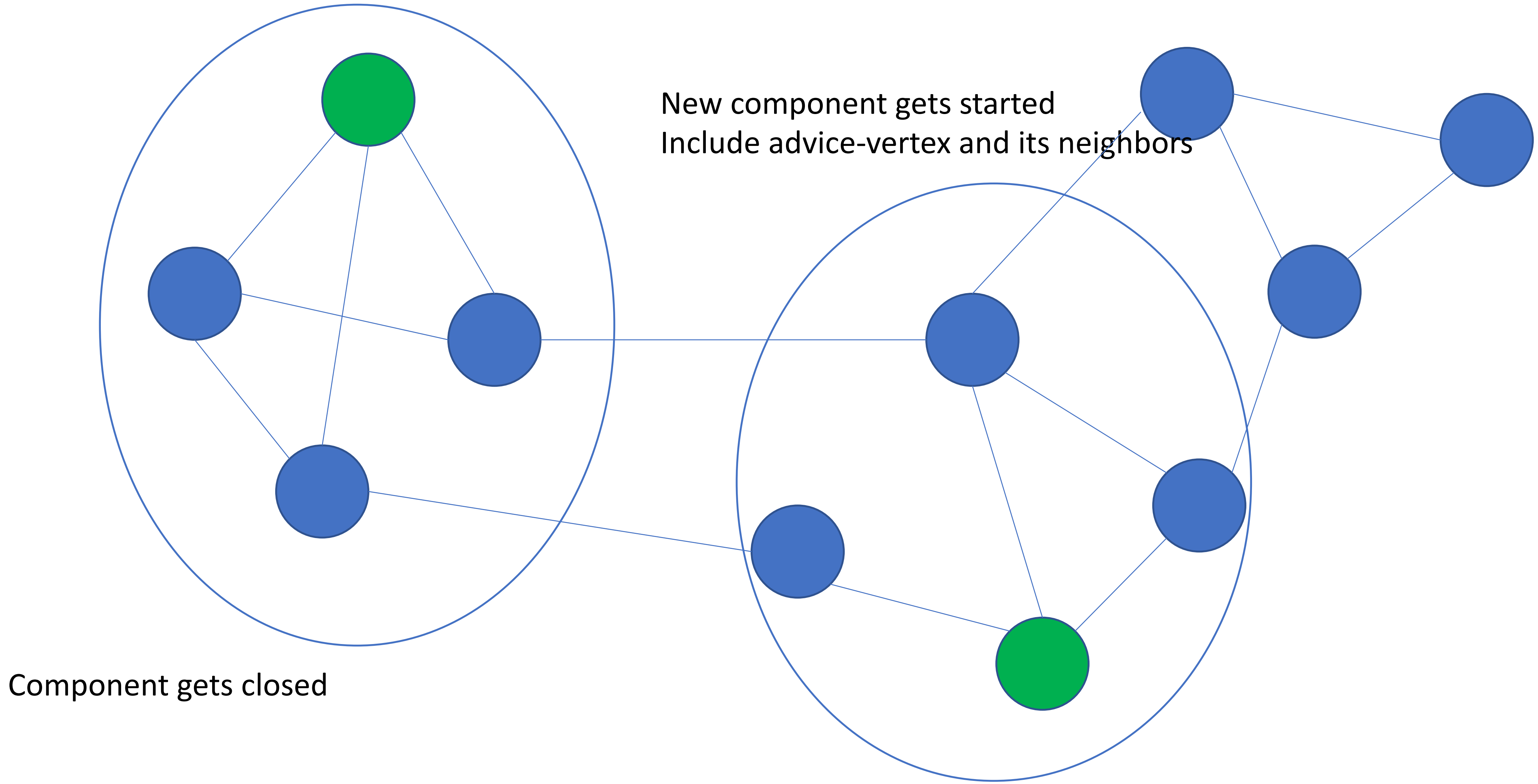
Example



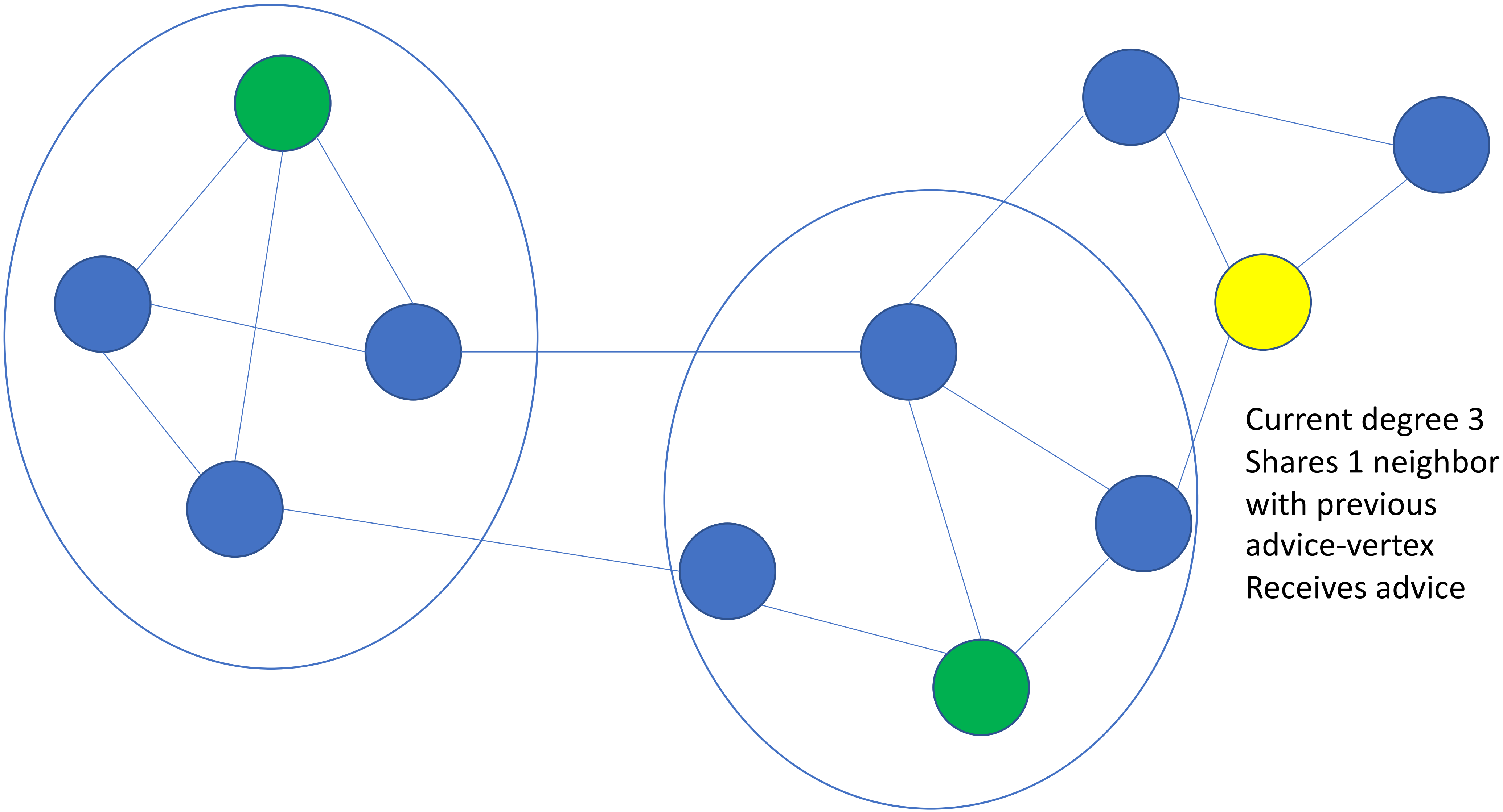
New component gets started
Include advice-vertex and its neighbors

Current degree 3
Receives advice to be accepted

Example

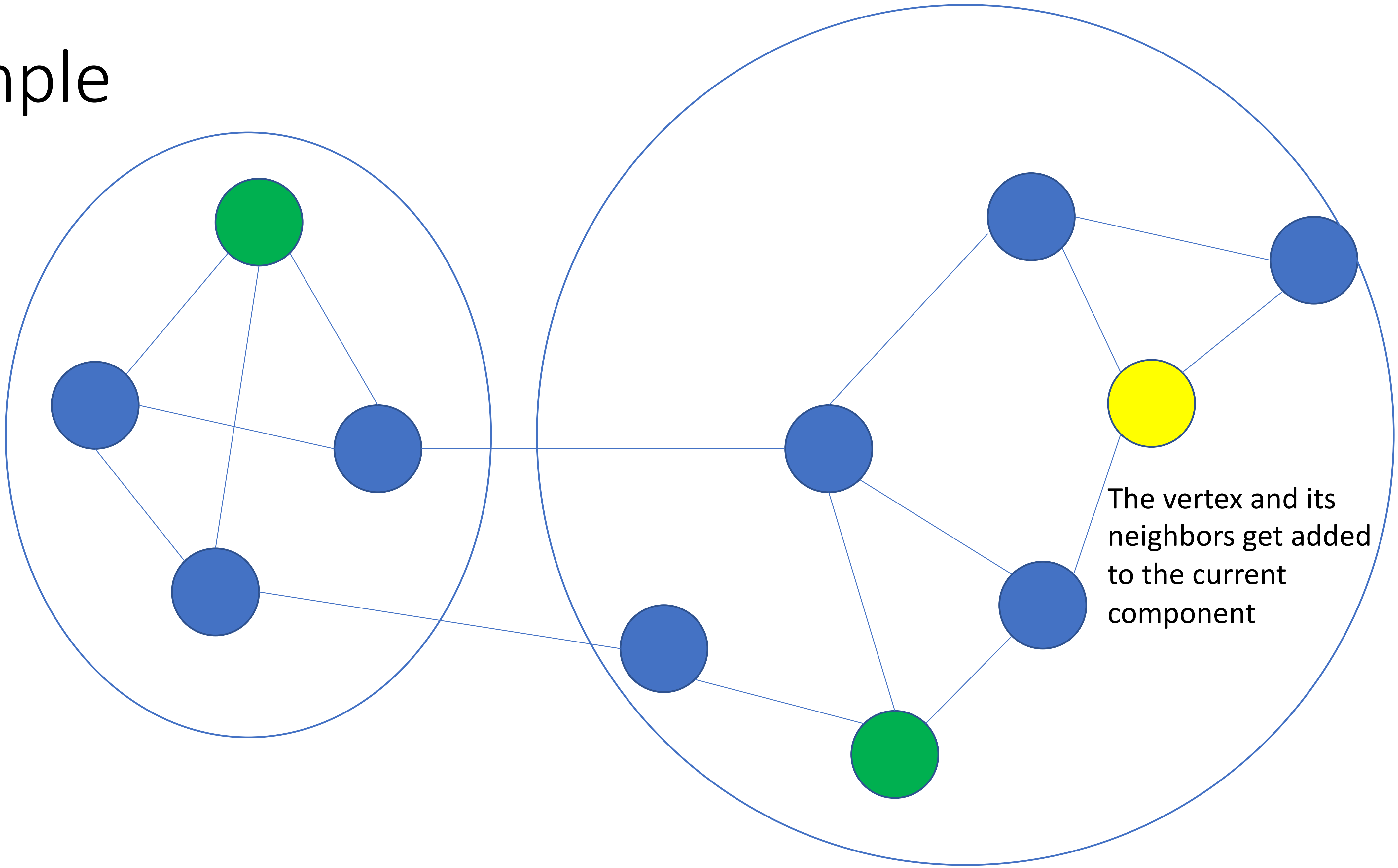

New component gets started
Include advice-vertex and its neighbors
Component gets closed

Example



Current degree 3
Shares 1 neighbor
with previous
advice-vertex
Receives advice

Example


The vertex and its neighbors get added to the current component

More notation

- For component $i \in [c]$
  - s_i - total number of vertices in component i
  - a_i - number of vertices that received advice in component i

$$n = \sum_{i=1}^{c} s_i \text{ and } a := \sum_{i=1}^{c} a_i$$

- **Lemma 1**

$$s_i \geq 3\, a_i + 1$$

- **Lemma 2**

$$10\, a - 4\, c \leq 3\, n$$

Proofs: elaborate charging arguments: vertex-based for Lemma 1 and edge-based for Lemma 2.

Proof of the Main Result Follows

- Total number of advice bits is a

$$3\,a + c \leq n \text{ (Lemma 1)}$$

$$10a\ - 4\,c \leq 3n \text{ (Lemma 2)}$$

$4 \times$ first inequality + $1 \times$ second inequality implies

$$22\,a \leq 7\,n\text{, i.e.,}$$

$$a \leq \frac{7}{22} n$$

QED.

Exact Algorithms

- Adaptive priority algorithm $\Rightarrow$ exact offline algorithm
- Enumerate all possible advice strings and run the algorithm
- Runtime is bounded by $2^{\left(\frac{7}{22}\right)n} poly(n) = O(1.2468^n)$
- Potentially could be improved by a more careful measure and conquer analysis technique
  - Generate advice recursively one bit at a time
  - Not all branches have equal depth
  - Keep track of its branch depths carefully

Thank You!