University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger, P. Schneider

Algorithm Theory
Sample Solution Exercise Sheet 12

Due: Tuesday, 25th of January, 2022, 4 pm

Exercise 1: Minimum Vertex Cover Approximation *(20 Points)*

Let $G = (V, E)$ be an *undirected, unweighted* graph. Consider the following algorithms that give approximate solutions to the minimum vertex cover problem.

Algorithm 1 `alg1`

```
1: S ← ∅                                   ▷ create an empty set
2: while E ≠ ∅ do
3:     pick some edge {u, v} ∈ E
4:     S ← S ∪ {u, v}
5:     remove edges incident to u or v from E
6: return S
```

Algorithm 2 `alg2`

```
1: S ← ∅                                   ▷ create an empty set
2: while E ≠ ∅ do
3:     pick vertex v ∈ V of maximal degree
4:     S ← S ∪ {v}
5:     remove edges incident to v from E
6: return S
```

(a) Show that `alg1` and `alg2` output valid vertex covers. *(3 Points)*

(b) Argue why `alg1` provides a 2-approximation of the minimum vertex cover problem. *(3 Points)*

Hint: You can use results that we proved in the lecture.

(c) Argue why `alg2` provides a $O(\log n)$-approximation of the minimum vertex cover problem. *(5 Points)*

Hint: You can use results that we proved in the lecture.

(d) Show that the solution provided by `alg2` is only a $\Theta(\log n)$ approximation for some graphs. *(9 Points)*

Hint: Give a bipartite graph with node set $V = L \cup R$ and $|L| = k$ and $|R| = \Theta(k \log k)$, where `alg2` outputs R but the best solution would be L.

Sample Solution

(a) **Termination:** Both algorithms eventually terminate with $E = \emptyset$, since in each iteration of the loop we remove at least one edge from E. This is obvious for `alg1`. In an iteration of `alg2`, we must pick a node v with $deg(v) > 0$ in line 4, as otherwise $E = \emptyset$ already. Therefore, v has at least one incident edge that is removed in line 5.

Correctness: In both algorithms, every edge that is removed from E (in either algorithm) is covered by some endpoint that is added to the vertex cover S as can be seen in lines 4 and 5. And since every edge gets removed eventually, the algorithms output vertex covers.

(b) The edges we pick in line 3 clearly form a maximal matching, call it M. We have seen in the lecture that the minimum vertex cover (call it S^*) must be at least as large as any matching, in particular $|S^*| \geq |M|$ (remark: the idea was that one endpoint of each matching edge must necessarily be in *any* vertex cover). Since we add *both* endpoints of each edge in M, the vertex cover S we obtain has size at most $|S| = 2|M| \leq 2|S^*|$.

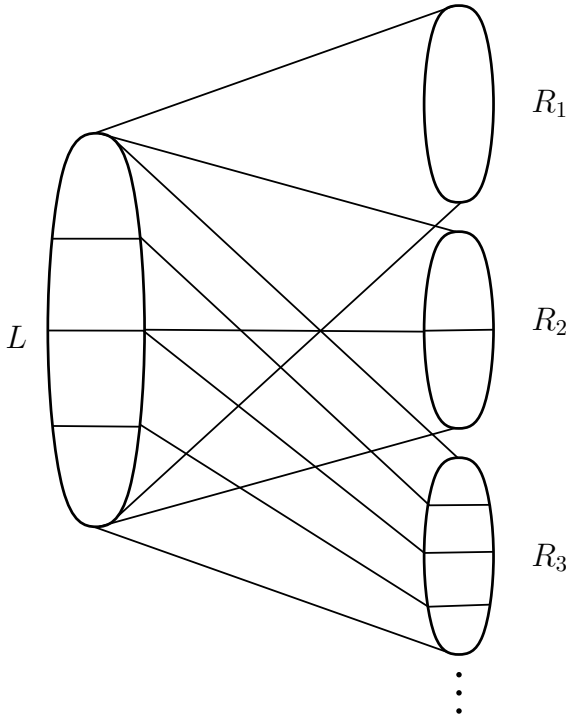
(c) The vertex cover problem can be seen as a set cover problem, where the groundset is the set of edges E, and the family of sets $\mathcal{S} \subseteq 2^E$ are sets of incident edges of any node (call those $I(v) \subseteq E$, for $v \in V$). So more formally we define $\mathcal{S} = \{I(v) \mid v \in V\}$.

Our greedy approach to solve set cover was to pick in each step the set that covers the most, yet uncovered, elements from the groundset. That is exactly what happens in `alg2`, where we always choose the node with highest degree in the remaining set of edges.

In the lecture we showed that this greedy approach gives a set cover that is at most $O(\log n)$ times worse than the best set cover.

(d) For simplicity, let k be a power of two. Let L consist of k nodes and let $R = R_1 \cup \cdots \cup R_\ell$ with disjoint sets of nodes R_i, $|R_i| = k/2$ and $\ell = \log_2(k) - 1$. It is clear that $|R| = \Theta(k \log k)$. We arrange the nodes in L and the R_i each in a column from top to bottom (c.f., image below).

We connect L and the R_i with edges as follows. We form a complete bipartite graph between L and R_1. The we form complete bipartite graphs with the top half of nodes in L and the top half of nodes of R_2 and do the same with the respective bottom halves of L and R_2. In general we divide L and R_i in 2^i-th quantiles, whereas the respective quantiles form pairwise complete bipartite graphs. This is roughly outlined by the following picture.



Now the degree of every node in R_1 is k whereas each node in L has degree $k/2+k/4+k/8+\cdots+1 < k$. That means the greedy algorithm first chooses all nodes in R_1. After removing those and the incident edges all nodes in L have degree $< k/2$ wheres those in R_2 have degree $k/2$. So next we remove all nodes of R_2. This continues until all nodes in R are removed and there are no more edges left. Obviously, the best solution would be L, which is by a factor $\Theta(\log k)$ smaller. Since $n \in \Theta(k \log k)$ we have that $\Theta(\log k) = \Theta(\log(n/\log k)) = \Theta(\log n - \log\log k) = \Theta(\log n)$.