Approximation Algorithms — Solution 3

2.6.2021

1) a) Obviously, we can check for a vertex that improves the number of crossing edges in time $O(n+m)$. Performing the swap can be done in $O(1)$, updating the number of crossing can be done in $O(1)$ as well. $\rightsquigarrow$ every iteration has polynomial running time.
In every step, we improve by at least one edge, so we can have at most $|E|+1$ iterations.
$\rightsquigarrow$ $O(m \cdot (n+m))$ total running time.

b) $S, T \subseteq V$ (Cut)
Let $v \in V$; then $d_S(v)$ is the number of neighbors of v in S. Analogously, let $d_T(v)$ be the number of neighbors of v in T, and $d_V(v)$ be the degree of v. For every vertex v: $d_V(v) = d_S(v) + d_T(v)$.

$$OPT \leq |E| = \frac{1}{2} \sum_{v \in V} d_V(v) = \frac{1}{2} \sum_{v \in V} d_S(v) + d_T(v)$$

$$= \frac{1}{2} \sum_{v \in S} d_S(v) + d_T(v) + \frac{1}{2} \sum_{v \in T} d_S(v) + d_T(v) = (*)$$

For $v \in S$, we have $d_S(v) \leq d_T(v)$, and for $v \in T$, we have $d_T(v) \leq d_S(v)$.

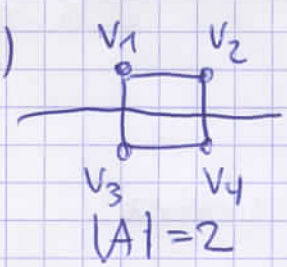
$\Leftrightarrow$ $d_S(v) + d_T(v) \leq 2 d_T(v)$ $(v \in S)$
$d_S(v) + d_T(v) \leq 2 d_S(v)$ $(v \in T)$

$$\Rightarrow (*) \leq \frac{1}{2} \sum_{v \in S} 2 d_T(v) + \frac{1}{2} \sum_{v \in T} 2 d_S(v) = \sum_{v \in S} d_T(v) + \sum_{v \in T} d_S(v)$$

$$= 2 \cdot |A|.$$

$\rightsquigarrow$ $OPT \leq |E| \leq 2 \cdot |A|$

(c)



$|A| = 2$

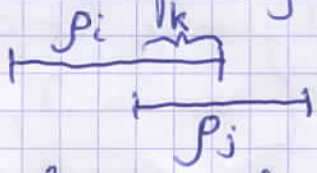
$OPT = 4$

In this example, A yields a cut with value 2, whereas OPT has 4 edges crossing its cut.

2) W.l.o.g., no string is contained in another string or its reverse. Otherwise, we eliminate the substring. We build a set cover instance from S.

The universe U is S. The set family is:

$F = \{s, s^R \mid s \in S\} \cup K$, where K consists of the following strings. For each ρ_i, ρ_j in $R = \{s, s^R \mid s \in S\}$ and each valid k, K contains a set corresponding to the string



. The weights of each set is the length of the corresponding string.

Each set (with corresponding string w) contains $\{s \mid s \text{ or } s^R \text{ is contained in } w\}$.

Our algorithm uses Greedy Set Cover to compute a set cover on this instance and concatenates the strings (in any order). Total length of our string:

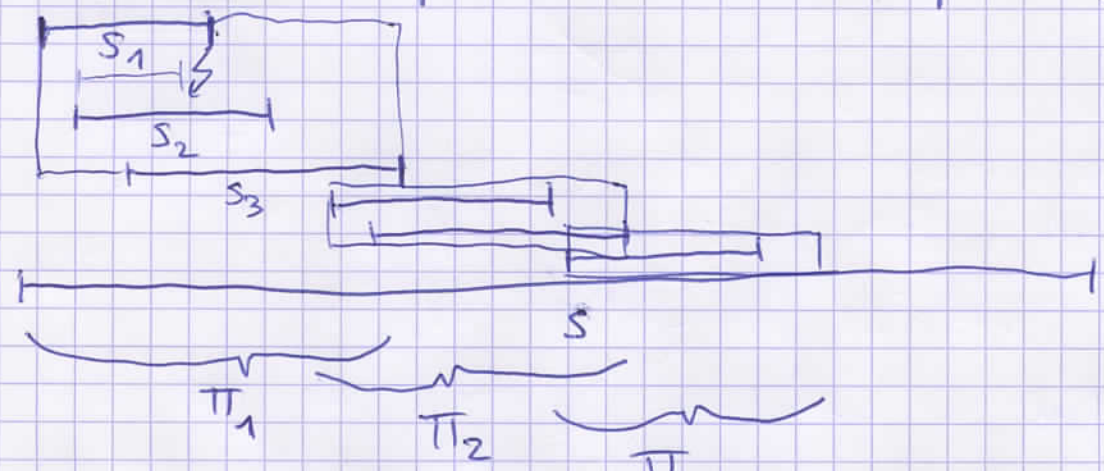
$G(\log n) \cdot OPT_{SC}$.

↑ optimal set cover solution

For $G(\log n)$ overall, we need to show that

$OPT_{SC} = G(1) \cdot OPT_S$.

Let S be the optimal shortest superstring.



Analogous to the lecture, Π_i and Π_{i+2} do not overlap.

$\rightsquigarrow$ Any letter in the optimal solution S is covered by at most 2 Π_i's.

$\rightsquigarrow$ $\sum_i w(\Pi_i) \leq 2 \cdot OPT_S$.

OPT_{SC}

$\rightsquigarrow$ Our algorithm is a $2 \cdot O(\log n) = O(\log n)$-approximation algorithm.