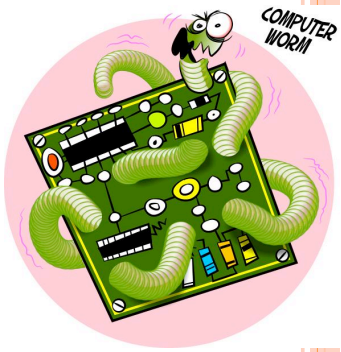
The Problem of Worm Propagation/Prevention i.e. the Minimum Vertex Cover Problem

Prof. Tiziana Calamoneri
Network Algorithms
A.y. 2015/16

The Problem

Worms (1)

- A computer worm is a standalone malware (malicious software, used or programmed by attackers to disrupt computer operation, gather sensitive information, or gain access to private computer systems) that replicates itself in order to spread to other computers.
- Often, it uses a computer network to spread itself, relying on security failures on the target computer to access it.



Worms (2)

- Unlike a computer virus, it does not need to attach itself to an existing program.
- Worms almost always cause at least some harm to the network, even if only by consuming bandwidth, whereas viruses almost always corrupt or modify files on a targeted computer.

Worms (3)

- The actual term "worm"' was first used in John Brunner's 1975 novel, *"The Shockwave Rider"*. In that novel, Nichlas Haflinger designs and sets off a data-gathering worm in an act of revenge against the powerful men who run a national electronic information web that induces mass conformity.
- One of the first worms was created by R. Morris in 1988 and called *Internet Worm*. It was able to affect between 4000 and 6000 machines, i.e. about the 4-6% of the computers connected to Internet at that time.

Worms (4)

- Worms usually exploit some sort of security hole in a piece of software or the operating system.
- They often carry payloads that do considerable damage.

  A payload is code in the worm designed to do more than spread the worm: it might delete files on a host system (e.g., the ExploreZip worm), encrypt files in a cryptoviral extortion attack, or send documents via e-mail. A very common payload for worms is to install a backdoor in the infected computer to allow the creation of a "zombie" computer under control of the worm author.

Worms (5)

- A worm tries to replicate itself in different ways:
  - E-mail: it looks for e-mail address in the infected machine; then it spontaneously generates additional e-mail messages contaning copies of itself
  - Social engineering techniques in order to induce people to open attachments containing the worm.
  - Bugs of e-mail clients, in order to auto-execute themselves, once the message is simply visualized.

Damages Caused by Worms (1)

- We can roughly divide the harmful effects caused by a worm in two types:
  - *direct damages*, resulting from the execution of the worm on the victim machine, and
  - *indirect damages*, arising from the techniques used for the diffusion.

Damages Caused by Worms (2)

Direct Damages:

- Simple worms, compound only by the instructions to replicate themselves do not create serious direct damage beyond the waste of computational resources.
- Often, however, they interfere with the software designed to find them and to counteract the spread (antivirus and firewall) thus obstructing the normal operation of the host computer.
- Very frequently a worm acts as a vehicle for automatic installation of backdoors or keyloggers, which can then be exploited by an attacker or another worm.
- They may also open TCP ports to create networks security holes for other applications.

Damages Caused by Worms (3)

Undirect Damages:

- These are the side effects of infection by a worm of a large number of computers connected to the network.
- The e-mail messages sent by the worm to replicate increase the amount of junk e-mail, wasting valuable resources in terms of bandwidth and attention.
- The worms that exploit known vulnerabilities of some software cause desease of such programs, with consequences such as instability of the operating system and sometimes forced reboots and shutdowns.

Worm Propagation (1)

- To simplify, assume that the time of transmission of information in any given connection in the network is the same, equal to T.
- If a worm has successfully infected a number of nodes such that with a single step of spread all nodes can be infected, in time T the entire network is infected ("first propagation step").
- To know the set of nodes to start is the first step to protect your network from attack.

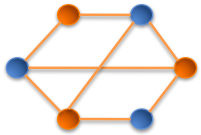
Worm Propagation (2)

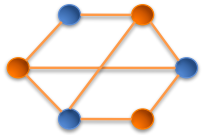
- The real problem is more complex because all the networks of considerable size have dynamic connections.
- From the point of view of the manager of the network, each filter to protect the network against attacks from worms of the first order slows down the communication and therefore it is necessary to minimize the number.
- The computers to be protected are those that could be in the first set.

Graph Model of the Problem

The Graph Model (1)

- Let $G=(V,E)$ be an undirected graph. The *vertex cover* is a subset V' of the nodes of the graph which contains at least one of the two endpoints of each edge.
- It is relevant to find the minimum vertex cover, i.e. the set V' of minimum cardinality.
- Obs. The minimum vertex cover is not unique:





The Graph Model (2)

- Intuitively, every minimum vertex cover represents an excellent starting point for a worm.
- The computers to be protected are those that represent the nodes in the minimum vertex cover of the communication graph.
- If the graph has more than one minimum vertex cover, the computers in the intersection of all the covers need to be protected.

The Minimum Vertex Cover

Minimum Vertex Cover (1)

- Def. Given $G=(V,E)$, V' subset of V is a vertex cover for G if $\forall\{a,b\}\in E$, $a\in V'$ or $b\in V'$.
- Obs. Set V is trivially a vertex cover.
- Given $G=(V,E)$, the minimum Vertex Cover Problem is to find a vertex cover for G of minimum cardinality.
- Obs. There are 2^n possible subsets to check.

Minimum Vertex Cover (2)

- Def. The Decisional version of the Minimum Vertex Cover Problem (VC) is to answer to the following question:

  given a graph G and an integer value k, is there a vertex cover for G of cardinality less than or equal to k?
- VC is an NP-complete problem (reduction from 3-SAT or from Clique [Karp]).
- VC is still NP-complete on cubic graphs [Garey, Johnson, Stockmeyer '74] and on planar graphs having degree at most 3 [Garey & Johnson '77].

Minimum Vertex Cover (3)

We introduce the follwing n decision variables:

for each $i=1, 2, \ldots, n$, $x_i=1$ if the node i belongs to V' and $x_i=0$ otherwise.

ILP formulation for VC:

$$\min \sum_{i=1}^{n} x_i$$

subject to constraints:

$$x_i + x_j \geq 1, \forall (i,j) \in E$$

$$x_i \in \{0,1\}, i = 1,2,\ldots,n$$

Note. Solving an ILP is NP-complete.

Minimum Vertex Cover (4)

- As already highlighted, VC is an NP-hard problem, so only superpolynomial agorithms are known.
- It is possible to approximate the solution in polynomial time.
- In the following we will describe two *naive* algorithms that seem intuitively good but have, instead, bad approximation ratios.
- Then, we will describe a $O(n+m)$ time approximate algorithm that finds a vertex cover V' such that $|V'| \leq 2|V^*|$, where V^* is an optimal solution.
- Finally, we propose another 2-approximate algorithm exploiting the ILP formulation.

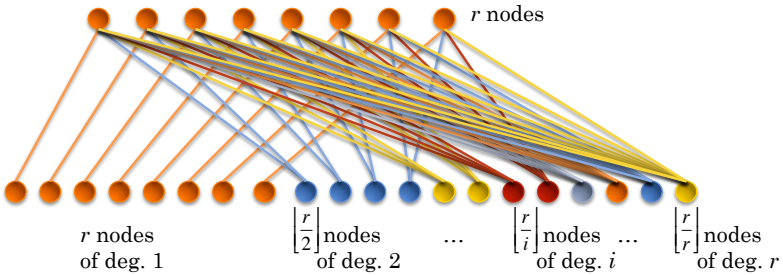
Minimum Vertex Cover (5)

Algorithm Greedy-VC(G)

- V'=empty set, $E'=E$
- While (E' is not empty) do
  - Select from E' an edge (i,j) and choose one of its endpoints i
  - Add i to V'
  - delete from E' all edges having i as an endpoint
- Return V'

Minimum Vertex Cover (6)

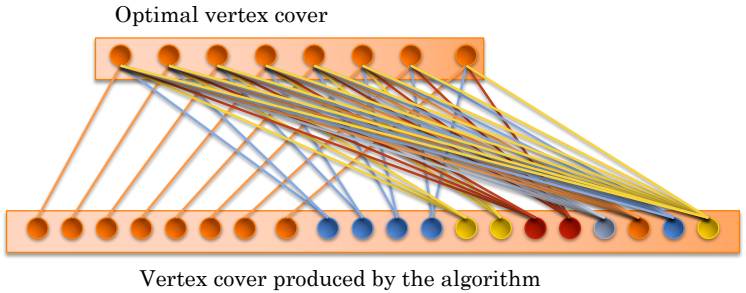
Unfortunately, Greedy-VC could produce a vertex cover whose cardinality is very far from optimal:



$$n = r + \sum_{i=1}^{r} \left\lfloor \frac{r}{i} \right\rfloor \le r + r\sum_{i=1}^{r} \frac{1}{i} = \Theta(r \log r)$$

Armonic sum; can be approximated by $ln\ r$

Minimum Vertex Cover (7)



Approximation ratio: $\Theta(\log r)$

Problem:
the algorithm could prefer small degree nodes instead of large degree nodes

Minimum Vertex Cover (8)

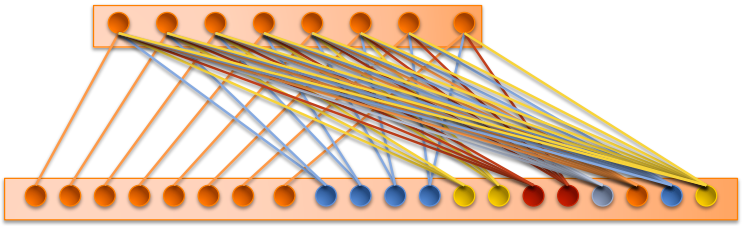
Let us try another approach:

Algorithm Greedy2-VC(G)

- V'=empty set, $E'=E$
- While (E' is not empty) do
  - Select a node v having max degree in the *current* graph
  - Add v to V'
  - Delete from E' all the edges having v as an endpoint
- Return V'

Minimum Vertex Cover (9)

This second algorithm may produce this vertex cover…



…but even this, starting from the nodes to the right

So, the approximation ratio does not change!

Minimum Vertex Cover (10)

A better algorithm:

Algorithm 2-Approx-VC(G)

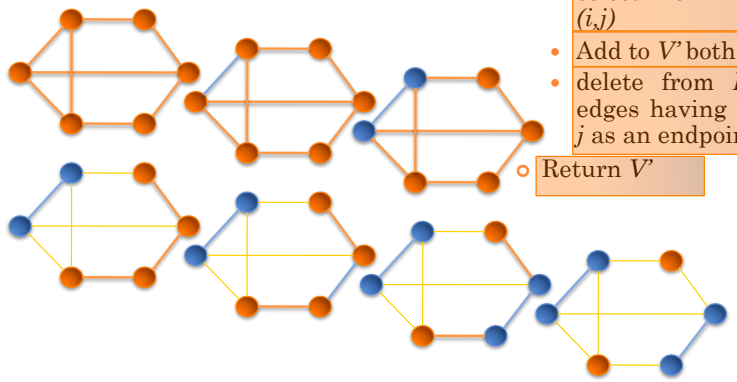
- V'=empty set
- $E'=E$
- While (E' is not empty) do
  - select from E' an edge (i,j)
  - Add to V' both i and j
  - Delete from E' all the edges having either i or j as an endpoint
- Return V'

Time complexity: $O(n+m)$

Minimum Vertex Cover (11)

Algorithm 2-Approx-VC(G)

- V'=empty set
- $E'=E$
- While (E' is not empty) do
  - select from E' an edge (i,j)
  - Add to V' both i and j
  - delete from E' all the edges having either i or j as an endpoint
- Return V'



Minimum Vertex Cover (12)

Th. Let V^* be a minimum vertex cover. The set V' returned by 2-Approx-VC is a vertex cover such that $|V'| \leq 2|V^*|$.

Proof. By construction, V' is a vertex cover.

Let A be the set of the edges selected from E'.

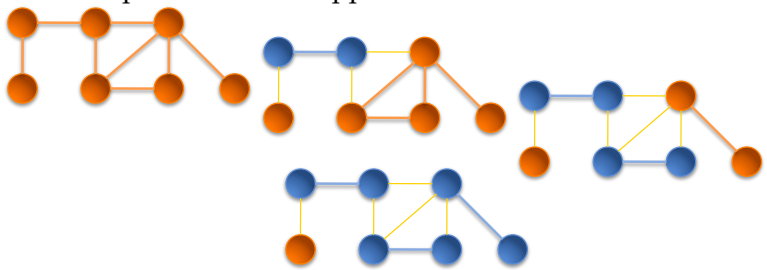
For each edge (i,j) in A, i and j are added to V' so: $|V'|=2|A|$.

Moreover, all the edges having either i or j as an endpoint are deleted from E', so edges in A cannot be incident and must be covered by any optimal solution, i.e. $|A| \leq |V^*|$.

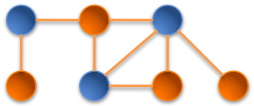
Putting together: $|V'|=2|A| \leq 2|V^*|$. *QED*

Minimum Vertex Cover (13)

An example where the upper bound is reached:



(optimal solution:)



Minimum Vertex Cover (14)

An algorithm based on the ILP formulation:

Algorithm 2-Approx2-VC(G)

- V'=empty set
- Relax the ILP formulation by eliminating the constraint that x_i must be integer.
- Invoke a polynomial time LP solver to get a solution $x_1, \ldots, x_n$
- For $i=1$ to n do
  - If $x_i \geq ½$ then
    - Add to V' node i
- Return V'

Time complexity: $O(n+m)$

Minimum Vertex Cover (15)

Th. The node set V' returned by 2-Approx2-VC is a vertex cover.

Proof. We know from our constraints that for each edge (i,j), $x_i+x_j \geq 1$. Therefore, at least one of x_i or $x_j \geq ½$ and so at least one of the nodes i,j from the edge (i,j) must belong to V'. *QED*

Minimum Vertex Cover (16)

Th. Let V^* be a minimum vertex cover. The vertex cover V' returned by 2-Approx2-VC is such that:

$|V'| \leq 2|V^*|$.

Proof. Let $Z^*=x_1+\ldots+x_n$ the "cost" of the optimal solution. (This is the sum of real numbers and not the size of any set.)

Since $x_1, \ldots, x_n$ is optimal for the LP, $Z^* \leq |V^*|$.

Let $x'_1, \ldots, x'_n$ the binary solution obtained from $x_1, \ldots, x_n$.

Of course, $x'_i \leq 2x_i$ for each $i=1, \ldots, n$, so

$|V'|=x'_1+\ldots+x'_n \leq 2(x_1+\ldots+x_n)=2Z^* \leq 2|V^*|$. *QED*

Minimum Vertex Cover (17)

- Even if these two latter algorithms are very easy, it is impossible to do much better, indeed:
- VC is not approximable in less than 1.1666 [Håstad '97] and then in less than 1.3606 [Dinur & Safra '05]
- The best known approximation ratios are:

$2 - \frac{\log\log|V|}{2\log|V|}$ [Monien& Speckenmeyer '85]

$2 - \frac{\log\log|V|}{\log|V|}$ [Bar-Yehuda, Even '85]

$2 - \frac{\ln\ln|V|}{\ln|V|}(1-o(1))$ [Halperin '00]

$2 - \Theta\left(\frac{1}{\sqrt{\log|V|}}\right)$ [Karakostas '04]

Properties of the Min Vertex Cover (1)

An independent set of $G=(V,E)$ is a set of nodes of V, no two of which are adjacent.

Th. A set of nodes V' is a vertex cover if and only if its complement $V-V'$ is an independent set.

Proof.

V' VC $\Rightarrow V-V'$ IS

- By contradiction. If in $V-V'$ there exist two adjacent nodes, then the corresponding edge is not covered. A contradiction.

$V-V'$ IS $\Rightarrow V'$ VC

- By contradiction. If there exists an edge e that is not covered by any node in V', the nodes incident to e are adjacent in $V-V'$. A contradiction. QED

Properties of the Min Vertex Cover (2)

- Cor. The number of nodes of a graph is equal to the size of its min vertex cover plus the size of a maximum independent set [Gallai '59]
- Nevertheless, these two problems are not equivalent, from an approximation point of view: IS cannot be approximated by any constant [Håstad '99].

Properties of the Min Vertex Cover (3)

Def. A matching of $G=(V,E)$ is a subset M of E without common nodes.

Th. Let M be a matching of G and C a vertex cover for G. Then $|M| \leq |C|$.

Proof. C is a vertex cover, so it must cover all edges in M.

From the other side, by definition of matching, for each edge in M, at least one of its endpoints must be in C.

So $|M| \leq |C|$. QED

Properties of the Min Vertex Cover (4)

Cor. Let M be a matching of G and C a vertex cover for G. If $|M|=|C|$ then M is a maximum matching and C is a minimum vertex cover.

It is polynomial to compute a max matching.

Could we think to solve the min vertex cover passing through the max matching problem?

No, because:

Fact: The reverse of the previous corollary is false.

Anyway, an algorithm based on this property has been proposed: …

Properties of the Min Vertex Cover (5)

Algorithm New-Approx-VC(G) [Gavril '79]

- Compute a max matching M
- V'= empty set
- For each e in M
  Insert in V' both the endpoints of e
- Return V'

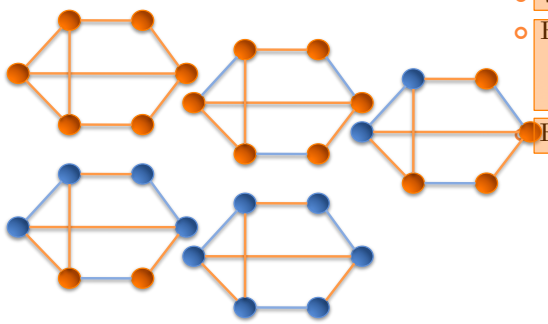
Time complexity:

It depends on the computation of the max matching:

$O(n^4)$ [Edmonds '65]

$O(m\sqrt{n})$ [Micali & Vazirani '80]

Properties of the Min Vertex Cover (6)



Algorithm New-Approx-VC(G) [Gavril '79]

- Compute a maximum matching M
- V'= empty set
- For each e in M
  Insert in V' both the endpoints of e
- Return V'

Properties of the Min Vertex Cover (7)

Th. The set V' returned by New-Approx-VC is a vertex cover for G such that $|V'|\leq 2|V^*|$.

Dim. V' is a vertex cover indeed an edge (u,v) in G is:

- either in M and hence both its endpoints are in V'
- or is in $E\setminus M$, and at least one of its endpoint is in V', otherwise it could be added to M, that is maximum.

By construction $|V'|=2|M|$.

Notice that each $(u,v)\in M$ must have at least one of its endpoints in any min. vertex cover: $|M|\leq|V^*|$

Putting together the two relations: $|V'|=2|M|\leq 2|V^*|$.

QED

Properties of the Min Vertex Cover (8)

If G is bipartite, a stronger relation holds between min vertex cover and max matching, so deducing that:

VC is polynomially solvable on bipartite graphs.

In particular, the previous algorithm can be modified in order to produce an optimal solution (time complexity: at least $O(m\sqrt{n})$ – it depends on the computation of the maximum matching).

König's Th. ['31] (Egervàry ['31]): In any bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum vertex cover.

Proof 1. (the most common) exploits the th. by P. Hall on the maximum matchings in bipartite graphs (to be studied later).

Properties of the Min Vertex Cover (8)

König's Th. ['31] (Egervàry ['31]): In any bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum vertex cover.

Proof 2. Suppose that $G=(X \cup Y, E)$ is bipartite. Let M be a matching for G.

By contradiction, $|M|$ and $|C|$ are different, so we must show that:

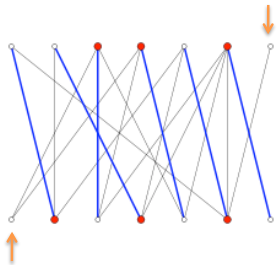
- or M is not a max matching
- or C is not a min vertex cover and there exists a cover of size $|M|$.

Case 1. If M is a perfect matching, every edge is incident to exactly one node on either side, so any partition of G is a vertex cover of size $|M|$ and we are done.

Properties of the Min Vertex Cover (9)

Proof 2 König's Th, (cnt.d)

Case 2. Otherwise:



There are some nodes out of M.

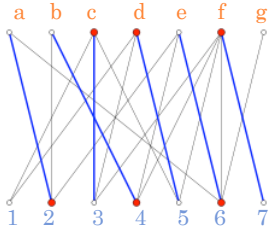
Partition the nodes of G into subsets S_i:

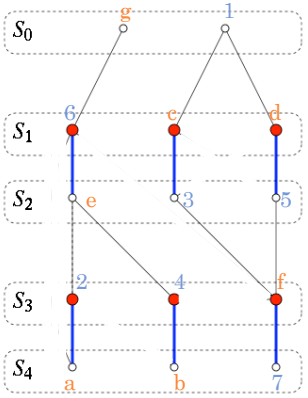
- Let S_0 consist of all nodes unmatched by M (in any side of the partition).
- Assume S_{2j} has already been defined for some $j \geq 0$. Let S_{2j+1} be the set of nodes that are adjacent to nodes in S_{2j} via some edge in $E \setminus M$ and have not been included in any previous set.
- …

Properties of the Min Vertex Cover (10)

Proof 2 of the König's th. (cnt.d)

- Each node v in S_{2j+1} must be adjacent to another node u via an edge in M (otherwise v is unmatched by M and would have been placed in S_0); if u has not been included yet in a set, insert u in S_{2j+2}.

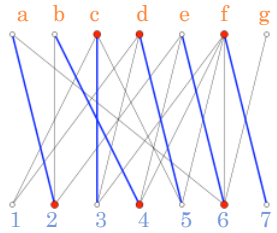


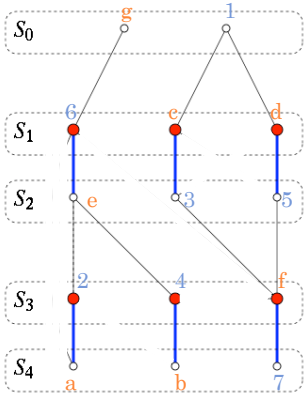


Properties of the Min Vertex Cover (11)

Proof 2 of the König's th. (cnt.d)

- If there are no nodes adjacent to S_{2j}, arbitrarily pick an unused node and continue in S_{2j+1}.

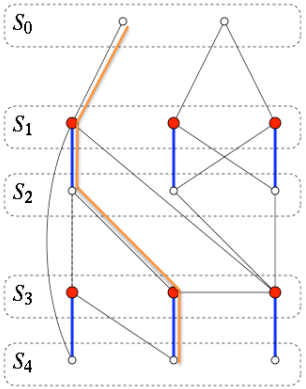




Properties of the Min Vertex Cover (12)

Proof 2 of the König's th. (cnt.d)

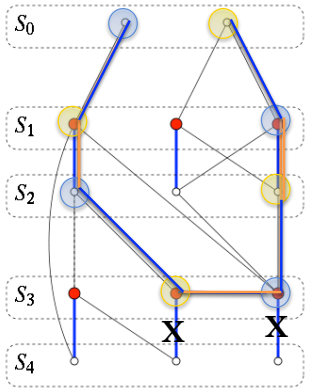
- Each node in S_i has an edge to a node in S_{i-1} => a path can be formed from a given node, going up one level at a time, ending either at an unmatched node at level S_0, or at some level S_{2j+1} containing a single (matched) node.
- This path is alternating (i.e. it alternates edges of M and edges of E\M).



Properties of the Min Vertex Cover (13)

Proof 2 of the König's th. (cnt.d)

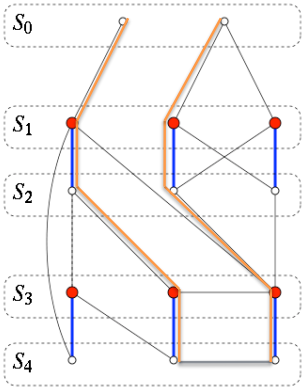
- If there exists an edge (u,v) *in* M between two nodes on the same odd level S_{2j+1}, the two alternating paths from u and v are connected by a unique alternating path passing through (u,v).
- These two paths cannot have any repeated nodes since G is bipartite, so u e v are in different partitions and the edges in M in the two paths are in different parity levels.
- It follows that the resulting path is augmenting => M is not maximum.



Properties of the Min Vertex Cover (14)

Proof 2 of the König's th. (cnt.d)

- Analogously, if there exists an edge (u,v) not in M connecting two nodes on the same even level S_{2j}, with similar reasonings we get an augmenting path => M is not maximum.



Properties of the Min Vertex Cover (15)

Proof 2 of the König's th. (cnt.d)

Thus, if M is maximum:

- Each edge in M has a single endpoint in an odd level set S_{2j+1}
- Each edge not in M has at least one endpoint in an odd level set S_{2j+1}
- The union of the odd level sets forms a vertex cover V' of size $|M|$
- Since no smaller set of nodes could cover M, it must be a minimum vertex cover, i.e. $|V'|=|M|$.

QED