Algorithmic Lower Bounds - Assignment 3

Professor: Mohammad T. Hajiaghayi

Problem 1. Feige (via PCP) proved unless $NP \subseteq \mathsf{DTIME}(n^{\log\log n})$, there is no approximation algorithm for set cover with approximation factor within $(1-\epsilon)\ln n$ for any $\epsilon > 0$. Now consider the problem of maximum coverage, in which the input is the same as set cover (with all sets having cost 1) and an additional integer k and the goal is to find k sets with maximum size union. Via a gap-preserving reduction from set cover prove that unless $NP \subseteq \mathsf{DTIME}(n^{\log\log n})$, there is no approximation algorithm for maximum coverage with an approximation factor within $(1-1/e-\epsilon)$ for any $\epsilon > 0$.

Solution. We will show that maximum coverage problem even with unit cost cannot be approximated within $\left(1-\frac{1}{e}-\varepsilon\right)$ unless, $NP \subseteq DTIME(n^{loglogn})$. To prove we use Feige theorem mentioned in the statement of the problem. Consider a unit cost set-cover. Weight of each element is one.
Say there is an algorithm A with an approximation factor $\alpha > 1-1/e$.
We guess the number of the sets in the optimal solution of set-cover. Let it be k. Since the number of sets in the optimal solution is k, it is possible to cover all the elements using at most k sets.

New algorithm for the set-cover:

- Run A with limit $L = k$. It will cover at least $\alpha\ n$ elements.
- Choose sets for this cover C of αn elements.
- Remove C and elements which are covered.
- Iterate by running A in the reduced set.

Analysis: Let n_i denote the number of the uncovered elements at the start of the i^{th} iteration. Since, A covers at least αn_i at iteration i^{th} we have $n_{i+1} \leq n_i(1-\alpha)$.

Suppose we iterate $L+1$ times for $n_L \geq 1$
$1 \leq n_L \leq n_{L-1}(1-\alpha) \leq n_{L-2}(1-\alpha)^2 \leq \cdots \leq n(1-\alpha)^L$

Therefore,

$$\begin{aligned} L &\leq \frac{\ln\ n}{\ln\left(\frac{1}{1-\alpha}\right)} \\ kL &\leq k\frac{\ln\ n}{\ln\left(\frac{1}{1-\alpha}\right)} \\ &= opt\frac{\ln\ n}{\ln\left(\frac{1}{1-\alpha}\right)} \end{aligned}$$

If $\alpha > 1 - \frac{1}{e}$ then, $\frac{1}{1-\alpha} > e$ therefore, $\ln \frac{1}{1-\alpha} > 1$. This is contradiction to the Feige theorem.

Problem 2. Prove there is a parameterized reduction from dominating set to set cover.

Solution.

Let (G, k) be an instance of Dominating Set. We create an instance $(\mathcal{F}, U, k)$ of Set Cover as follows. We let $U := V(G)$ and for every $v \in V(G)$, we introduce the set $N_G[v]$ (the closed neighborhood of v) into $\mathcal{F}$. Suppose that D is a dominating of size k in G. Then the union of the corresponding k sets of $\mathcal{F}$ covers U: an uncovered element would correspond to a vertex of G not dominated by D. Conversely, if the union of k sets in $\mathcal{F}$ is U, then the corresponding k vertices of G dominate every vertex: a vertex not dominated in G would correspond to an element of U not covered by the k sets. □

Problem 3. Connected dominating set is a dominating set which induces a connected graph on vertices in the dominating set.

(a) Prove there is a parameterized reduction from dominating set to connected dominating set.

(b) Prove connected dominating set is in $W[2]$ by creating an instance of Weighted Circuit Satisfiability with weft two for it.

(c) Prove that connected dominating set is $W[2]$-complete.

Solution. (a)

Let (G, k) be an instance of Dominating Set. We construct a graph G' the following way.

(i) For every vertex $v \in V(G)$, we introduce two adjacent vertices v_1, v_2.
(ii) We make the set $\{v_1 \ : \ v \in V(G)\}$ a clique K of size $|V(G)|$.
(iii) We make v_1 and u_2 adjacent if v and u are adjacent in G.

We claim that (G, k) is a yes-instance of Dominating Set if and only if (G', k) is a yes-instance of Connected Dominating Set. Suppose first that $S = \{v^1, \ldots, v^k\}$ is a dominating set of size k in G. Then we claim that $S' = \{v_1^1, \ldots, v_1^k\}$ is a connected dominating set of size k in G'. Clearly, $G'[S']$ is a clique and hence it is connected. To see that S' is a dominating set in G', observe that v_1^1 dominates K, and if u is dominated by v^i in G, then u_2 is dominated by v_1^i in G'.

For the proof of the reverse direction of the equivalence, let S' be a connected dominating set of size k in G'. Let v be in S if at least one of v_1 and v_2 is in S'; clearly, $|S| \leq |S'| = k$. We claim that S is a dominating set of G. Consider any vertex $u \in V(G)$. Vertex u_2 of G' is dominated by some vertex v_1 or v_2 that belongs to S'. Then v is in S and, by the construction of G', it dominates u in G, as required. □

(b) Given an instance (G, k) of Connected Dominating Set, we construct an equivalent instance (C, k) of Weighted Circuit Satisfiability. Let $v_1, \ldots, v_n$ be the vertices of G. The input nodes of the constructed circuit are $x_{i,j}$ for $1 \leq i \leq k$, $1 \leq j \leq n$. The interpretation of $x_{i,j} = 1$ is that the i-th vertex of the solution is v_j.

(i) For $1 \leq i \leq k$, node z_i is the disjunction of $\{x_{i,j} : 1 \leq j \leq n\}$ and O_1 is the conjunction of all the z_i-s. Then $O_1 = 1$ expresses the requirement that there are integers $s_1, \ldots, s_k$ such that $x_{1,s_1}, \ldots, x_{k,s_k}$ are exactly the input nodes with value 1, describing a tuple $(v_{s_1}, \ldots, v_{s_k})$ of k vertices.

(ii) For $1 \leq j \leq i$, node w_j expresses that vertex v_j is dominated in the solution: it is the disjunction of the input nodes $\{x_{i,j'} : 1 \leq i \leq k, v_{j'} \in N[v_j]\}$.

(iii) Node O_2 expresses that the inputs represents a dominating set: it is the conjunction of w_j for every $1 \leq j \leq n$.

(iv) For every P such that $\emptyset \subsetneq P \subsetneq [k]$, node c_P expresses that there is an edge between $\{v_{s_i} : i \in P\}$ and $\{v_{s_i} : i \notin P\}$. For every $i_1 \in P$, $i_2 \in [k] \setminus P$, and pair (v_{j_1}, v_{j_2}) of adjacent or equal vertices in G, we introduce a node e_{P,i_1,i_2,j_1,j_2} that is the conjunction of x_{i_1,j_1} and x_{i_2,j_2}; the node c_P is the disjunction of all these nodes.

(v) Node O_3 expresses that $(v_{s_1}, \ldots, v_{s_k})$ induces a connected graph: it is the conjunction of all the nodes c_P.

(vi) The output node is the conjunction of O_1, O_2, and O_3 (more precisely, there is a small node O' that the conjunction of O_1 and O_2, and the output node is a small node that is the conjunction of O' and O_3).

Observe that the constructed circuit C has constant depth (independent of k) and weft 2: a path from an input to the output contains either

- a large node z_i and the large node O_1,
- or a large node w_j and the large node O_2,
- or a large node c_P and the large node O_3.

It is easy to see that if $(v_{s_1}, \ldots, v_{s_k})$ is a connected dominating set, then setting $x_{1,s_1}, \ldots, x_{k,s_k}$ to 1 satisfies the circuit. In particular, as $(v_{s_1}, \ldots, v_{s_k})$ induces a connected graph, for every $\emptyset \subsetneq P \subsetneq [k]$ there has to be an $i_1 \in P$ and $i_2 \in [k] \setminus P$ such that $v_{s_{i_1}}$ and $v_{s_{i_2}}$ are adjacent. This means that the node $e_{i_1,i_2,s_{i_1},s_{i_2}}$ has value 1, implying that c_P is 1 as well.

The reverse direction is also easy to show: it is clear that any satisfying assignment describes a k-tuple $(v_{s_1}, \ldots, v_{s_k})$ of vertices (possibly with repeated vertices) that forms a dominating set of the graph. If these vertices did not induce a connected graph, then they can be partitioned into two disconnected parts, that is, there is a $\emptyset \subset P \subset [k]$ such that there is no edge between v_{i_1} and v_{i_2} for any $i_1 \in P$ and $i_2 \in [k] \setminus P$. Then c_P has value 0, implying that O_3 has value 0, and hence the output has value 0 as well. □

(c) Since dominating set is $W[2]$-complete (as mentioned in the class), Part (a) proves connected dominating set is $W[2]$-hard. Part (b) proves that indeed connected dominating set is in $W[2]$. Thus by the definition of completeness, connected dominating set is $W[2]$-complete.

Problem 4. In the strongly connected Steiner subgraph problem, the input is a directed graph G, a set $K \subseteq V(G)$ of terminals, and an integer l; the goal is to find a strongly-connected subgraph of G with at most l vertices that contains every vertex of K. Prove that strongly connected Steiner subgraph is $W[1]$-hard by a parameterized reduction from multi-colored clique.

Solution.

Proof. We present a parameterized reduction from Multicolored Clique. Let $(G, k, (V_1, \ldots, V_k))$ be an instance of of Multicolored Clique. Let $E_{i,j}$ be the set of edges between V_i and V_j. We construct an instance (G', K, ℓ) of Strongly Connected Steiner Subgraph as follows (see Fig. 13.5).

(i) G' contains all the vertices of G, a vertex x, vertices $y_{i,j}$ $(1 \leq i < j \leq k)$, and a vertex w_e for every $e \in E(G)$. We define $E'_{i,j} = \{w_e \ : \ e \in E_{i,j}\}$ for $1 \leq i < j \leq k$.

(ii) Let $K := \{x\} \cup \{y_{i,j} \ : \ 1 \leq i < j \leq k\}$ and let $\ell := k + 1 + 2\binom{k}{2} = |K| + k + \binom{k}{2}$.

(iii) For every $1 \leq i \leq k$ and $v \in V_i$, we introduce the edges (x, v) and (v, x).

(iv) For every $1 \leq i < j \leq k$ and $e \in E_{i,j}$, we introduce the edges $(y_{i,j}, w_e)$ and $(w_e, y_{i,j})$.

(v) For every $1 \leq i < j \leq k$ and $e \in E_{i,j}$, if $a \in V_i$ and $b \in V_j$ are the endpoints of e, then we introduce the edges (a, w_e) and (w_e, b).

The intuitive idea of the reduction is the following. The solution can afford to select only one vertex from each of $V_1, \ldots, V_k$ ("vertex gadgets" describing k vertices of G) and one vertex from each $E'_{i,j}$ ("edge gadgets" describing $\binom{k}{2}$ edges of G). Each vertex w_e in $E_{i,j}$ has only one inneighbor and one outneighbor different from $y_{i,j}$, thus if w_e is part of the solution, then those two vertices have to be selected as well. Therefore, the state of each edge gadget forces a state on two vertex gadgets, implying that the k-vertex gadgets describe a k-clique in G.

Formally, let $v_1 \in V_1, \ldots, v_k \in V_k$ be a k-clique in G and let $e_{i,j}$ be the edge between v_i and v_j. We claim that the set $S := K \cup \{v_i \ : \ 1 \leq i \leq$

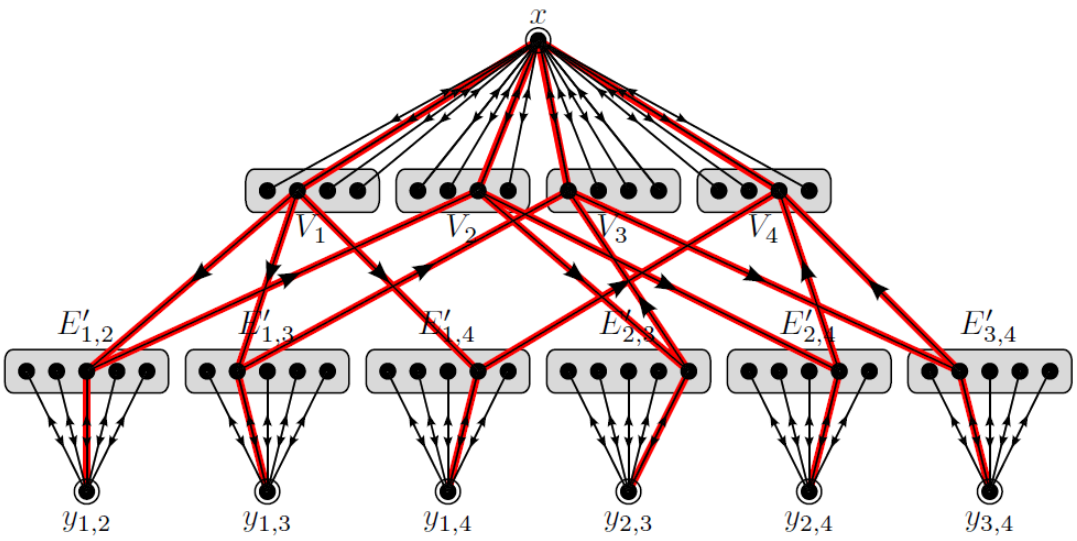

Fig. 13.5: An overview of the reduction in the proof of Theorem 13.33. The circled vertices are the terminals and the red edges show a possible solution. For clarity, we show only those edges of the graph between the V_i's and $E'_{i,j}$ that are in the solution.

$k\} \cup \{e_{i,j} \ : \ 1 \leq i < j \leq k\}$ induces a strongly connected subgraph $G'[S]$; note that $|S| = \ell$. For every $1 \leq i < j \leq k$, if e is the edge (v_i, v_j), then the directed closed walk $xv_iw_ey_{i,j}w_ev_jx$ uses only vertices of S. Moreover, these $\binom{k}{2}$ directed closed walks cover every vertex of S, hence $G'[S]$ is strongly connected.

For the reverse direction, let S be a set of size at most ℓ such that $K \subseteq S \subseteq V(G')$ and $G'[S]$ is strongly connected. Clearly, such a set has to include at least one outneighbor of each vertex in K, which means that S contains at least one vertex of $E'_{i,j}$ for every $1 \leq i < j \leq k$. All the inneighbors of $E'_{i,j} \cup \{y_{i,j}\}$ are in V_i, while all the outneighbors are in V_j, hence each of V_i and V_j has to contain at least one vertex of the solution. Taking into account the quota of $\ell = |K| + k + \binom{k}{2}$ on the size of the solution, this is only possible if S contains exactly one vertex $v_i \in V_i$ for every $1 \leq i \leq k$ and exactly one vertex $w_{e_{i,j}} \in E'_{i,j}$ for every $1 \leq i < j \leq k$. We claim that $v_1 \in V_1$, ..., $v_k \in V_k$ is a k-clique in G and $e_{i,j}$ is the edge between v_i and v_j. Suppose that v_i is not an endpoint of $e_{i,j}$. Then v_i is not an inneighbor of $w_{e_{i,j}}$. The only other inneighbor of $w_{e_{i,j}}$ is $y_{i,j}$ and the only inneighbor of $y_{i,j}$ in S is $w_{e_{i,j}}$ itself. Therefore, the strongly connected component of $G'[S]$ containing $w_{e_{i,j}}$ consists of only $y_{i,j}$ and $w_{e_{i,j}}$, a contradiction. The argument is similar if v_j is not an endpoint of $e_{i,j}$. Therefore, $e_{i,j}$ is the edge between v_i and v_j, hence $\{v_1, \ldots, v_k\}$ indeed induces a clique in G. □

Problem 5. Tree-diam(k), for $k \geq 3$, is the problem of deciding whether the diameter of an input graph G which is a tree is at least k. Prove that any single-pass streaming algorithm for Tree-diam(k) needs at least $\Omega(n)$ memory. [1]

Solution.

Before proving the problem, we need to prove the following theorem.

Theorem (LSUBSEQ). *Given a bit sequence x of size n, there is a single pass streaming algorithm $LSUBSEQ$ which uses $\mathcal{O}(\lg n)$ bits of memory outputs and*

(l, j) such that $x_k = 0, \forall k \in [j, j+l-1]$ and there is no $l' > l$ such that the previous property holds. In case of multiple solutions the algorithms picks the minimal j.

Proof of Theorem . It's quite easy to see how this problem can be solved by a single pass algorithm which mantains 3 counters (one for the start of the best sequence seen so far, one for length of the current subsequence of $0's$ and one for the best length seen so far). This requires $\mathcal{O}(\lg n)$ bits of memory.
■

Now we are ready to prove the statement of the problem.

We propose the following protocol for solving $INDEX(x, i)$: Alice starts with a graph containing only the dummy vertex 0, that is $G_0 = (V_0, E_0)$ with $V_0 = 0$ and $E_0 = \emptyset$. As she streams bit x_i, she builds graph $G_i = (G_i, E_i)$ with 2 choices depending on x_i:

- if $x_i = 0$ then $V_i = V_{i-1} \cup i$ and $E_i = E_{i-1} \cup (i-1, i)$.
- if $x_i = 1$ then $V_i = V_{i-1} \cup i$ and $E_i = E_{i-1} \cup (0, i)$.

[1] Hint: For this problem read notes for streaming algorithms and their lower bounds in advance of the class.

Intuitively, if $x_i = 0$ she connects a new vertex to a previous chain. Else she connects a new vertex to the dummy vertex 0. The resulting graph is a spider graph having vertex 0 as the root. Assume WLOG that G_n is not a chain (or else the problem would be trivial). Also assume WLOG that $x_1 = 1$ (or else we could stream the augmented bit string $1x$ of length $n+1$). In parallel, Alice runs algorithm $LSUBSEQ(x)$. After streaming the whole input x, Alice sends the resulting memory image $Mem(G_n)$ of the graph constructed so far together with the output of $LSUBSEQ(x)$ (which takes only $\mathcal{O}(\lg n)$ bits) to Bob. Given that we want to prove a lower bound of $\mathcal{O}(n)$ bits we can afford the extra $\mathcal{O}(\lg n)$ bits. Bob receives the memory image of the graph streamed $Mem(Gn)$ so far together with the pair (l, j) which is the solution of $LSUBSEQ(x)$. Bob uses *TREE - DIAM(k)* to find the diameter of G_n. Note than in order for Bob to not alter his memory image, he creates an auxiliary copy of the memory image he has so far and runs *TREE - DIAM(k)* on it. Let the diameter that he finds be d (by iterating k over all possible values). Given that G_n is not a chain, the diameter of G_n would be composed by adding the lengths of the 2 longest branches of G_n. We already know l, which is the length of the longest branch of G_n. Therefore, we are able to get $l' = d - l$ which is the length of the second-longest branch of G_n. There are 3 cases to consider:

- if $i \in [j, j + l - 1]$ then clearly $INDEX(x, i) = 0$ (vertex i is strictly part of a branch in G_n).
- if $i = j - 1$ then clearly $INDEX(x, i) = 1$ (vertex i is the start of a new branch in G_n).
- we add a chain of length $l' - 1$ to vertex i, in the same manner as for proving the lower bound for Problem 4 .

For the third case, Bob runs *TREE-DIAM(k)* again to find the diameter for this new graph. The key point to note is that the diameter increases iff $INDEX(x, i) = 0$. Note that if $INDEX(x, i) = 1$ then adding a chain of length $l' - 1$ to vertex i will not increase the diameter. Let $c_{l'-1}$ be the end point of the chain added to vertex i, and let e_i be the last vertex on the branch containing i (before the chain was added). Clearly, $d_{G_n}(0, c_{l'-1}) = l'$. Also, $d_{G_n}(ei, cl' - 1) = l' - 1 + d_{G_n}(i, e_i) \leq l' - 1 + l - 1 \leq d$. On the other hand, if $INDEX(x, i) = 0$ then adding a chain of length $l' - 1$ to i would create a path of length $d_{G_n}(0, c_{l'-1}) \geq l' - 1 + 2 \geq l' + 1$ because $d_{G_n}(0, i) \geq 2$. Given that there is another branch of length l in G_n (by assumption of the three cases we considered) we would get a diameter of at least $l' - 1 + 2 + l \geq l' + l + 1 > d$. Therefore, we proved that we can use *TREE-DIAM(k)* to solve $INDEX(x, i)$. Therefore, Problem 5 has a lower bound of $\mathcal{O}(n)$ bits of working memory. ■

BILL'S WRITEUP WITH QUESTIONS

Assume, by way of contradiction, that there is a streaming algorithm for Tree-diam(k) with $o(n)$ memory.

QUESTION ONE I first thought that for each k you could have a different algorithm. From what you say later it seems as though you need to have the algorithm stream and store the same thing in memory whether $k = 1$ or $k = 2$ or $\cdots$ or $k = n$.

Recall the communication complexity problem $INDEX(x, i)$: Alice has x, a string of n bits, Bob has $i \in \{1, \ldots, n\}$, Alice sends a message to Bob, and Bob then knows x_i. It is known that $INDEX(x, i)$ requires $\geq n$ bits of communication.

We show that if there is an $o(n)$ streaming algorithm for Tree-diam(k) then there is an $o(n)$ commnication protocol for $INDEX(x, i)$.

1. Alice gets $x \in \{0, 1\}^n$. Bob gets $i \in \{1, \ldots, n\}$

2. If $x = 0^n$ then Alice sends 0. If $x = 1^n$ then Alice sends 1. (In all later steps we will assume $x \notin \{0^n, 1^n\}$.)

3. Alice (who is all powerful) finds the longest string of 0's in x (if there is a tie take the one that appears earliers in x). Let the string be $x_j, x_{j+1}, \ldots, x_{j+\ell-1}$ Note that j, ℓ are of length $\lg n$ (Alice will later tell them to Bob which is fine since they are very short).

   QUESTION TWO You have Alice find (ℓ, j) by streaming x. This seems unneeded since Alice is all powerful.

4. Alice creates the graph $G = (V, E)$ as follows.

   (a) $V = \{0, \ldots, n\}$. We think of 0 as a dummy vertex and, for $1 \leq j \leq n$, j is associated to x_j.

   (b) For all $1 \leq j \leq n$ if $x_j = 0$ then put in edge $(j - 1, j)$, else put in edge $(0, j)$.

5. Alice runs the Tree-diam streaming algorithm on G and notes the string y of length $o(n)$ which is in the memory at the end. Note that from y (1) one can find if the tree diam is 1, 2,..., n, and hence can find the Tree-diam, and (2) one can add edges and vertices to the graph and find the Tree-diam of the new graph.

6. Alice gives Bob (ℓ, j, y). This is of length $o(n)$.

7. If $i \in \{j, j + 1, \ldots, j + \ell - 1\}$ then Bob nows $x_i = 0$.

8. If $i = j - 1$ or $j = j + \ell$ then $x_i = 1$.

9. Bob uses y to compute the tree-diam of G. Let it be d. Note that the longest path in G is formed by the edges arising from the largest and second largest sequences of 0's in x. Lets see how long that is. We know the longest sequence is of length ℓ. Lets say it starts at x_j. So $x_j = 0$ and $x_{j-1} = 1$.

   Then we have the following edges

   $(0, j - 1)$, $(j - 1, j)$, ..., $(j + \ell - 2, j + \ell - 1)$. ($\ell + 1$ edges)

Lets say the second largest sequence of 0's is of length ℓ' and begins with $x_{j'}$. Then we have the following edges:

$(0, j'-1)$, $(j'-1, j')$, ..., $(j'+\ell'-2, j'+\ell'-1)$. ($\ell'+1$ edges)

If you put these together you get a path with $\ell+\ell'+2$ edges.

So the diameter is $d = \ell+\ell'+2$.

QUESTION THREE You have the diamter as $\ell+\ell'$.

Since Bob knows d and ℓ he can compute $\ell' = d-\ell-2$.

10. Recall that Bob wants to know x_i. Bob uses the y and the streaming algorithm to find the diameter of the graph G' which is G with a chain of length $\ell'+X$ starting at vertex i.

    QUESTION You have $\ell'-1$. I think $\ell'+1$ is correct. Read on and I will explain.

11. *Case 1* $x_i = 0$. Then there is an edge from $i-1$ to i. Go back until you get to a 1, so you will have edges

    $(0,k)$ $(k, k+1)$, ... (i, c_1), ..., $(c_{\ell'+X-1}, c_{\ell'+X})$.

    NOT GOING TO BOTHER, IT WORKS OUT.

    and then a chain of length $\ell'+1$. There is also an edge from 0 to j and then a chain of length ℓ. Hence there is a path of length