Vertex Cover Problem

Exact and Approximation Algorithm

Iftekhar Hakim Kaowsar - 1705045
Apurba Saha - 1705056

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

July 7, 2021

What is the minimum number of cameras needed to cover the whole place?

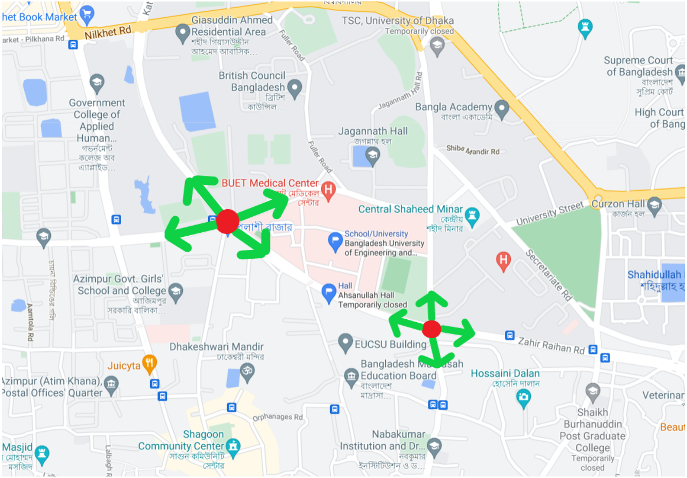
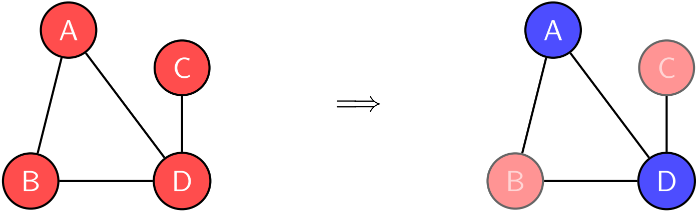


Figure: Map of BUET,Palashi

Problem Definition

- Def^n : Finding smallest subset of vertices, so that for every edge (u, v) at least one of u and v is in the subset.
- Example:



Exact Algorithm

Decision version of this problem is whether there is a vertex cover of size at most K.

Exact algorithm to solve it is quite straight-forward.

- Simply check for all possible subsets of size K.

Bruteforce!

Exact Algorithm

Decision version of this problem is whether there is a vertex cover of size at most K.

Exact algorithm to solve it is quite straight-forward.

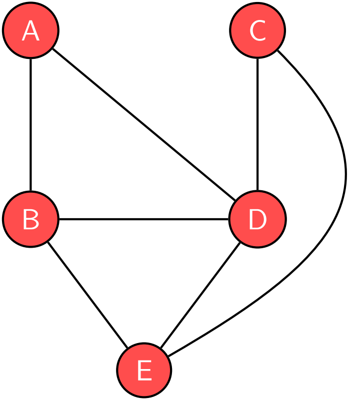
- Simply check for all possible subsets of size K.

Bruteforce!

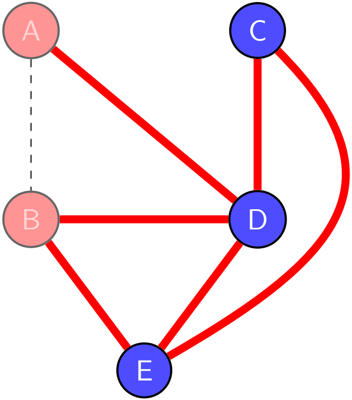
Let's try an example!

Exact Algorithm: Simulation

$K = 3$

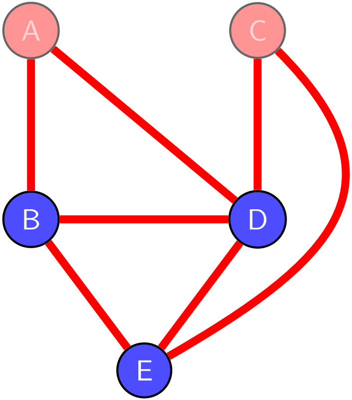


Subsets of size 3 –
$\{A, B, C\}, \{A, B, D\}, \{A, B, E\},$
$\{A, C, D\}, \{A, C, E\}, \{A, D, E\},$
$\{B, C, D\}$, ...



Check for $\{C, D, E\}$

✗ Edge (A, B)



Check for $\{B, D, E\}$

✓ **No edge left over**
✓ **Possible vetex cover**
We made our decision.

Time Complexity

- If Graph has n vertices and m edges, number of possible subsets of size k is $\binom{n}{k}$.
- Testing any subset takes $O(m)$ or $O(nk)$
- Overall time complexity $O\left(nk\binom{n}{k}\right)$ or $O(kn^{k+1})$

Too large...

Parameterized Algorithm

A little different approach.

(u, v) is any edge of Graph G.

- Decision for (G, k) is *yes*, if and only if decision for $(G - u, k - 1)$ or $(G - v, k - 1)$ is *yes*.
- If $k = 0$, decision is *yes* when there is no edge.

Parameterized Algorithm

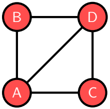
A little different approach.

(u, v) is any edge of Graph G.

- Decision for (G, k) is *yes*, if and only if decision for $(G - u, k - 1)$ or $(G - v, k - 1)$ is *yes*.
- If $k = 0$, decision is *yes* when there is no edge.

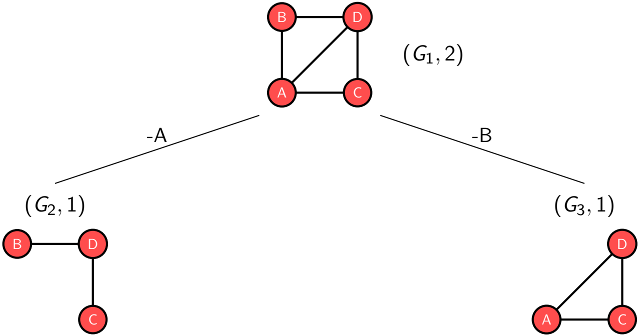
Let's try an example!

Parameterized Algorithm: Simulation

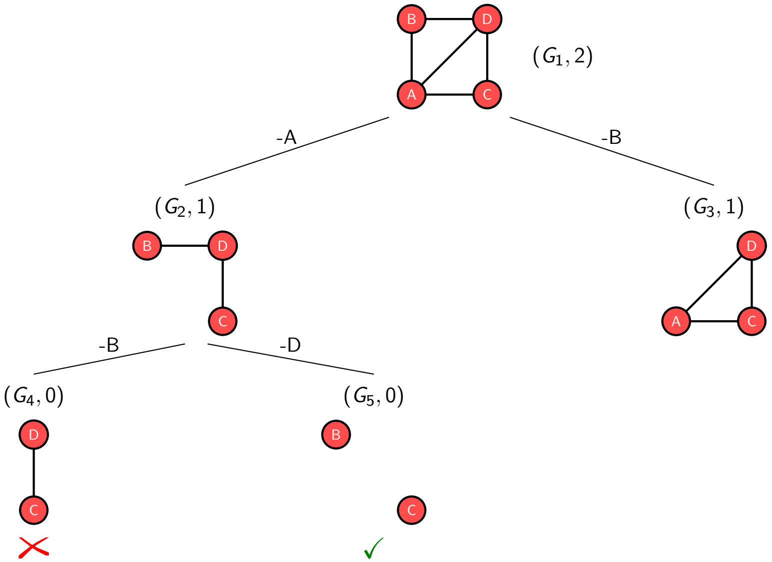


$(G_1, 2)$

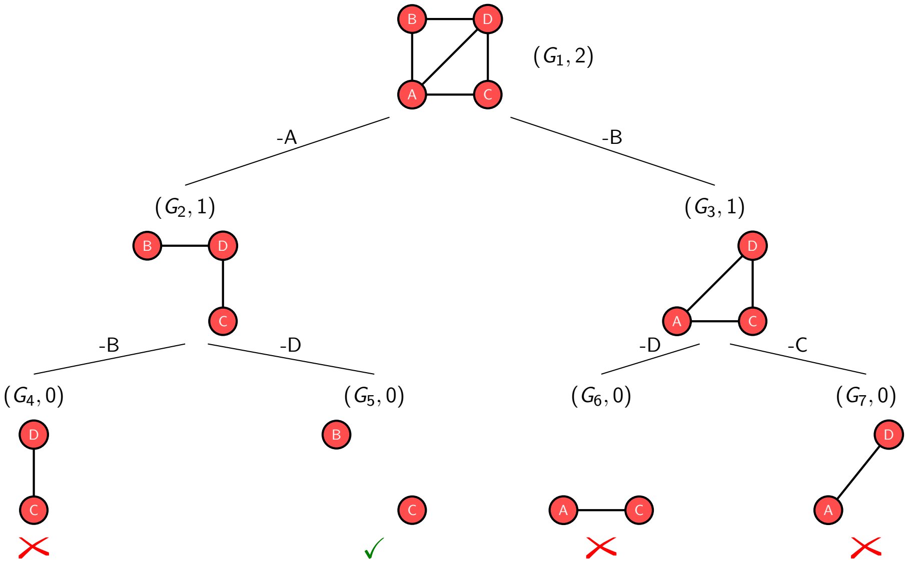
Parameterized Algorithm: Simulation



Parameterized Algorithm: Simulation



Parameterized Algorithm: Simulation



Parameterized Algorithm: Time Complexity

$$
\begin{aligned}
T(n,k) &\leq 2T(n-1,k-1) + cm \\
&\leq 2T(n-1,k-1) + cnk \\
&\dots \quad (Iteration) \\
&= O(2^k nk)
\end{aligned}
$$

Parameterized Algorithm: Time Complexity

$$
\begin{aligned}
T(n,k) &\leq 2T(n-1,k-1) + cm \\
&\leq 2T(n-1,k-1) + cnk \\
\ldots & \quad (Iteration) \\
&= O(2^k nk)
\end{aligned}
$$

Brute-force algorithm's time complexity was $O(kn^{k+1})$.

Approximation algorithm

But we cannot use the exact algorithm in large graphs. What is the solution then?

Approximation algorithm

But we cannot use the exact algorithm in large graphs. What is the solution then?

Use approximations!
Using approximation algorithms, we can efficiently find a vertex cover that is near-optimal.

Approximation algorithm

Let's see a simple approximation algorithm.

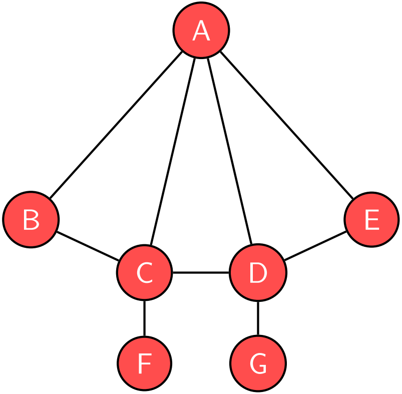
Algorithm

initialize vertexCover $= \emptyset$
while the graph has at-least one edge:
    pick a random edge (u, v)
    vertexCover $=$ vertexCover $\cup \{u, v\}$
    remove all incident edges of u
    remove all incident edges of v

Time complexity : $O(|V| + |E|)$

Approximation algorithm: Simulation

Let's consider a graph G, which has 7 vertices and 9 edges and simulate the algorithm mentioned.

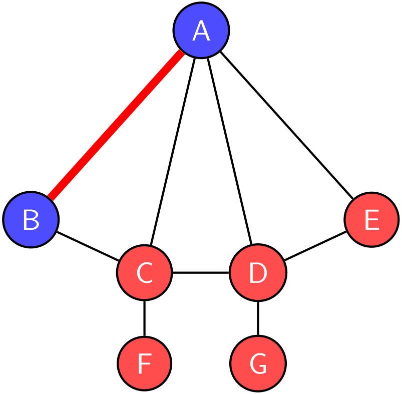


Vertex cover $= \emptyset$

The vertices are
$\{A, B, C, D, E, F, G\}$
The edges are

1. (A, B)
2. (A, C)
3. (A, D)
4. (A, E)
5. (B, C)
6. (C, D)
7. (D, E)
8. (C, F)
9. (D, G)

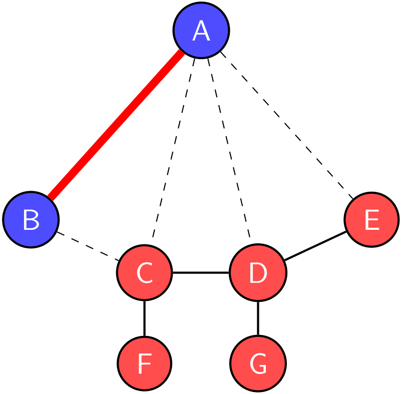
Approximation algorithm: Simulation



Vertex cover $= \{A, B\}$

- Pick a random edge **(A,B)** .
- Add **A** and **B** to the vertex cover set.

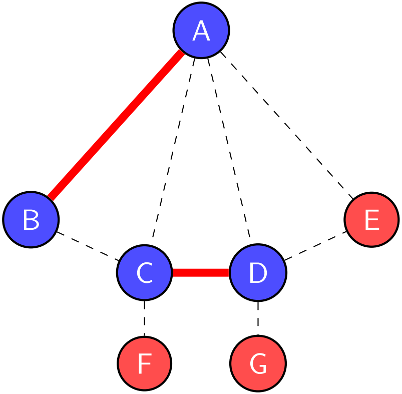
Approximation algorithm: Simulation



Vertex cover $= \{A, B\}$

- Remove every edge incident to either **A** or **B**.

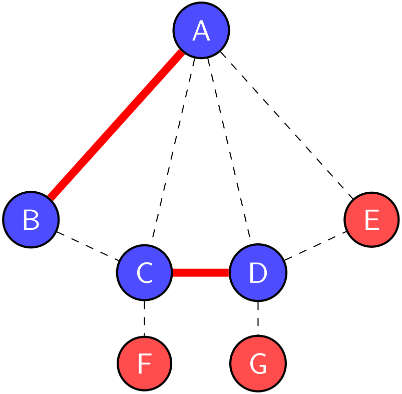
Approximation algorithm: Simulation



Vertex cover $= \{A, B, C, D\}$

- Again pick a random edge **(C,D)** .
- Add **C** and **D** to the vertex cover set.
- Remove every edge incident to either **C** or **D**.

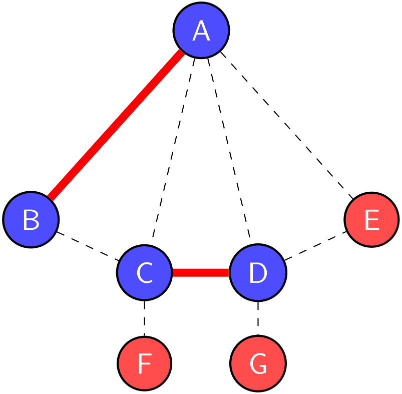
Approximation algorithm: Simulation



$$\textbf{Vertex cover} = \{A, B, C, D\}$$

- No more edges left to pick. We are done.
- We have found a vertex cover of size 4.

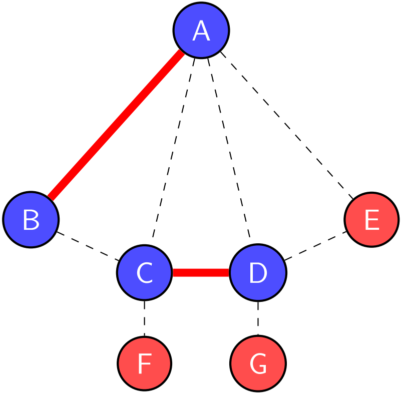
Approximation algorithm: Simulation



Vertex cover $= \{A, B, C, D\}$

- No more edges left to pick. We are done.
- We have found a vertex cover of size 4.

Is this the minimum cover?



Vertex cover $= \{A, B, C, D\}$

- No more edges left to pick. We are done.
- We have found a vertex cover of size 4.

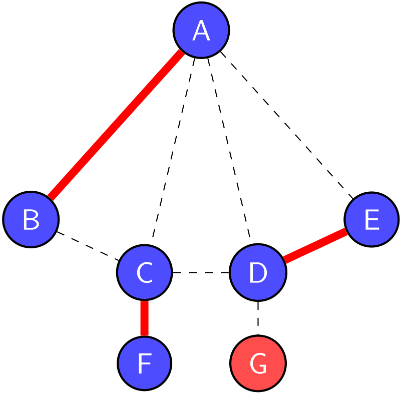
Is this the minimum cover?
No! Minimum Vertex Cover is of size 3.
$\{A, C, D\}$

Result analysis

It may seem that our result is quite good. But what would have happened if during the second step we had picked the edge **(D,E)**?

Result analysis

It may seem that our result is quite good. But what would have happened if during the second step we had picked the edge **(D,E)**?



This time we get a vertex cover of size 6 which is twice the size of an optimal vertex cover!!

Vertex cover $= \{A, B, C, D, E, F\}$

Result analysis

Can it get any worse?

Result analysis

Can it get any worse?

No! The vertex cover returned by APPROXIMATE VERTEX COVER is at most twice the size of an optimal vertex cover. We can also prove that.

Proof

Observations

- In the set of edges that we picked, no two share an endpoint.
- If the algorithm picked k edges, the vertex cover found has size $2k$.

From these observations, any vertex cover must have size at least k since it needs to have at least one endpoint of each of these edges, and since these edges don't touch, there are k different vertices. So the algorithm is a factor 2 approximation.

Factor 1.99

As we have achieved a factor of 2, is it possible to efficiently achieve a factor 1.99?

Factor 1.99

As we have achieved a factor of 2, is it possible to efficiently achieve a factor 1.99?

Unfortunately, nobody knows if it is possible to efficiently achieve a factor 1.99.

Thank You