Problem Set 9

Due date: Electronic submission of this homework is due on **4/14/2023** on canvas.

Name: Ayushri Jain

Resources.

Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms, 3rd edition, The MIT Press, 2009 (or 4th edition)

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to prepare this homework.

Signature: AJain

Problem 1. (30 points)

Solution. (a) If we show that there is a polynomial time verifier for DSAT, then we can say that DSAT is in NP. If we have a Boolean formula B, we can find whether an assignment of true/false values to B satisfies B or not in polynomial time. In this case, our certification will be to check if 2 assignments exist for which evaluates B to true and the two assignments are different. We can check if both satisfying assignments are different or not in linear time. If we get a valid certificate, our verifier will accept it in polynomial time. If we get an invalid certificate, our verifier will reject it in polynomial time. Therefore, DSAT is in NP.
(b) Let us assume a boolean formula B for 3SAT problem and a new variable x which is not part of B. Now, we will add $x \vee \neg x$ to each clause of B to get a new boolean formula B'. If B was unsatisfiable, then some clause of B must have been $false$. So, B' is also unsatisfiable. Otherwise, if B was satisfiable for some combination of assignment values then 1. we can take those assignment values as it is and keep $x = true$: all clauses in B' are satisfiable; 2. we can negate those assignment values and keep $x = false$: again, all clauses in B' are satisfiable because all of them contain $x \vee \neg x$ in which $\neg x$ is true this time. Thus, there are atleast 2 satisfying assignments for the boolean formula B. This reduction is done in polynomial time(size of B and B' is polynomial). Hence, 3SAT $\leq_p$ DSAT.
Using (a) and (b), we can say that DSAT is NP complete.

Problem 2. (20 points)

Solution. (a) According to theorem 34.11, each literal present in every clause will represent a vertex in graph. So, we will have the following vertices in graph $x_1, \neg x_2, \neg x_3, \neg x_1, \neg x_2, x_3, \neg x_1, x_2, \neg x_3$. Total 9 vertices. Then we need to draw an edge from one literal x of a clause to all other literals $y_1, y_2, y_3 \cdots$ of remaining clauses if the literals x and y_i are not negations of each other. We do not add any edge between literals of same clause. So, we will have the following edges in graph -
For literals of clause $1(x_1 \vee \neg x_2 \vee \neg x_3)$:

- x_1 - $\neg x_2$
- x_1 - x_3
- x_1 - x_2
- x_1 - $\neg x_3$
- $\neg x_2$ - $\neg x_1$ (of clause 2)
- $\neg x_2$ - $\neg x_2$
- $\neg x_2$ - x_3
- $\neg x_2$ - $\neg x_1$ (of clause 3)

- $\neg x_2$ - $\neg x_3$
- $\neg x_3$ - $\neg x_1$ (of clause 2)
- $\neg x_3$ - $\neg x_2$
- $\neg x_3$ - $\neg x_1$ (of clause 3)
- $\neg x_3$ - x_2
- $\neg x_3$ - $\neg x_3$

For literals of clause 2 ($\neg x_1 \vee \neg x_2 \vee x_3$):

- $\neg x_1$ - $\neg x_2$
- $\neg x_1$ - $\neg x_3$ (of clause 1)
- $\neg x_1$ - $\neg x_1$
- $\neg x_1$ - x_2
- $\neg x_1$ - $\neg x_3$ (of clause 3)
- $\neg x_2$ - x_1
- $\neg x_2$ - $\neg x_2$
- $\neg x_2$ - $\neg x_3$ (of clause 1)
- $\neg x_2$ - $\neg x_1$
- $\neg x_2$ - $\neg x_3$ (of clause 3)
- x_3 - x_1
- x_3 - $\neg x_2$
- x_3 - $\neg x_1$
- x_3 - x_2

For literals of clause 3 ($\neg x_1 \vee x_2 \vee \neg x_3$):

- $\neg x_1$ - $\neg x_2$ (of clause 1)
- $\neg x_1$ - $\neg x_3$
- $\neg x_1$ - $\neg x_1$
- $\neg x_1$ - $\neg x_2$ (of clause 2)
- $\neg x_1$ - x_3
- x_2 - x_1

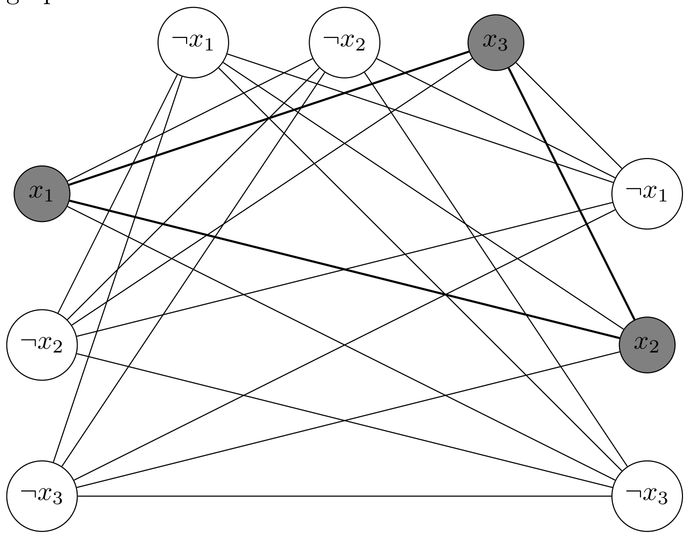
- x_2 - $\neg x_3$
- x_2 - $\neg x_1$
- x_2 - x_3
- $\neg x_3$ - x_1
- $\neg x_3$ - $\neg x_2$ (of clause 1)
- $\neg x_3$ - $\neg x_3$
- $\neg x_3$ - $\neg x_1$
- $\neg x_3$ - $\neg x_2$ (of clause 2)

(Please note that edges are repeated above for simplicity : e.g. x_2 - x_3 is same as x_3 - x_2. However, in the graph there will be only 1 edge representing both (undirected graph).
(b) If we take x_1 = true, x_2 = true, and x_3 = true, then clause values are -

$x_1 \vee \neg x_2 \vee \neg x_3$ - true because x_1 is true

$\neg x_1 \vee \neg x_2 \vee x_3$ - true because x_3 is true

$\neg x_1 \vee x_2 \vee \neg x_3$ - true because x_2 is true
So, the entire instance $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$ value is also true. Hence, x_1 = true, x_2 = true, and x_3 = true is one satisfying assignment. This corresponds to a clique of size 3 with three vertices x_1, x_2, x_3 and edges between them : x_1 - x_3, x_2 - x_3, x_1 - x_2. Clique is highlighted in the graph below.



Problem 3. (30 points)

Solution. (a) No, (S, t) is not a yes-instance for given $S = \{2, 3, 5, 7, 8\}$ and $t = 19$. Below are all the subsets and their corresponding sums -

{} sum : 0

{2} sum : 2

{3} sum : 3

{5} sum : 5

{7} sum : 7

{8} sum : 8

{2,3} sum : 5

{2,5} sum : 7

{2,7} sum : 9

{2,8} sum : 10

{3,5} sum : 8

{3,7} sum : 10

{3,8} sum : 11

{5,7} sum : 12

{7,8} sum : 15

{5,8} sum : 13

{2,3,5} sum : 10

{2,3,7} sum : 12

{2,3,8} sum : 13

{2,5,7} sum : 14

{2,5,8} sum : 15

{2,7,8} sum : 17

{3,5,7} sum : 15

{3,5,8} sum : 16

{3,7,8} sum : 18

{5,7,8} sum : 20

{2,3,5,7} sum : 17

{2,3,5,8} sum : 18

{2,3,7,8} sum : 20

{2,5,7,8} sum : 22

{3,5,7,8} sum : 23

{2,3,5,7,8} sum : 25
None of the subsets has sum 19, so it is not a yes-instance.
(b) The brute force algorithm for this problem is to get every subset of the set S and check if sum of its elements is equal to t which was done in (a). If a set has n elements, then it will have 2^n subsets which is exponential and in the worst case, we will have to check all subsets. Therefore for larger sets S, the brute force algorithm will take exponential time, so it is not feasible.
(c) The time complexity of dynamic programming algorithm for SUBSET SUM problem is proportional to nt where n is the number of elements in set S and t is the expected sum of subsets in S. So the algorithm seems to be polynomial in terms of n and t. However, the input size is actually comparable to the number of bits used to represent both B (if B contains $a_1, a_2, \cdots, a_n$, then a_i requires $\log a_i$ bits, so B requires $\sum_{i=1..n}(\log a_i)$ bits) and t ($\log t$ bits). So, nt is actually exponential in terms of number of bits. When n and t are very large, the algorithm is exponential.

Problem 4. (20 points)

Solution. To show that the set-partition problem is NP-complete, we need to show that it is in NP, and SUBSET SUM $\leq_p$ SET PARTITION.
1. First, we need to prove that SET PARTITION is in NP. If we have a set S and two subsets A and $\bar{A}$, then we can easily check if the sum of the elements in A is equal to the sum of the elements in $\bar{A}$ in polynomial time. This will be our certificate, it will be valid if sum are equal and invalid if sums are not equal which we will be able to verify in $O(|S|)$ time. So, set-partition problem is in NP.
2. Let's assume we have a set S and target sum is t for the SUBSET SUM problem. Let s denote the sum of all elements of set S and subset A represents the solution subset of the SUBSET SUM problem, i.e. $\sum_{a\in A} a = t$. So, for the complementary subset $\bar{A}$, we will have $\sum_{a\in\bar{A}} a = s - t$. Now, we will create another set $S' = S \cup \{|s - 2t|\}$ which will be input to the SET PARTITION problem. Sum of elements of $S' = s + s - 2t = 2s - 2t$. If $t \leq s - t$, then we will add element $s - 2t$ in A. So sum of A will be $\sum_{a\in A} a = t + (s - 2t) = s - t$ which is same as $\sum_{a\in\bar{A}} a$. In the other case, when $t > s - t$, then we will add element $2t - s$ in $\bar{A}$. So sum of $\bar{A}$ will be $\sum_{a\in\bar{A}} a = s - t + (2t - s) = t$ which is same as $\sum_{a\in A} a$. Hence we have reduced SUBSET SUM problem to SET PARTITION problem, i.e. SUBSET SUM $\leq_p$ SET PARTITION.
Using 1 and 2, we can say that SET PARTITION problem is NP complete.