Subgraphs

Stepan Kuznetsov

Computer Science Department, Higher School of Economics

Outline

Cliques, Independent Sets

Vertex Covers

Approximating Optimal Vertex Cover

Subgraphs

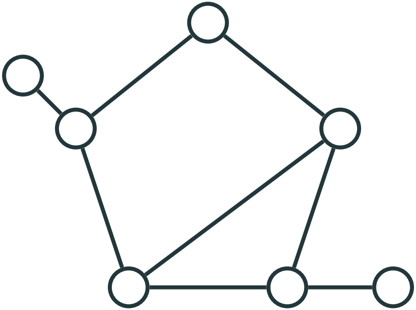
- A *subgraph* is a part of a graph which is obtained by taking a subset of vertices and a subset of edges.

Subgraphs

- A *subgraph* is a part of a graph which is obtained by taking a subset of vertices and a subset of edges.
- The vertex subset should cover the edge subset.

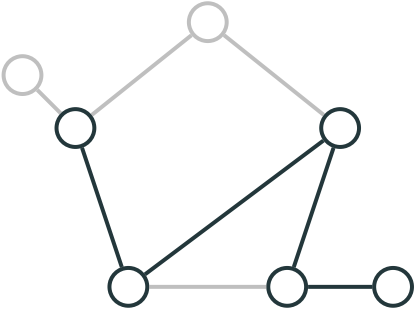
Subgraphs

- A *subgraph* is a part of a graph which is obtained by taking a subset of vertices and a subset of edges.
- The vertex subset should cover the edge subset.



Subgraphs

- A *subgraph* is a part of a graph which is obtained by taking a subset of vertices and a subset of edges.
- The vertex subset should cover the edge subset.



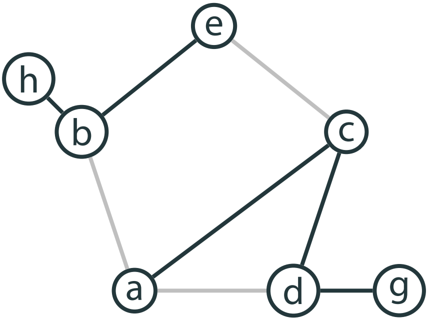
Subgraphs

- An *induced* subgraph includes all the edges of the original graph, whose endpoints are in the vertex subset.

Subgraphs

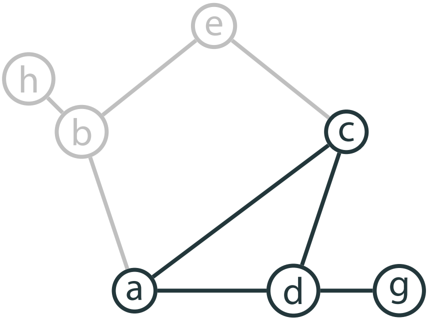
- An *induced* subgraph includes all the edges of the original graph, whose endpoints are in the vertex subset.
- A *spanning* subgraph includes all vertices of the original graph (but maybe not all edges).

Subgraphs



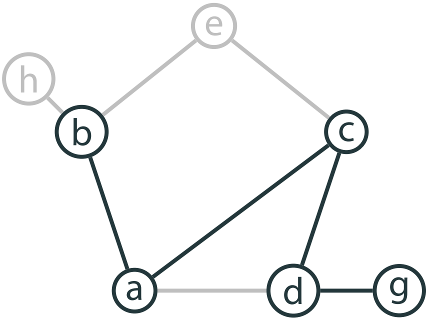
spanning

Subgraphs



induced

Subgraphs



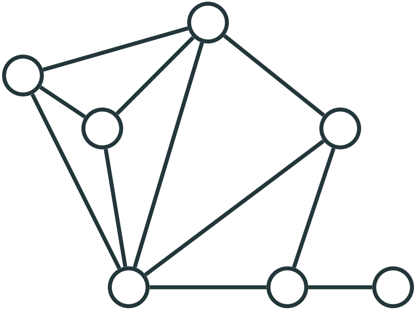
neither

Special Subgraphs

- A *clique* is a complete subgraph.

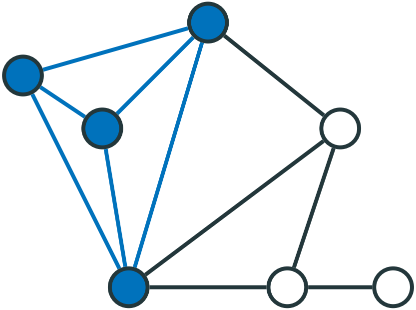
Special Subgraphs

- A *clique* is a complete subgraph.
- Example of a clique on 4 vertices:



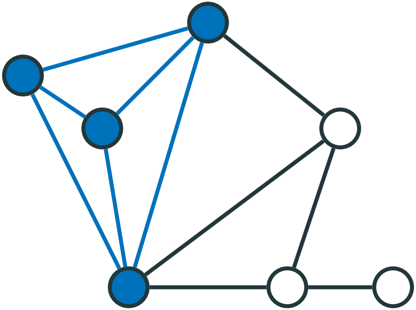
Special Subgraphs

- A *clique* is a complete subgraph.
- Example of a clique on 4 vertices:



Special Subgraphs

- A *clique* is a complete subgraph.
- Example of a clique on 4 vertices:



- Clique in social network graph = group of users who are friends with each other.

Independent Sets

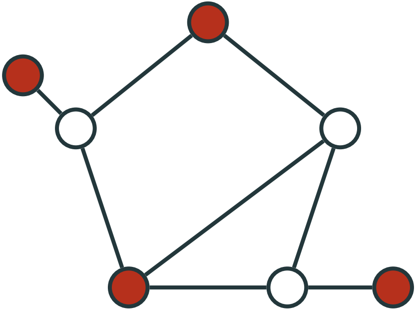
- An *independent set* is an empty induced subgraph.

Independent Sets

- An *independent set* is an empty induced subgraph.
- No pair of vertices from an independent set could be connected.

Independent Sets

- An *independent set* is an empty induced subgraph.
- No pair of vertices from an independent set could be connected.
- Example: independent set of 4 vertices.



Duality

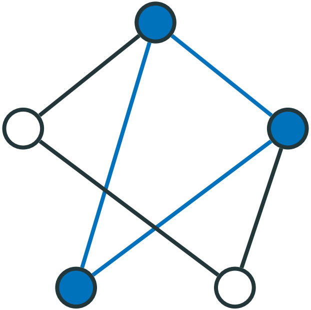
Fact

Cliques in a graph are exactly independent sets in the complement graph.

Duality

Fact

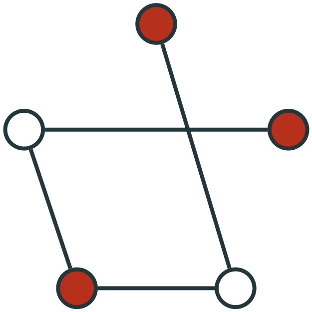
Cliques in a graph are exactly independent sets in the complement graph.



Duality

Fact

Cliques in a graph are exactly independent sets in the complement graph.



Outline

Cliques, Independent Sets

Vertex Covers

Approximating Optimal Vertex Cover

Vertex Covers

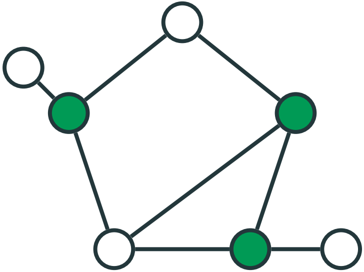
- A *vertex cover* is a subset C of vertices such that for any edge at least one its endpoint belongs to C.

Vertex Covers

- A *vertex cover* is a subset C of vertices such that for any edge at least one its endpoint belongs to C.
- A vertex cover is *optimal,* if it contains the smallest possible number of vertices.

Vertex Covers

- A *vertex cover* is a subset C of vertices such that for any edge at least one its endpoint belongs to C.
- A vertex cover is *optimal,* if it contains the smallest possible number of vertices.
- Example:



Yet Another Duality

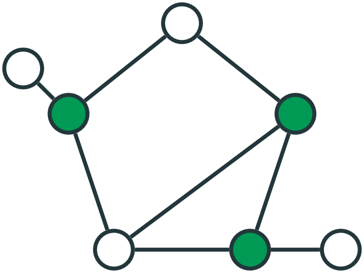
Fact

A set C of vertices is a vertex cover if and only if its complement, $V - C$, is an independent set.

Yet Another Duality

Fact

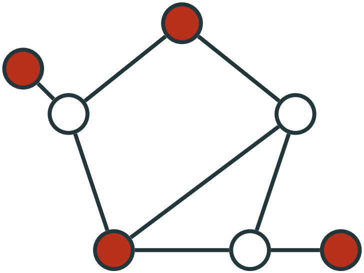
A set C of vertices is a vertex cover if and only if its complement, $V - C$, is an independent set.



Yet Another Duality

Fact

A set C of vertices is a vertex cover if and only if its complement, $V - C$, is an independent set.



Yet Another Duality

Fact

A set C of vertices is a vertex cover if and only if its complement, $V - C$, is an independent set.

Yet Another Duality

Fact

A set C of vertices is a vertex cover if and only if its complement, $V - C$, is an independent set.

- Indeed, no edge could connect two vertices outside C.

Yet Another Duality

Fact

A set C of vertices is a vertex cover if and only if its complement, $V - C$, is an independent set.

- Indeed, no edge could connect two vertices outside C.
- Notice that this is a *different* duality than that of independent sets and cliques: here we take the complement of the set, not of the graph.

Computing the Optimal Vertex Cover

- For some specific classes of graphs, the optimal (smallest) vertex cover can be efficiently computed.

Computing the Optimal Vertex Cover

- For some specific classes of graphs, the optimal (smallest) vertex cover can be efficiently computed.
- An easy example is the class of trees.

Computing the Optimal Vertex Cover

- For some specific classes of graphs, the optimal (smallest) vertex cover can be efficiently computed.
- An easy example is the class of trees.
    - Given a tree, we have to choose whether the root vertex belongs to C.

Computing the Optimal Vertex Cover

- For some specific classes of graphs, the optimal (smallest) vertex cover can be efficiently computed.
- An easy example is the class of trees.
  - Given a tree, we have to choose whether the root vertex belongs to C.
  - If yes, we recursively call our algorithm for all children subtrees.

Computing the Optimal Vertex Cover

- For some specific classes of graphs, the optimal (smallest) vertex cover can be efficiently computed.
- An easy example is the class of trees.
    - Given a tree, we have to choose whether the root vertex belongs to C.
    - If yes, we recursively call our algorithm for all children subtrees.
    - If no, we do the same, but force the roots of these subtrees to belong to C.

Computing the Optimal Vertex Cover

- For some specific classes of graphs, the optimal (smallest) vertex cover can be efficiently computed.
- An easy example is the class of trees.
  - Given a tree, we have to choose whether the root vertex belongs to C.
  - If yes, we recursively call our algorithm for all children subtrees.
  - If no, we do the same, but force the roots of these subtrees to belong to C.
  - Finally, we compare the results and yields the smaller one.

Computing the Optimal Vertex Cover

- For a more general case of bipartite graphs, there is also an efficient algorithm, based on the Kőnig – Egerváry theorem.

Computing the Optimal Vertex Cover

- For a more general case of bipartite graphs, there is also an efficient algorithm, based on the Kőnig – Egerváry theorem.
- For arbitrary graphs, however, the task of finding an optimal vertex cover is hard (not solvable in polynomial time unless P=NP).

Computing the Optimal Vertex Cover

- For a more general case of bipartite graphs, there is also an efficient algorithm, based on the Kőnig – Egerváry theorem.
- For arbitrary graphs, however, the task of finding an optimal vertex cover is hard (not solvable in polynomial time unless P=NP).
- In the next section we present an approximation: an algorithm which finds a vertex cover which is no more than twice bigger than the optimal one.

Outline

Cliques, Independent Sets

Vertex Covers

Approximating Optimal Vertex Cover

Vertex Covers vs. Matchings

- Suppose there is a set of m edges in a graph, such that none of these edges are adjacent.

Vertex Covers vs. Matchings

- Suppose there is a set of m edges in a graph, such that none of these edges are adjacent.
- Such a set is called a *matching*.

Vertex Covers vs. Matchings

- Suppose there is a set of m edges in a graph, such that none of these edges are adjacent.
- Such a set is called a *matching*.
- In this case, any vertex cover should include at least m vertices.

Vertex Covers vs. Matchings

- Suppose there is a set of m edges in a graph, such that none of these edges are adjacent.
- Such a set is called a *matching*.
- In this case, any vertex cover should include at least m vertices.
- Otherwise, some of the edges in the matching would be left uncovered.

Constructing a Big Matching

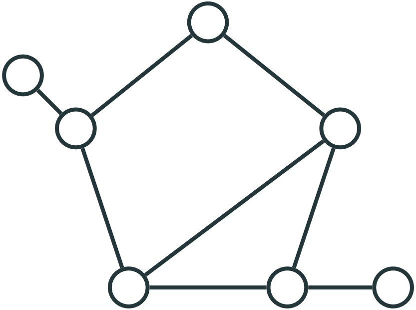
- Given a graph, we construct a matching *greedily*.

Constructing a Big Matching

- Given a graph, we construct a matching *greedily*.
- At each step, we pick a new edge and remove edges which are adjacent to it.

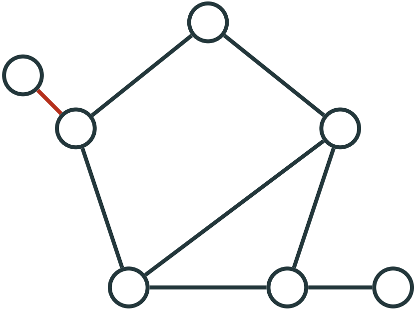
Constructing a Big Matching

- Given a graph, we construct a matching *greedily.*
- At each step, we pick a new edge and remove edges which are adjacent to it.
- Example:



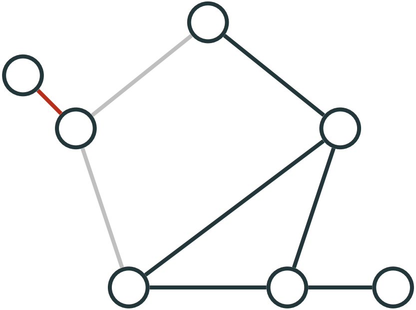
Constructing a Big Matching

- Given a graph, we construct a matching *greedily*.
- At each step, we pick a new edge and remove edges which are adjacent to it.
- Example:



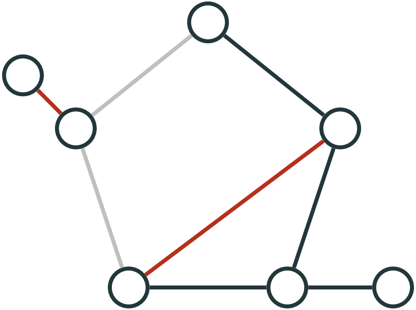
Constructing a Big Matching

- Given a graph, we construct a matching *greedily*.
- At each step, we pick a new edge and remove edges which are adjacent to it.
- Example:



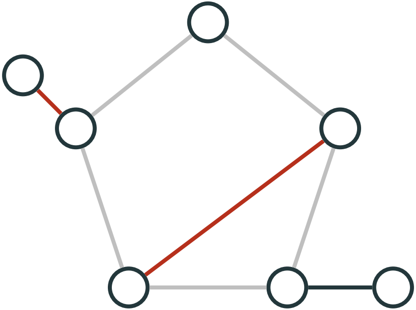
Constructing a Big Matching

- Given a graph, we construct a matching *greedily*.
- At each step, we pick a new edge and remove edges which are adjacent to it.
- Example:



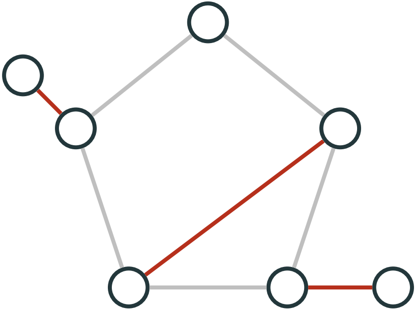
Constructing a Big Matching

- Given a graph, we construct a matching *greedily*.
- At each step, we pick a new edge and remove edges which are adjacent to it.
- Example:



Constructing a Big Matching

- Given a graph, we construct a matching *greedily.*
- At each step, we pick a new edge and remove edges which are adjacent to it.
- Example:



Vertex Cover from Matching

- Once we have constructed our matching M, we take all endpoints of its edges and form a set of vertices C.

Vertex Cover from Matching

- Once we have constructed our matching M, we take all endpoints of its edges and form a set of vertices C.
- Our matching is *maximal* in the sense that no edge can be added to it.

Vertex Cover from Matching

- Once we have constructed our matching M, we take all endpoints of its edges and form a set of vertices C.
- Our matching is *maximal* in the sense that no edge can be added to it.
- Thus, every edge is adjacent to one from M.

Vertex Cover from Matching

- Once we have constructed our matching M, we take all endpoints of its edges and form a set of vertices C.
- Our matching is *maximal* in the sense that no edge can be added to it.
- Thus, every edge is adjacent to one from M.
- Thus, C is a vertex cover!

Size Estimations

- Let us estimate, how large is C.

Size Estimations

- Let us estimate, how large is C.
- Evidently $C = 2 \cdot |M|$.

Size Estimations

- Let us estimate, how large is C.
- Evidently $C = 2 \cdot |M|$.
- On the other side, for *any* vertex cover C' we have $|C'| \geq |M|$.

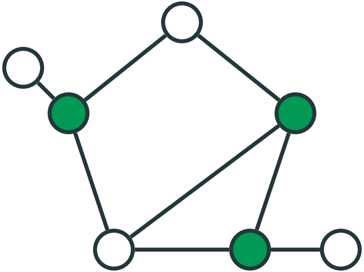
Size Estimations

- Let us estimate, how large is C.
- Evidently $C = 2 \cdot |M|$.
- On the other side, for *any* vertex cover C' we have $|C'| \geq |M|$.
- In particular, this holds for the smallest (optimal) vertex cover C_{opt}.

Size Estimations

- Let us estimate, how large is C.
- Evidently $C = 2 \cdot |M|$.
- On the other side, for *any* vertex cover C' we have $|C'| \geq |M|$.
- In particular, this holds for the smallest (optimal) vertex cover C_{opt}.
- Thus, we have constructed a vertex cover which is no more than twice bigger than the optimal one: $|C| \leq 2 \cdot |C_{\text{opt}}|$.

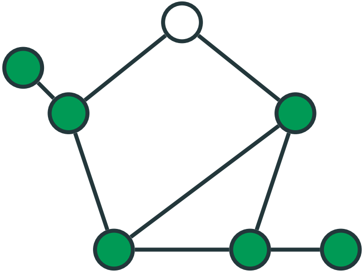
Size Estimations



the optimal vertex cover

$$|C_{\text{opt}}| = 3$$

Size Estimations



the approximation given by the algorithm

$$|C| = 6$$