

*Note:* Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

## 1 Reducing Hamiltonian Cycle to Hamiltonian Path

(Also known as Rudrata Cycle and Rudrata Path in DPV.)

**Hamiltonian Cycle:** Given a directed graph  $G = (V, E)$ , is there a cycle that visits every vertex exactly once?

**Hamiltonian Path:** Given a directed graph  $G = (V, E)$ , is there a path that visits every vertex exactly once?

Given that Hamiltonian Cycle is **NP**-complete, prove that Hamiltonian Path is **NP**-complete.

- Show that Hamiltonian Path is in **NP**.
- Show that Hamiltonian Path is **NP**-hard by giving a polynomial-time reduction from Hamiltonian Cycle to Hamiltonian Path. Prove correctness in both directions.

### Solution:

- Given a purported solution  $P$  (a list of vertices), we verify it by checking that (i)  $P$  contains each vertex of  $V$  exactly once, and (ii) every consecutive pair  $(u, w)$  in  $P$  is an edge of  $G$ . Checking (i) takes  $O(|V|^2)$  time (or  $O(|V|)$  with a hash set) and checking (ii) takes  $O(|V| \cdot |E|)$  time naively, both polynomial in the input size.
- We give a reduction from Hamiltonian Cycle to Hamiltonian Path.

**Reduction.** Given an instance  $G = (V, E)$  of Hamiltonian Cycle, pick an arbitrary vertex  $v \in V$  and construct  $G' = (V', E')$  as follows:

- Replace  $v$  with two new vertices  $v'$  and  $v''$ .
- For every directed edge  $(u, v) \in E$  into  $v$ , add the edge  $(u, v')$  to  $E'$ .
- For every directed edge  $(v, w) \in E$  out of  $v$ , add the edge  $(v'', w)$  to  $E'$ .
- Keep all other vertices and edges of  $G$  unchanged.

In particular, in  $G'$  the vertex  $v'$  has no outgoing edges and  $v''$  has no incoming edges. The transformation takes  $O(|V| + |E|)$  time.

**Correctness ( $\Rightarrow$ ).** Suppose  $G$  has a Hamiltonian cycle. Without loss of generality we can write the cycle as

$$v \rightarrow w_1 \rightarrow w_2 \rightarrow \cdots \rightarrow w_{n-1} \rightarrow v.$$

Then  $v'' \rightarrow w_1 \rightarrow w_2 \rightarrow \cdots \rightarrow w_{n-1} \rightarrow v'$  is a Hamiltonian path in  $G'$ : the edge  $(v, w_1)$  becomes  $(v'', w_1)$ , the edge  $(w_{n-1}, v)$  becomes  $(w_{n-1}, v')$ , and all internal edges are unchanged. The path visits every vertex of  $V' = (V \setminus \{v\}) \cup \{v', v''\}$  exactly once.

**Correctness ( $\Leftarrow$ ).** Suppose  $G'$  has a Hamiltonian path  $P'$ . Since  $v''$  has no incoming edges in  $G'$ ,  $P'$  must start at  $v''$ ; since  $v'$  has no outgoing edges,  $P'$  must end at  $v'$ . So  $P'$  has the form

$$v'' \rightarrow w_1 \rightarrow w_2 \rightarrow \cdots \rightarrow w_{n-1} \rightarrow v',$$

where  $\{w_1, \dots, w_{n-1}\} = V \setminus \{v\}$ . Merging  $v'$  and  $v''$  back into  $v$  gives the cycle  $v \rightarrow w_1 \rightarrow \cdots \rightarrow w_{n-1} \rightarrow v$  in  $G$ , which is a Hamiltonian cycle.

Combining both directions,  $G$  has a Hamiltonian cycle if and only if  $G'$  has a Hamiltonian path, so Hamiltonian Path is **NP**-hard. Together with part (a), it is **NP**-complete.

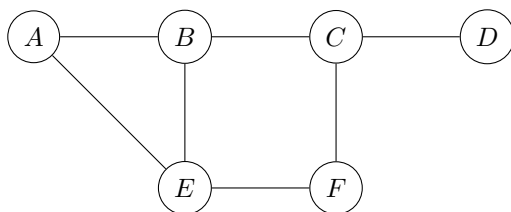
## 2 Vertex Cover to Set Cover

To help jog your memory, here are some definitions:

**Vertex Cover:** given an undirected unweighted graph  $G = (V, E)$ , a vertex cover  $C_V$  of  $G$  is a subset of vertices such that for every edge  $e = (u, v) \in E$ , at least one of  $u$  or  $v$  must be in the vertex cover  $C_V$ .

**Set Cover:** given a universe of elements  $U$  and a collection of sets  $\mathcal{S} = \{S_1, \dots, S_m\}$ , a set cover is any (sub)collection  $C_S$  whose union equals  $U$ .

In the *minimum vertex cover problem*, we are given an undirected unweighted graph  $G = (V, E)$ , and are asked to find the smallest vertex cover. For example, in the following graph,  $\{A, E, C, D\}$  is a vertex cover, but not a minimum vertex cover. The minimum vertex covers are  $\{B, E, C\}$  and  $\{A, E, C\}$ .



Then, recall in the *minimum set cover problem*, we are given a set  $U$  and a collection  $\mathcal{S} = \{S_1, \dots, S_m\}$  of subsets of  $U$ , and are asked to find the smallest set cover. For example, given  $U := \{a, b, c, d\}$ ,  $S_1 := \{a, b, c\}$ ,  $S_2 := \{b, c\}$ , and  $S_3 := \{c, d\}$ , a solution is  $C_S = \{S_1, S_3\}$ .

- Give an efficient reduction from the minimum vertex cover problem to the minimum set cover problem, and prove its correctness.
- What does your reduction imply about the complexity of the minimum set cover problem? (You may assume minimum vertex cover is **NP**-hard.)

### Solution:

- Intuition.** A vertex cover must “hit” every edge. A set cover must “cover” every element of the universe. So the natural dictionary is: *edges become the things we need to cover, and vertices become the sets that cover them*. Each vertex  $v$  is responsible for the edges it touches, so the set  $S_v$  should be the collection of edges incident to  $v$ .

**Reduction.** Given an instance  $G = (V, E)$  of minimum vertex cover, construct a set cover instance  $(U, \mathcal{S})$  by

$$U = E, \quad \mathcal{S} = \{S_v : v \in V\}, \quad S_v = \{e \in E : v \text{ is an endpoint of } e\}.$$

This transformation takes  $O(|V| + |E|)$  time. Given a set cover  $C_S = \{S_{u_1}, \dots, S_{u_k}\}$  on the constructed instance, return the vertex cover  $C_V = \{u_1, \dots, u_k\}$ .

**Example.** For the graph above, the universe is  $U = \{AB, AE, BC, BE, CD, CF, EF\}$  and the sets are

$$\begin{array}{lll} S_A = \{AB, AE\}, & S_B = \{AB, BC, BE\}, & S_C = \{BC, CD, CF\}, \\ S_D = \{CD\}, & S_E = \{AE, BE, EF\}, & S_F = \{CF, EF\}. \end{array}$$

The set cover  $\{S_A, S_C, S_E\}$  covers every element of  $U$  and corresponds exactly to the vertex cover  $\{A, C, E\}$ . Conversely,  $\{S_B, S_D\}$  leaves the element  $AE$  uncovered—matching the fact that  $\{B, D\}$  is not a vertex cover because the edge  $AE$  has neither endpoint in it.

**Correctness.** We show a size-preserving bijection between vertex covers of  $G$  and set covers of  $(U, \mathcal{S})$ :

- $(C_V \Rightarrow C_S)$ : Let  $C_V \subseteq V$  be a vertex cover. Take  $C_S = \{S_v : v \in C_V\}$ , which has the same size. For any edge  $e = (u, w) \in E = U$ , the vertex cover property gives  $u \in C_V$  or  $w \in C_V$ , so  $e \in S_u$  or  $e \in S_w$ , i.e.  $e$  is covered.
- $(C_S \Rightarrow C_V)$ : Let  $C_S = \{S_{u_1}, \dots, S_{u_k}\}$  be a set cover. Take  $C_V = \{u_1, \dots, u_k\}$ , same size. For any edge  $(u, w) \in E$ , the element  $(u, w) \in U$  must be covered by some  $S_{u_i}$ ; by construction  $u_i \in \{u, w\}$ , so the edge has an endpoint in  $C_V$ .

Since both directions preserve size, a minimum set cover on  $(U, \mathcal{S})$  yields a minimum vertex cover of  $G$ .

- (b) The reduction runs in polynomial time and transforms any instance of minimum vertex cover into an instance of minimum set cover with the same optimum value. So if we had a polynomial-time algorithm for minimum set cover, we could use it (via this reduction) to solve minimum vertex cover in polynomial time. Since minimum vertex cover is **NP**-hard, minimum set cover is **NP**-hard as well.

**Takeaway on approximation.** The reduction also tells us something about *approximability*: any  $\alpha$ -approximation for minimum set cover immediately gives an  $\alpha$ -approximation for minimum vertex cover on the constructed instance, because the bijection above preserves cover sizes exactly. For instance, the classical greedy algorithm for set cover achieves an  $O(\log |U|) = O(\log |E|)$  approximation, so by pushing a vertex-cover instance through this reduction we get an  $O(\log |E|)$ -approximation for minimum vertex cover “for free.” (In fact vertex cover has a better, constant-factor 2-approximation via a different technique—but the general message is that reductions can transfer algorithmic ideas as well as hardness.)

**Does the reduction go the other way?** No—not in a way that would let us conclude vertex cover is at least as hard as set cover. Set cover is a strict generalization: vertex cover is the special case where every set in  $\mathcal{S}$  has size exactly 2 (each edge has two endpoints). A reduction in the opposite direction would have to encode arbitrary sets using only size-2 “edges,” which is not in general possible in polynomial time without changing the optimum. So this reduction establishes “set cover is *at least as hard as* vertex cover” but not the reverse.

### 3 Maximum Coverage

In the maximum coverage problem, we have  $m$  subsets of the set  $\{1, 2, \dots, n\}$ , denoted  $S_1, S_2, \dots, S_m$ . We are given an integer  $k$ , and we want to choose  $k$  sets whose union is as large as possible.

Give an efficient algorithm that finds  $k$  sets whose union has size at least  $(1 - 1/e) \cdot OPT$ , where  $OPT$  is the maximum number of elements in the union of any  $k$  sets. In other words,  $OPT = \max_{i_1, i_2, \dots, i_k} |\cup_{j=1}^k S_{i_j}|$ . Provide an algorithm description and justify the lower bound on the number of elements covered by your solution.

*Hint: be greedy! For the proof, you may use the following property without proof:  $(1 - 1/n)^n \leq 1/e$  for all  $n \in \mathbb{Z}$ .*

**Solution:** Similar to set cover, we choose sets greedily, each time adding the set that includes the most elements that are not covered by any sets included in our solution so far.

Let  $n_i$  be  $OPT$  minus the number of elements covered by the first  $i$  sets we choose. Initially, we have  $n_0 = OPT - 0 = OPT$ . Note that for any  $i$ , the  $k$  sets forming  $OPT$  cover at least  $n_i$  elements that our solution does not cover, which means one of these sets covers at least  $n_i/k$  elements our solution doesn't cover. The set we add in iteration  $i + 1$  is guaranteed to cover more new elements than this set, i.e., the  $(i + 1)$ -th set we add covers at least  $n_i/k$  new elements. Thus, we have

$$\begin{aligned} n_{i+1} &= OPT - (\text{number of elements covered by the first } i + 1 \text{ sets}) \\ &= OPT - (OPT - n_i + (\text{number of new elements covered by the } (i + 1)\text{-th set})) \\ &\leq OPT - (OPT - n_i + n_i/k) \\ &= n_i - n_i/k \\ &= n_i(1 - 1/k) \end{aligned}$$

In turn, we have  $n_k \leq (1 - 1/k)^k n_0 = (1 - 1/k)^k OPT \leq \frac{1}{e} \cdot OPT$ , or equivalently our solution covers at least  $(1 - 1/e) \cdot OPT$  elements.