CS3230 Assignment 10

Zhuang Jianning
A0214561M

1. polynomial time algorithm to decide satisfiability of 3-SAT formulas
use algorithm to find a satisfying assignment for a given 3-SAT formula in polynomial time

let ϕ be a 3-SAT formula
and A be the polynomial time algorithm to decide satisfiability of ϕ

Assume that ϕ has at most n variables $\{x_1, x_2 \dots x_n\}$

3-SAT-Assignment (ϕ, A, n)

Use A to check if ϕ is satisfiable
If NO,
return NULL

For $i = 1$ to n
set x_i = False
Use A to check if ϕ with x_i = False is satisfiable
If YES,
fix x_i = False
Else
fix x_i = True

} there must exist an assignment of x_i that satisfies ϕ since A determined ϕ is satisfiable

return recorded assignment of $\{x_1, x_2 \dots x_n\}$

Since A runs in polynomial time, $O(n)$ calls of A also runs in polynomial time

2. LONGEST PATH (optimization)

Given undirected graph G and 2 vertices u, v
return # edges in a longest simple path between u and v

LONGEST PATH LENGTH (decision)

Given additional nonnegative integer k, decide whether there exist a simple path $\geq k$ length?

Prove there is a polynomial time algorithm for longest path (LP)
iff there is a polynomial time algorithm for longest path length (LPL)

$\Rightarrow$ Suppose there is a polynomial time algorithm for LP

decision problems reduce to optimization problems

$LPL \leq_p LP$

input for LPL used for LP

longest path returned from LP used to check if length $\geq k$ } polynomial time reduction

since there is a polynomial time reduction from LPL to LP and a poly-time algorithm for LP, there is a poly-time algorithm for LPL //

$\Leftarrow$ Suppose there is a polynomial time algorithm for LPL

edges in longest path is at most $|E|$

Given input for LP, construct an instance of LPL with $k = \frac{|E|}{2}$

If yes, length of longest path between $\frac{|E|}{2}$ and $|E|$

Else, less than $\frac{|E|}{2}$

Using binary search, determine length of path in $\lg E$ steps (polynomial in input $G = (V, E)$)

$LP \leq_p LPL$

since there is a polynomial time reduction from LP to LPL and a poly-time algorithm for LPL, there is a poly-time algorithm for LP //

3. PARTITION

Given a list of positive integers $a_1, a_2 \dots a_n$
determine if they can be partitioned into 2 disjoint parts with same sum

a) Let M be the sum of all the integers.

If M is odd, there are no subsets with equal sum $\frac{M}{2}$

Else, calculate $\frac{M}{2}$ and reduce the PARTITION problem to KNAPSACK

Given a list of positive integers $a_1, a_2 \dots a_n$ as input to PARTITION

let $(a_1, a_1), (a_2, a_2) \dots (a_n, a_n)$, $\frac{M}{2}$ be input to KNAPSACK (lectures) $\Rightarrow O(n)$ sum a_i

which outputs a subset $S \subseteq \{1, 2, \dots n\}$ that maximizes $\sum_{i \in S} a_i$ such that $\sum_{i \in S} a_i \leq \frac{M}{2}$ $\Big\} O(nM)$

solution to knapsack can be used to sum a_i for $i \in S$ and decision is made if sum $= \frac{M}{2}$ $\Rightarrow O(n)$

(maximum sum of weights = maximum sum of value = $\frac{M}{2}$) Decision problem

since reduction is in $O(n)$ and KNAPSACK with $W = \frac{M}{2}$ runs in $O(nM)$

PARTITION also runs in $O(nM)$ //

b) algorithm in a) is pseudo polynomial as it runs in time polynomial
to numeric value of input (M = sum of a_i)
but is exponential in length of input (2^n subsets)

pp algorithm for knapsack is pseudo-polynomial //