Vertex Cover with Bounded Search Tree Algorithm

Task:

Let $\boldsymbol{k}$ be a parameter with $0 < \boldsymbol{k} \leq \boldsymbol{n}$. Assume that, for some reason, we are only interested in a vertex cover $\boldsymbol{S}$ that has at most $\boldsymbol{k}$ elements, i.e., $|S| \leq \boldsymbol{k}$ holds. Design a search tree algorithm with branch and bound that has a search tree with $\boldsymbol{O}(\mathbf{2}^{\boldsymbol{k}})$ nodes, instead of $\boldsymbol{O}(\mathbf{2}^{\boldsymbol{n}})$.

```
1: procedure VertexCoverBoundedSearch(G, k)
2:          Let e = {u, v} ∈ G //Consider any edge
            // (u ∈ S) or (v ∈ S) or both
3:          Try B1 = VertexCoverBoundedSearch(G − u, k − 1)  // Remove all edge connected with u
4:          Try B2 = VertexCoverBoundedSearch(G − v, k − 1)  // Remove all edge connected with v
5:          Return min(B1, B2) + 1
```

6: *Base case 1:* If k = 0, $\left\{ \begin{array}{ll} If\ G\ not\ empty\ , & return\ FAIL \\ If\ G\ empty\ , & return\ 0 \end{array} \right\}$

7: *Base case 2:* If G empty, return 0.

Explanation of Algorithm

Inputs:

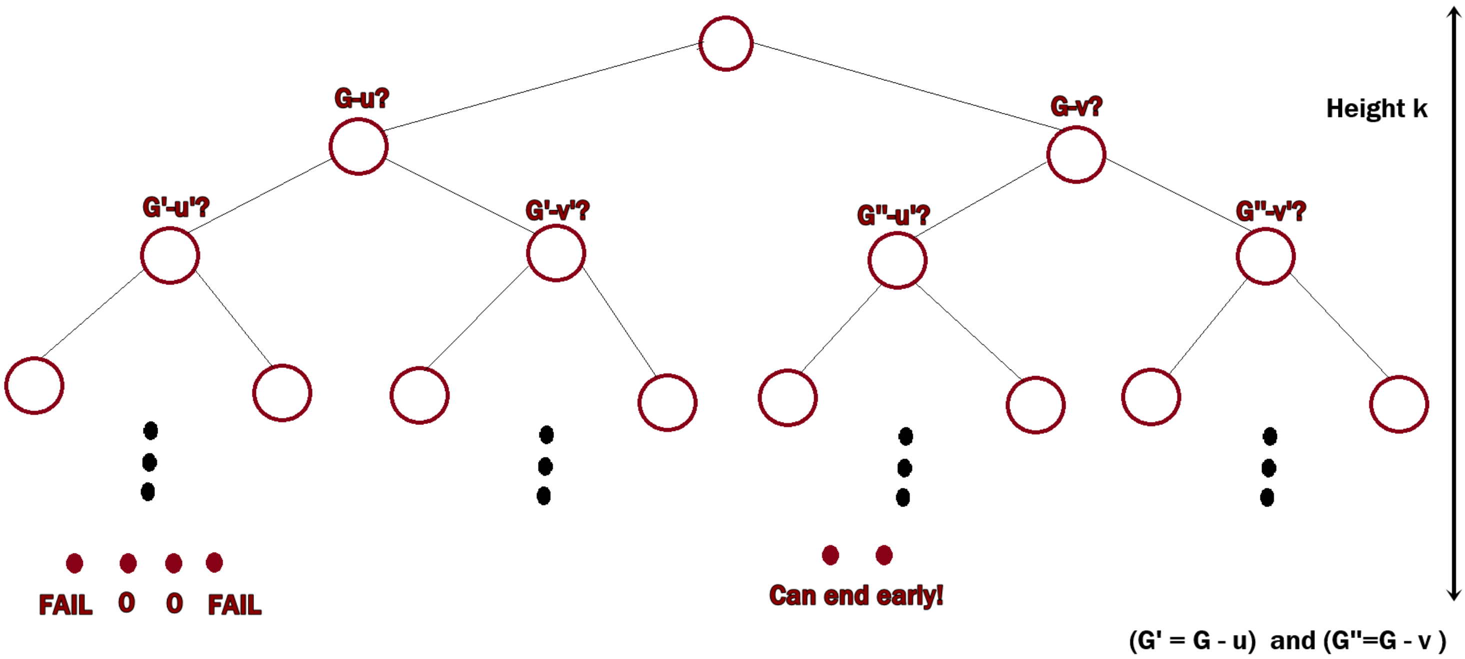
G ==> Graph

k ==> Bound parameter (k is the bound for vertex number.)

The e variable we create represents all the edges connecting the u and v nodes on the Graph. The algorithm is called recursively. In steps 3 and 4, we remove the u and v vertexes that make up the e edge and all the edges connected to them from the Graph. Then we assume that we found 1 of the desired number of vertexes. In this case, we decrease k by 1 to find other vertexes. And we call our function again with G-u and k-1 parameters.

If the k parameter is equal to 0, this means that we use the number of vertexes we want. k vertex (3 vertex,2vertex…) and the related items are deleted. In this case, if Graph is still not empty, the vertex cover algorithm cannot be implemented with k vertex. (FAIL) If Graph is empty, it returns 0.

TREE



Complexity Analysis

T(n, k) = running time on G of size n, parameter k.

$T(n, k) = 2 \cdot \underbrace{T(n, k-1)}_{(G - u,\ k - 1)} + O(n)$ (since removing edges take O(n) time)

= $O(2^k n)$