CS 170 HW 11

Daniel Deng, SID 3034543526

1 Study Group

1. Auston Lin, SID 3034554056
2. Yes

2 NP Basics

1. No information on B
2. A is in P
3. B is NP-hard
4. No information on A

3 Runtime of NP

False. While P is NP-complete, and that all problems in NP can be reduced to it in polynomial time, by fixing k, we cannot guarantee that the reduction step, which runs in some arbitrary polynomial time $O(n^i)$ time, will not have $i > k$, in which case some problem in NP will have runtime $O(n^i)$ instead of $O(n^k)$

4 Dominating Set

First we show that the Minimum Dominating Set (MDS) problem is NP. Given a solution set of size $\leq k$ to the graph $G = (V, E)$, we can easily verify that all vertices not in the solution set has neighbors in the solution set by checking the edges, which is $O(|V| + |E|)$ time. Therefore, MDS is NP.

Then, we show that MDS problem is NP-complete because the Vertex Cover (VC) problem, which is NP-complete, reduces to it. Given graph $G = (V, E)$, construct a new graph G' such that for each pair of vertices u, v such that $(u, v) \in E$ we add a dummy vertex w that connects both u and v (to ensure that all edges in the original graph are accounted for, as running MDS on the original graph could result in some neighbors having edges between them but not included in the DS). Suppose MDS finds a solution set OF SIZE $\leq k$, we can process it to ensure that the set if free of dummy vertices by switching each w with either the corresponding u or v, and still have the same dominating effect (since they are in a triangular relationship). By definition of a Dominating Set, it is apparent that all edges are incident of the resulting dominating set containing only original vertices, and is thus an instance of Vertex Cover with size $\leq k$. Therefore, VC can be reduced to MDS, and MDS is NP-complete.

5 More Reductions

(a) To reduce Subset Sum to Partition, we first find the absolute different

$$d = \left| \sum_n a_n - 2k \right|$$

and append d to A. Then, if Partition determines that there exist partitions of equal subset sum, there must exists a subset $P \in [n]$ that is a solution to Subset Sum with integer k. Otherwise, the answer is false.

To prove the correctness of the reduction, we consider two cases:

- $d \leq 2k$. By adding d, the Partition algorithm will find a partition such that both subsets sum up to k. Since we only added 1 extra element d to the list, one of the subsets must sum up to k with only original elements, and thus satisfies the requirement for Subset Sum.
- $d > 2k$. By adding d, the Partition algorithm will find a partition such that both subsets sum up to $k + d$. Since we added only 1 extra element, d must exist in exactly one of the subsets, and removing d from that subset will result in a subset that sums to k, which satisfies Subset Sum.

The runtime of the reduction is $O(n)$ because we can compute the sum of the sequence is linear time and computing the absolute difference is constant time. Appending to A should also be constant time.

(b) To reduce Subset Sum to Knapsack, we would simply run Knapsack with parameters $w_i = v_i = a_i$ and $W = V = k$. The output of the Knapsack instance is the output of the Subset Sum instance.

This reduction is true because, given the parameter, the Knapsack instance will determine the existence of a P such that

$$\sum_{i \in P} w_i = \sum_{i \in P} a_i \leq W = k$$
$$\sum_{i \in P} v_i = \sum_{i \in P} a_i \geq V = k$$

which is the same as asking if

$$\sum_{i \in P} a_i = k$$

This is the exactly same problem statement as Subset Sum.

The runtime of the reduction algorithm is $O(1)$ since we directly feed some parameter into Knapsack without any modification.

6 Orthogonal Vectors

To show that there is a $O(2^{cn/2}m)$-time algorithm for 3-SAT, we need to show that 3-SAT reduces to the Orthogonal Vectors (OV) problem. First split the n variables of 3-SAT into two sets of equal size, and generate all possible assignments for the variables in the two sets (there will be a total of $2^{n/2}$ combinations for the boolean variables in each set). Then, for each subset, determine if the the m clauses can be satisfied by the assignment of half the variables alone, and represent the results for the clause evaluation in vectors V such that

$$V[i] = \neg\,\text{boolean(clause } m_i \text{ is satisfied by the partially assigned variables)}$$

This would result in $2^{n/2}$ vectors that represent the clauses evaluated on different partial assignments for each subset of variables, which we will pass into OV as sets A and B. Based on the way we constructed the vectors, the only way for a clause to be satisfied is for $a_i b_i = 0$, and 3-SAT is true if and only if all clauses are true (i.e., $a \cdot b = 0$). Therefore, OV is true under the given parameters implies that 3-SAT is true.

It takes $O(2^{n_{3-SAT}/2})$ time to assign the variables, evaluate the clauses, and construct the vectors. Since the size of n_{OV} for OV is $2^{n_{3-SAT}/2}$, the algorithm for OV will run in $O(2^{cn_{3-SAT}/2}m)$. The overall runtime of the algorithm for 3-SAT will therefore be $O(2^{cn_{3-SAT}/2}m)$.