CSE515: Advanced Algorithms
Notes on Lecture 16: Algorithms for Vertex Cover

Antoine Vigneron

April 22, 2021

1 On ApproxVertexCover

Let C^* denote a minimum vertex cover.

Theorem. ApproxVertexCover *returns a vertex cover* C *of size at most* $2|C^*|$.

Proof. The algorithm picks a subset of edges one by one. (These are the red edges in the figure.) Let's call this set of edges F. These edges are disjoint, as we remove the vertices u and v after picking edge (u, v). As C^* is a vertex cover, there is at least one vertex of C^* in each edge of F, and thus $|F| \leqslant |C^*|$. As we return both endpoints of each edge in F, we have $|C| = 2|F|$, and thus $|C| \leqslant 2|C^*|$.

We also need to argue that C is a vertex cover. At each step, we delete the edges covered by the two vertices we inserted in C. At the end, no edge remains, which means that all the edges initially in E are covered by C. □

This analysis is tight, because ApproxVertexCover is not an α-approximation algorithm for any $\alpha < 2$. For instance, if the input graph G consists of only two vertices $V = \{u, v\}$ connected by an edge (u, v), then the algorithm returns $C = \{u, v\}$, while an optimal solution is $C^* = \{u\}$.

2 On FPVertexCover

Lemma. *If* FPVertexCover *returns* infeasible *at Line 13, then* G *has no vertex cover of size* k.

Proof. We make a proof by contradiction. Suppose that G has a vertex cover C' of size k. Then at least one of the two vertices of (u, v) is in C'. Without loss of generality, assume that $u \in C'$. Then $C' \setminus \{u\}$ is a vertex cover of $G \setminus \{u\}$ of size $k - 1$. Then the algorithm would have returned $C' = C \cup \{u\}$ at Line 13, and we would not have reached Line 13. □

Theorem. FPVertexCover *returns a vertex cover of size* k*, if there is one. It runs in time* $O(2^k kn)$.

Correctness follows from the Lemma above, and the observations we made on lines 5 and 9.

To analyze the algorithm, we can use two methods: The substitution method or the recursion tree method. Our analysis is uses the following simple observation: In each recursive call, we spend

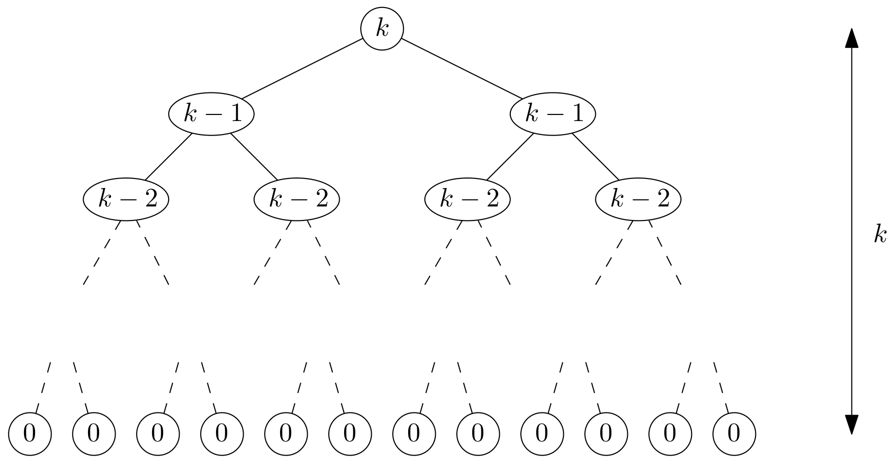

Figure 1: Recursion to of FPVertexCover

time $O(kn)$. We denote by $T(n,k)$ an upper bound on the running time on input of size n and with cover size k. So we have, for some constant c:

$$T(n,1) \leqslant cn \tag{1}$$

$$T(n,k) \leqslant 2T(n,k-1) + ckn \qquad \text{whenever} \geqslant 2 \tag{2}$$

Substitution method. The substitution method is a proof by induction. (You can find a more detailed explanation in CSE331, or in the CLRS textbook.) Remember that a proof by induction has to steps: the base step and the inductive step.

We prove by induction on k that $T(n,\ell) \leqslant c2^k \ell k$ for all k. The base step $k = 1$ follows from (1). We now prove the inductive step. Suppose that $k \geqslant 2$, and that for any $1 \leqslant \ell < k$, we have $T(n,\ell) \leqslant c2^\ell \ell n$. Then we have

$$\begin{aligned} T(n,k) &\leqslant 2T(n,k-1) + ckn && \text{by (2)} \\ &\leqslant 2c2^{k-1}(k-1)n + ckn && \text{by induction hypothesis (IH)} \\ &= c2^k kn - c2^k n + ckn && \\ &\leqslant c2^k n && \text{because } k \geqslant 1 \end{aligned}$$

This completes the proof.

Recursion tree. The recursion tree is drawn in Figure 1. There are k levels, and the number of nodes doubles at each level. So the total number of nodes is $1 + 2 + 4 + \cdots + 2^k = 2^{k+1} - 1$. At each node the algorithm spends time at most ckn, so the running time is upper bounded by $2^{k+1}ckn = O(2^k kn)$.

Brute force approach. A vertex cover of size k, if it exists, can also be computed by brute force. The brute-force algorithm simply tries all possible subsets C of size k, and checks if each set is a vertex cover. The number of subsets of size k is

$$\binom{n}{k} = \frac{n(n-1)\ldots(n-k+1)}{k!} = O(n^k).$$

For each subset we can check in time $O(kn)$ whether it is a vertex cover. Hence the running time is $O(kn^{k+1})$. This is polynomial, but it is much larger than the bound $O(2^k kn)$ on the running time of FPVertexCover: It is $(n/2)^k$ times larger.