CS170–Fall 2022 — Homework 11 Solutions

Deming Chen cdm@pku.edu.cn

Last Modified: December 28, 2022

Collaborators: NONE

2 Some Sums

(a) Subset Sum $\rightarrow$ Partition

Suppose $a_1 + a_2 + \cdots + a_n = s$. There are two cases. First, if $s \leqslant 2t$, then let $a_{n+1} = 2t - s \geqslant 0$. Consider the partition of $A \cup \{a_{n+1}\}$. If there is a way to partition $A \cup \{a_{n+1}\}$ into two disjoint subsets B and C such that the sum of the elements in each subset equal, i.e.

$$\sum_{b \in B} b = \sum_{c \in C} c = t, \tag{1}$$

the subset which does not contain a_{n+1} is just the solution of the original Subset Sum problem. In addition, if B is a solution of the original problem, then $A \cup \{a_{n+1}\}$ can be divided into B and $A \cup \{a_{n+1}\} \backslash B$.

The second case is when $s > 2t$. We now add $a_{n+1} = s - 2t$ into A. If there is a way to partition $A \cup \{a_{n+1}\}$ into two disjoint subsets B and C such that the sum of the elements in each subset equal, i.e.

$$\sum_{b \in B} b = \sum_{c \in C} c = s - t, \tag{2}$$

the subset which contains a_{n+1} is just the solution of the original Subset Sum problem (of course we need to exclude a_{n+1}). In addition, if B is a solution of the original problem, then $A \cup \{a_{n+1}\}$ can be divided into $B \cup \{a_{n+1}\}$ and $A \backslash B$.

Computing s takes time $\mathcal{O}(n)$ and other operations take time $\mathcal{O}(1)$, so the reduction is linear.

(b) Subset Sum $\rightarrow$ Knapsack

Construct n items such that $w_i = v_i = a_i, i = 1, 2, \cdots, n$. Let $W = V = t$, so $\sum_{i \in P} a_i \leqslant t$ and $\sum_{i \in P} a_i \geqslant t$ implies that $\sum_{i \in P} a_i = t$.

3 Max k-XOR

(a) Suppose undirected graph $G = (V, E)$ has n vertices and m edges. Consider n variables $x_1, x_2, \cdots, x_n$. Every edge (v_i, v_j) corresponds to a clause $x_i \oplus x_j$. There is a one-to-one mapping from a cut of G to a valid assignment of variables: given $V = V_1 \cup V_2$, $x_i =$ `true` if and only if $v_i \in V_1$. Then, an edge crosses the cut means the value of its corresponding clause is true (one of variables is `true` and the other is `false`).

Therefore, the number of edges crossing the cut equals the number of true clauses.

(b) To reduce Max 3-XOR to Max 4-XOR we simply add a new variable x_{n+1} to each clause.

If there is some assignment of variables that satisfies at least r clauses in 3-XOR clauses, we just set $x_{n+1} =$ `false`. Then these variables satisfy at least r clauses in 4-XOR clauses.

If there is some assignment of variables that satisfies at least r clauses in 4-XOR clauses, there are two cases. The first is when $x_{n+1} =$ `false`. Ignoring x_{n+1} we get an assignment of variables that satisfies at least r clauses in 3-XOR clauses. The second is when $x_{n+1} =$ `true`. Changing all `true` variables into `false` and all `false` variables into `true` we get back to the first case.

4 Dominating Set

To reduce Set Cover to Min Dominating Set, we consider a graph $G = (V, E)$, where each vertex represents an element in one of these sets. $(u, v) \in E$ if and only if u and v both in some set. Different sets correspond to different cliques in graph G.

If there is a dominating set $V' \subset V$ of size $\leqslant k$, each vertex $v \in V'$ is in a clique which represents a set. There are at most k such sets, since some may be represented more than once. These sets can cover all elements. For arbitrary element x represented by v, $v \in V'$ implies that x is covered by some set; if $v \notin V'$, then one of v's neighbors u is covered by some set. Since $(u, v) \in E$, v also be covered by the same set.

If there are at most k sets which cover all elements, pick a vertex from each clique which represents one of such sets. Since every vertex is in some clique and in each clique some vertex is in S, S is a dominating set.

Because Set Cover is an NP-complete problem, Min Dominating Set is also NP-complete.

5 Orthogonal Vectors

Split the variables in the 3-SAT problem into two groups $X = \{x_1, x_2, \cdots, x_{\frac{n}{2}}\}$ and $Y = \{y_1, y_2, \cdots, y_{\frac{n}{2}}\}$. Label all clauses by $c_1, c_2, \cdots, c_m$. Let C be the set of all clauses. We construct two sets of vectors A, B, each of which is of size $2^{\frac{n}{2}}$. Each vector in A corresponds to an assignment of variables $x_1, x_2, \cdots, x_{\frac{n}{2}}$, and each vector in B corresponds to an assignment of variables $y_1, y_2, \cdots, y_{\frac{n}{2}}$. Each dimension of a vector corresponds to a clause.

For $\mathbf{a} = (a_1, a_2, \cdots, a_m)^T$ corresponding $x_1, x_2, \cdots, x_{\frac{n}{2}}$, if c_i has nothing to do with variables in X, then $a_i = 1$; otherwise, $a_i = 1$ if and only if c_i can always be satisfied by the literals from X (even if the remaining literals from Y are `false`). For instance, $c_i = (x_1, \bar{x_2}, y_1)$. If $x_1 =$ `true` or $x_2 =$ `false`, then $a_i = 0$. Otherwise, $a_i = 1$. The vectors in $|B|$ are constructed similarly.

By this rule, these variables satisfy all clauses if and only if their corresponding vectors are orthogonal. Since $|A| = |B| = 2^{\frac{n}{2}}$, the 3-SAT problem can be solved in time $\mathcal{O}((2^{\frac{n}{2}})^c m) = \mathcal{O}(2^{cn/2} m)$.