Homework 9

Thomas Kim tsk389 51835

David Munoz dam2989 51840

CS331 Algorithms and Complexity

Problem Q1(a). $4-SAT$: Each clause has exactly 4 variables.
Prove or disprove: $4-SAT$ is NP-Complete
Claim: $4-SAT$ is NP-Complete
Poly-time reduction

1: Let Φ be an expression of a 3-SAT problem
2: Let Φ' be an initially empty expression
3: **for** $\forall C_j = (x \vee y \vee z) \in \Phi$ where x, y, z are literals **do**
4: Create a new boolean variable α
5: $C_j^1 := (x \vee y \vee z \vee \alpha)$
6: $C_j^2 := (x \vee y \vee z \vee \overline{\alpha})$ $\Phi' := \Phi \wedge C_j^1 \wedge C_j^2$
7: **end for**
8: **return** $4-SAT(\Phi')$

Proof of Correctness

Proof. Take $(x \vee y \vee z \vee \alpha) \wedge (x \vee y \vee z \vee \overline{\alpha})$
$= ((x \vee y \vee z) \vee \alpha) \wedge ((x \vee y \vee z) \vee \overline{\alpha})$
$= ((x \vee y \vee z) \wedge (x \vee y \vee z)) \vee ((x \vee y \vee z) \wedge \overline{\alpha}) \vee (\alpha \wedge (x \vee y \vee z)) \vee (\alpha \wedge \overline{\alpha})$
$= (x \vee y \vee z) \vee ((x \vee y \vee z) \wedge \overline{\alpha}) \vee (\alpha \wedge (x \vee y \vee z))$
$= (x \vee y \vee z) \vee (((x \vee y \vee z) \wedge \overline{\alpha}) \vee (\alpha \wedge (x \vee y \vee z)))$
$= (x \vee y \vee z) \vee ((x \vee y \vee z) \wedge (\overline{\alpha} \vee \alpha))$
$= (x \vee y \vee z) \vee ((x \vee y \vee z) \wedge T)$
$= (x \vee y \vee z) \vee (x \vee y \vee z)$
$= (x \vee y \vee z)$
Therefore, $(x \vee y \vee z \vee \alpha) \wedge (x \vee y \vee z \vee \overline{\alpha}) = (x \vee y \vee z)$
By line 5-6, every clause $C_j \in \Phi$ is transformed into an equivalent set of two clauses $C_j^1, C_j^2 \in \Phi'$
Now, by the associative property, $C_1^1 \wedge C_1^2 \wedge C_2^1 \wedge C_2^2 \cdots \wedge C_n^1 \wedge C_n^2 = (C_1^1 \wedge C_1^2) \wedge \cdots \wedge (C_n^1 \wedge C_n^2)$
By the previous proof, we have established $(C_j^1 \wedge C_j^2) = C_j$
Therefore, $\Phi' = C_1 \wedge C_2 \wedge \cdots \wedge C_n$
Therefore, $\Phi' = \Phi$ which means $4-SAT$ can solve any 3-SAT problem Φ by using the above reduction.
□

Proof of Runtime

Proof. Let the number of clauses in C_j be expressed as N The reduction processes each clause $C_j \in \Phi$ once and generates two new clauses C_j^1, C_j^2
Therefore, the runtime is $\theta(2N) = \theta(N)$ which is polynomial in N
Therefore, if the reduction is correct, $3-SAT \leq_p 4-SAT$
□

Problem Q1(b). $T-SAT$: All variables appear in non-negated form only
Prove or disprove: $T-SAT$ is NP-Complete
Claim: $T-SAT$ is not NP-Complete
T-SAT Algorithm

1: **return** 1

Proof of correctness:

Proof. Let all variables be assigned T
Each clause is of the form $(x_1 \vee x_2 \vee \cdots \vee x_k)$

When every variable is assigned T, this reduces to:
$(T \vee T \vee \cdots \vee T)$
So every clause evaluates True.
Now $\Phi = C_1 \wedge C_2 \wedge \cdots \wedge C_n$
Since every clause is True, this becomes:
$\Phi = T \wedge T \wedge T \cdots \wedge T$
This trivially evaluates to True.
Therefore, every T-SAT expression is satisfiable.
The algorithm returns 1 by line 1, so it is correct.

□

Proof of runtime:

Proof. The runtime is constant, since there is only 1 line and it is a return statement

□

Problem Q1(c). $P-SAT$: Each clause has precisely 3 variables, and each variable appears in exactly 3 clauses.
Prove or disprove: $P-SAT$ is NP-Complete **Claim**: $P-SAT$ is always satisfiable

Proof. Claim: For an expression in n variables, there are n clauses.
Given n variables each appearing 3 times, there are $3n$ literals.
Given $3n$ literals and 3 literals per clause, there are n clauses.

Claim: There exists a one-to-one and onto mapping F between clauses and variables with the following property:
Each variable v is mapped to a clause containing a literal v or $\overline{v}$
Each variable v appears 3 times in negated or non-negated form

This means F can map $v \to C_i$, where i can take on 3 distinct values
Suppose there is no one-to-one and onto mapping.
This means all 3 clauses must be mapped to other variables.
By the pigeonhole principle, after mapping each clause to precisely 1 variable, there will be one left over.
Construct an undirected graph as follows:
1. Each node represents a clause.
2. Each edge represents a shared variable between clauses.

Now, suppose the leftover clause is part of a connected component of this graph which is disjoint from the component containing v
Note that there are k variables and $k+1$ clauses associated with this connected component
This is clear because only one variable is unmapped, so k variables map to k clauses and the leftover clause can only map to those k variables.
This is a contradiction, as there are $3k+3$ literals with only k variables.
Therefore, the leftover clause must be part of a connected component containing v
This connected component, by definition, must contain a path from the leftover variable to some clause containing v
For each edge $e = (C_l, C_p)$ on this path, take the leftover clause C_l and replace the clause it is connected with on the path C_p to create a new mapping $F'(F(C_p)) = C_l$, creating a new leftover clause $C'_l = C_p$.

At the end of this path, the last leftover clause must be a clause which uses v, and thus can be mapped to v
At each step of this iteration, a new valid mapping is created to replace C_p, so the resulting mapping is one-to-one and onto

For each clause C, take the literal l corresponding to the variable $F^{-1}(C)$
Simply satisfy this literal and C is satsified.
There can be no conflicting assignments, because the mapping is one-to-one and onto.
Therefore, every instance of $P - SAT$ is satisfiable.
Therefore, $P - SAT$ can be decided in constant time, simply return 1 every time.

□

Problem Q2: 8.16. *Intersection Inference Problem*:
Given a finite set U of size n and collection $A_1, \ldots, A_m$ of subsets of U, cardinalities $c_1, \ldots, c_m$, does there exist a set $X \subset U$ such that $X \cap A_i = c_i$?
Prove NP-Completeness

Let 3 disjoint sets X, Y, Z of size n and $T \subset X \times Y \times Z$ be an instance of 3D matching
Let $m = |X \cup Y \cup Z|$
Let U be an initially empty set
Let $A_1, \ldots, A_m$ be initially empty sets
Let $c_1, \ldots, c_m = 1$
$U := T$
for $\forall v_i \in X \cup Y \cup Z$ **do**
  for $\forall t_j = (x, y, z) \in T$ **do**
    if $v_i = x \vee v_i = y \vee v_i = z$ **then**
      $A_i := A_i \cup t_j$
    end if
  end for
end for
return $Intersection - Inference(U, A_1, \ldots, A_m, c_1, \ldots, c_m)$

Proof of correctness:

Proof. Suppose there exists some X which fulfills the Intersection Inference condition.
By construction, $X \subset T$
Claim: X contains at most one triple (x, y, z) for each unique value of x, y, z
Suppose two triples in X contain the same element.
Without loss of generality, suppose this element is x
Find the set A_i containing all triples that include x
Now, $|X \cap A_i| = 2$ because two triples contain x, both of which are in A_i and both of which are in X
Contradiction.
Claim: X contains a triple containing each unique value at least once Suppose X does not contain a triple that includes some unique value v.
Find A_i corresponding to v
$|X \cap A_i| = 0$ because A_i contains only triples which include v
Contradiction Therefore, X contains each element exactly once.

Suppose there does not exist any X which fulfills the Intersection Inference condition.

Claim: There does not exist any 3D matching
Suppose there existed a 3D matching.
Take each triple t_i in the 3D matching and let $X = \{t_i\}$
By the 3D matching problem statement, X contains precisely 1 triple for every value $v \in X \cup Y \cup Z$
Take any arbitrary A_i
$|X \cap A_i| = 1$, because precisely one triple in X_i contains the value associated with A_i
Contradiction.

□

Proof of runtime:

Proof. Let n be the cardinality $|X \cup Y \cup Z|$
Let m be the cardinality $|T|$
For each element of $X \cup Y \cup Z$, the reduction checks each t_i
All other operations are constant time.
Therefore, the reduction has runtime $\theta(mn)$
Let $N = max(|X|, |Y|, |Z|)$
We see $n = |X \cup Y \cup Z| \leq 3N$
Also, $m = |T| \leq N^3$
So the runtime as a function of the largest set is $\theta(N^4)$, which is polynomial. □

Problem Q3: 8.17. *Zero-Weight-Cycle Problem*:
Given a directed graph $G = (V, E)$ with weights $w_e \in \mathbb{Z}$, does there exist a simple cycle C such that $\sum_{e \in C} w_e = 0$?
Prove NP-Completeness
Proof of NP

Proof. Let the certificate be some cycle $c = \{e_1, e_2, \ldots, e_n\}$ in G
To verify whether the solution is valid, check whether $\sum_{i=1}^{n} w_i = 0$

□

Reduction from Subset-Sum to Zero-Weight-Cycle

Let $T, A = \{a_1, a_2, \ldots, a_n\}$ be an instance of the subset-sum problem with target sum T and set A
Let $G = (V, E)$ be an initially empty graph
Let $e = (x, y)_w$ be a directed edge from x to y with weight w
$V := \{s, v_1, v_2, \ldots, v_n\}$
for $\forall a_i \in A$ **do**
    for $\forall 1 \leq j < i$ **do**
        $E := E \cup \{(v_j, v_i)_{a_i}\}$
    end for
    $E := E \cup \{(v_i, s)_{-T}\}$
end for
return $Zero - Weight - Cycle(G)$

Proof of correctness:

Proof. Claim: Any cycle $\{e_1, e_2, \ldots, e_n\}$ contains exactly one or zero edges corresponding to each weight $a_i \in A$ Suppose 2 edges in the cycle have the same weight a_i.
This means 2 edges entered v_i, as only edges incoming to v_i have weight a_i
This means v_i was used at least twice, thus it is not a simple cycle.

As a corollary, this means every cycle corresponds precisely to one valid subset of A

Claim: Any simple cycle must pass through s.
Remove all edges with one endpoint at s from the graph
By construction, there exists no edge between nodes v_i, v_j where $i > j$
Therefore, there exists a topological ordering of this new graph.
Therefore, there exist no cycles in this new graph without any edges to s.
Therefore, any cycle must pass through s.

Claim: No edge in the graph has a weight $w \neq a_i \wedge w \neq -T$
By construction, every graph is assigned a weight either from $a_i \in A$ or $-T$

Claim: If a simple cycle C has weight 0, there exists a subset with sum T
Remove the edge (v_k, s) from the cycle to get C'. By construction, $\sum_{e_i \in C'} w_i = T$ since the weight of $(v_k, s) = -T$
Construct a subset A' which contains all a_i such that $e_i \in C'$
The sum of this subset is precisely T
Claim: If no simple cycle C has weight 0, there exists no subset with sum T
Suppose there existed a subset $\{x_1, x_2, \ldots, x_n\}$ with sum T.
Order this set such that $x_i < x_j \equiv i < j$
Create a cycle using edges $(s, x_1), (x_1, x_2), \ldots, (x_n, s)$
The first n edges have a total weight of T by construction.
The last edge has a weight of $-T$
Therefore the cycle has zero weight.
Contradiction.

□

Proof of runtime:

Proof. Let the size of the set be N
For each set element, precisely 1 vertex and at most $\theta(N)$ edges are made
Therefore, the runtime of the reduction is $\theta(N^2)$
This is polynomial in N.

□