

CS2510 Homework

Feng-Hao Liu

Department of Computer Science, Brown University

Problem: Exercise 1.1 (Acyclic Subgraph)

- Input: A directed graph $G = (V, E)$
- Output: A set $E' \subseteq E$ such that $G' = (V, E')$ forms a acyclic graph, i.e. no cycle in G' .
- Objective: Maximize the cardinality of E' .
- Approximation Algorithm Goal: Give a 2-approx algorithm, i.e. the output of the algorithm $\geq \frac{1}{2}OPT$

Algorithm A: Let $|V| = n$, $|E| = m$,

1. Take an arbitrary labeling on each vertex with $\{1, 2, \dots, n\}$, so $V = \{v_1, v_2, \dots, v_n\}$ for the indices correspond to the labeling.
2. Let $E_1 = \{\overrightarrow{v_i v_j} \mid i < j \text{ and } \overrightarrow{v_i v_j} \in E\}$, $E_2 = \{\overrightarrow{v_i v_j} \mid i > j \text{ and } \overrightarrow{v_i v_j} \in E\}$. This step could be done by scanning through all edges in E and for each edge we decide which set it will be thrown into. This takes $O(|E|)$.
3. Output E_1 if $|E_1| \geq |E_2|$; else E_2 .

Remark 1. For simplicity we assume that G is a simple directed graph with NO self edges, NO multiple edges.

Note: the efficiency of the approximation is quite obvious.

Proof of feasibility:

Lemma 1. $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ are both acyclic graphs.

Proof. Let's use proof by contradiction. For symmetry, we prove G_1 is an acyclic graph, and the same argument goes with G_2 . Suppose there is a cycle with length k in G_1 , let's call it $C = v_{i_1} \rightarrow v_{i_2} \dots v_{i_k} \rightarrow v_{i_1}$ and let i_1 is the least index. By the rule we choose C_1 , we must have $i_1 < i_2 < \dots < i_k < i_1$, thus a contradiction. Therefore, G_1 is acyclic. \square

Proof of 2-approx:

Lemma 2. Algorithm A is really a 2-approx algorithm. That is $|E' \leftarrow A(G)| \geq \frac{1}{2}OPT$.

Proof. Here we observe some facts that help our proof:

1. $OPT \leq |E|$: The maximum subset of a set is the set itself. So this bound is trivial (and in fact we are using this bound.)
2. $E_1 \cup E_2 = E$, $E_1 \cap E_2 = \emptyset$. Since in the algorithm, for each edge, we either put it into either E_1 or E_2 . We never put an edge into both sets or no set; instead into exactly one of the two sets. Thus this fact is also obvious.

From the second fact, since we know E_1, E_2 can be viewed as a partition of E ; thus $|E_1| + |E_2| = |E|$. WLOG, suppose $|E_1| \geq |E_2|$, then we have $|E_1| + |E_1| \geq |E_1| + |E_2| = |E| \geq OPT$. Thus $|E_1| \geq \frac{1}{2}OPT$. *Remark: it is the same argument for the case $|E_1| < |E_2|$.* \square

Tightness of the algorithm A:

After knowing a 2-approx algorithm, we are going to ask if 2 is tight. Consider the following graph: $v_1 \rightarrow v_4 \rightarrow v_2 \rightarrow v_5 \rightarrow v_3$, (a line.) The algorithm on this input will output $E_1 = \{\overrightarrow{v_1v_4}, \overrightarrow{v_2v_5}\}$, and $|E_1| = 2$; however $OPT = 4$. In general, we can consider a graph family with $2k + 1$ vertices: $G_{2k-1} = v_1 \rightarrow v_{k+1} \rightarrow v_2 \rightarrow v_{k+2} \rightarrow \dots \rightarrow v_{2k-1} \rightarrow v_k$. The same argument works with such family of graph.

Remark 2. The output depends on how we label the graph. If we are lucky, perhaps we can get a better outcome. However, there exists some "unlucky" labelings, which lead to the tight $\frac{1}{2}OPT$. Also, since it is a 2-approx, even in the worst case we are still guaranteed to get $\frac{1}{2}OPT$.