

# CSCI3160: Regular Exercise Set 11

Prepared by Yufei Tao

**Problem 1.** Consider a complete bipartite graph  $G = (V, E)$ :

- $V$  has  $2n$  vertices, including  $n$  black vertices and  $n$  white vertices.
- $E$  has  $n^2$  edges, including an edge between every black vertex and every white vertex.

Use  $G$  to explain why 2 is the best the approximation ratio that we can prove for the vertex cover algorithm discussed in our lecture.

**Solution.** It is easy to verify that our vertex cover algorithm picks all the  $2n$  vertices. An optimal solution, however, should include only  $n$  vertices (e.g., all the black ones).

**Problem 2\*.** Let  $G = (V, E)$  be an input graph to the vertex cover problem. If  $G$  is a tree, describe an  $O(|V|)$ -time algorithm that finds an optimal vertex cover of  $G$ .

(Hint: Dynamic programming.)

**Solution.** Root the tree  $G$  at an arbitrary node. For each node  $u$  of the tree, define  $T(u)$  the subtree rooted at  $u$ . In addition, define

- $\text{OPT}(u, \text{yes})$  as the size of an optimal vertex cover of  $T(u)$ , provided that  $u$  belongs to the vertex cover.
- $\text{OPT}(u, \text{no})$  as the size of an optimal vertex cover of  $T(u)$ , provided that  $u$  does not belong to the vertex cover.

If  $u$  is a leaf, then

$$\begin{aligned}\text{OPT}(u, \text{yes}) &= 1 \\ \text{OPT}(u, \text{no}) &= 0.\end{aligned}$$

If  $u$  is an internal node, then

$$\begin{aligned}\text{OPT}(u, \text{yes}) &= \sum_{\text{child } v \text{ of } u} \min\{\text{OPT}(v, \text{yes}), \text{OPT}(v, \text{no})\} \\ \text{OPT}(u, \text{no}) &= \sum_{\text{child } v \text{ of } u} \text{OPT}(v, \text{yes})\end{aligned}$$

Let  $r$  be the root of  $G$ . It is now rudimentary to compute  $\text{OPT}(r, \text{yes})$  and  $\text{OPT}(r, \text{no})$  in  $O(|V|)$  time (go through the nodes in a bottom-up order). The optimal vertex cover size is  $\min\{\text{OPT}(r, \text{yes}), \text{OPT}(r, \text{no})\}$ . To obtain an optimal vertex, apply the piggyback technique.

**Problem 3\*\*.** Prof. Goofy proposes the following algorithm to find a vertex cover of  $G = (V, E)$ :

**algorithm** max-deg-VC

Input:  $G = (V, E)$

1.  $S = \emptyset$
2. **while**  $E$  not empty **do**
3.      $v \leftarrow$  a vertex with the maximum degree in the current  $G$
4.     add  $v$  to  $S$
5.     remove from  $E$  all the edges of  $v$

Show that the approximation ratio of this algorithm is greater than 2.

**Solution.** Let us construct a bipartite graph  $G$  as follows. The set  $L$  of left vertices is  $\{u_1, u_2, \dots, u_{16}\}$ . To generate the right vertices, for each  $i \in [2, 16]$ , we create a group  $R_i$  which contains  $s_i = \lfloor 16/i \rfloor$  vertices, denoted as  $R_i[1], R_i[2], \dots$ , and  $R_i[s_i]$ , respectively. The set  $R$  of right vertices is the union of  $R_2, R_3, \dots, R_{16}$ . The size of  $R$  is  $\sum_{i=2}^{16} s_i = 34$ .

Generate the edges of  $G$  as follows: for each group  $i \in [2, 16]$ , connect  $R_i[j]$  to the  $i$  vertices  $u_{i(j-1)+1}, u_{i(j-1)+2}, \dots, u_{ij}$ .

Running Prof. Goofy's algorithm, you will see that it picks all the 34 right vertices. As an optimal solution, we can pick the 16 left vertices.

**Problem\* 4 (Max-Cut).** Let  $G = (V, E)$  be a simple undirected graph. Given a subset  $S \subseteq V$ , a *cut* induced by  $S$  is the set of edges  $e \in E$  such that  $e$  has a vertex in  $S$  and another vertex in  $V \setminus S$ . Let  $\text{OPT}_G$  be the maximum size of a cut that can be induced by any  $S \subseteq V$ . Design a  $\text{poly}(|V|)$ -time (i.e., polynomial time in  $|V|$ ) algorithm that returns a cut of size at least  $2 \cdot \text{OPT}_G$  in expectation.

(Hint: Random assignment.)

**Solution.** Start with an empty  $S$ . For each vertex  $u \in V$ , toss a fair coin. If the coin comes up heads, add  $u$  to  $S$ ; otherwise, leave  $u$  out of  $S$ . It is easy to prove that each edge  $\{u, v\} \in E$  contributes to the cut induced by  $S$  with probability  $1/2$ . Hence, the cut has size  $|E|/2$  in expectation, which is at least  $2 \cdot \text{OPT}_G$  by the trivial fact  $|E| \geq \text{OPT}_G$ .

**Problem 5.** Let  $G = (V, E)$  be a simple undirected graph where each edge  $e \in E$  is assigned a non-negative weight  $w(e)$ .  $G$  is connected. A *spanning walk* of  $G$  is a path that visits every vertex at least once. Let  $\text{OPT}_G$  be the shortest length of all spanning walks. Design a  $\text{poly}(|V|)$ -time algorithm to find a spanning walk with length at most  $2 \cdot \text{OPT}_G$ .

**Solution.** Let  $\lambda$  be the weight sum of all the edges in a minimum spanning tree of  $G$ . Any spanning walk must contain the edges of some spanning tree. Therefore,  $\text{OPT}_G \geq \lambda$ . On the other hand, our approximation algorithm for the traveling salesman problem (discussed in the lecture) finds a spanning walk with length at most  $2 \cdot \lambda$ , which is thus at most  $2 \cdot \text{OPT}_G$ .