CS3230 Tutorial 9

Deng Tianle (T15)

24 October 2025

(Polynomial) Reductions

Suppose we know how to solve problem B. Then for another problem A, we can solve it if we reduce it to problem B. This is a very common idea.

For this and the next chapter we are mainly intersted in whether a problem can be solved in polynomial time ('efficiently', this is explained in the lecture [1]). Therefore we need the translation from A to B and then back to A be all polynomial time. In many cases this would be obvious.

If A has a polynomial time reduction to B, we write $A \leqslant_p B$.

[1] I would attach an interesting excerpt related to this in the repo

Reductions for decision problems

Let A and B be two decision problems. In this context, we know how to solve B. A transformation ϕ from instances α of A to instances of B satisfies two conditions:

- α is a YES-instance for $A \iff \phi(\alpha)$ is a YES-instance for B
- The transformation takes polynomial time in the size of α

We say $A \leqslant_p B$.

In other words: YES-instance of $A \mapsto$ YES-instance of B, and NO-instance of $A \mapsto$ NO-instance of B.

OR: $\phi(\alpha)$ YES $\implies \alpha$ YES, and $\phi(\alpha)$ NO $\implies \alpha$ NO.

We need this to ensure that the decision for $\phi(\alpha)$ gives us the decision for α.

Note that there is no requirement for ϕ to be one-to-one or onto.

Q1

Fix a graph G. Graph colouring so that no adjacent vertices share the same colour. Decision problem: given k, whether it is possible to colour with k colours. Optimisation problem: minimum number of colours so that it is possible.

Part $(a) \iff (c)$: True, see if minimum is $\leqslant k$.

Part $(b) \iff (d)$: True, solve sequentially or binary search to see if it is possible with k colours. The search is polynomial time.

Q2

PARTITION: given a set of positive integers S, can we partition it into two subsets of equal total sum?

BALL-PARTITION: given k balls, can we divide them into two boxes with equal number of balls? (Basically, whether k is even, LOL)

Attempted transformation A from PARTITION to BALL-PARTITION: use the sum of all numbers in S as the number k for BALL-PARTITION.

Q2

1. It does run in polynomial time because it only sums the integers in S.
2. Transformation is not correct, see below.
3. Indeed, this is the problem. For example, $S = \{1, 7\}$ maps to $k = 8$ which is a YES instance, but S is a NO instance.
4. This is not the problem. In a YES instance S, the partition shows that the total sum k is even, giving a YES instance $A(S)$.

In this case, a YES decision for $A(S)$ can mean both YES or NO decision for S, so the transformation is not useful.

Q3

PARTITION: given a set of positive integers $\{w_1, \ldots w_n\}$, can we partition it into two subsets of equal total sum?

KNAPSACK: given weight-value pairs $\{(w_1, v_1), \ldots, (w_n, v_n)\}$, capacity W and threshold V (all positive integers), can we choose a subset $I \subset \{1, \ldots, n\}$ such that $\sum_{i \in I} w_i \leqslant W$ and $\sum_{i \in I} v_i \geqslant V$?

Transformation: given PARTITION instance $\{w_1, \ldots, w_n\}$ with $S := \sum_{i=1}^{n} w_i$, construct a KNAPSACK instance $\{(w_1, w_1), \ldots, (w_n, w_n)\}$ with $W = V = S/2$.

Q3

1. True, transformation is polynomial-time as it just copies n weights to n pairs (to be precise $O(n \log(w_{\max})))$.
2. True. This is obvious (just choose one of the two partition subsets).
3. True, since $\{(w_1, w_1), \ldots, (w_n, w_n)\}$ being a YES instance implies that there is a subset $I \subset \{1, \ldots, n\}$ such that $S/2 \leqslant \sum_{i \in I} w_i \leqslant S/2$. Hence this I gives a partition of $\{w_1, \ldots w_n\}$.

Q4

Hamiltonian-cycle (HC): a cycle (meaning, start $=$ end) that visits each vertex exactly once.

Travelling-salesperson (TSP): for a complete graph, whether there is a Hamiltonian cycle with cost $\leqslant n$.

Transformation: let $G = (V, E)$ be a graph (so an instance of HC), complete G into $\bar{G}$ as follows: for every pair $u, v \in V$, let $w(u, v) = 1$ if $(u, v) \in E$, otherwise let $w(u, v) = \infty$ (or anything > 1). Apply TSP to cost $n = |V|$.

The transformation is polynomial-time, to be precise $O(n^2)$ as at most $n(n-1)/2$ edges are added.

Observe that G has a Hamiltonian cycle $\iff$ $\bar{G}$ has a TSP tour of cost at most n. Can you see what this means?

Q4

Observe that G has a Hamiltonian cycle $\iff$ $\bar{G}$ has a TSP tour of cost at most $n = |V|$.

3. ($\Rightarrow$) Since $\bar{G}$ has the same vertices and only adds edges to G, the Hamiltonian cycle is still a Hamiltonian cycle. Cost is n since it consists of n edges (including return to start) in E.
4. ($\Leftarrow$) Let C be a TSP tour of cost at most n in $\bar{G}$. So C has exactly n edges. Then cost $\leqslant n$ implies that all these edges are in E. Hence C is a Hamiltonian cycle for G.

LeetCode

We give a sketch for the solution of TSP using bitmask DP.

Let S be the set (not including vertex 0 meaning the starting vertex) of visited vertices and v be the current vertex. Consider all paths starting at v that visits the remaining $(V \setminus S)$ vertices exactly once and ending at vertex 0. Let $dp[S][v]$ denote the minimum cost of such paths. We get $dp[V][0] = 0$ and

$$dp[S][v] = \min\{dp[S \cup \{u\}][u] + w(v,u) : u \notin S\}.$$

To apply DP, we have to parse the subset S into a number. A natural option is a binary number of length $n = |V|$ (where i-th bit is 1 iff i-th element is present in $S \subset V$). Then we can use memorisation with table size $2^n \times n$. Complexity is $O(n^2 2^n)$ because we are minimising over all $u \notin S$ in each step.