CS 344: Design and Analysis of Computer Algorithms

Instructor: Kangning Wang

Homework 5: Due on **May 3, 2025**

Acknowledgment: Our team members are Alexander Lin (al1655), Pranav Tikkawar (pt422), and Ivan Zheng (iz72) .

Problem 1

There are n children forming a circle. The i-th child currently has a_i apples. Let $\bar{a} = \frac{\sum_{i=1}^{n} a_i}{n}$ be the average number of apples that a child has, and we assume that it is an integer.

In one move, we can take one apple from a child and give it to an *adjacent* child. What is the minimum number of moves we have to make to make every child have exactly $\bar{a}$ apples? Model this problem by a flow network with costs, and therefore show that this problem can be solved in polynomial time.

Solution. Modeling a flow network, each child is a node in the network with nodes 1, 2,...,n. Each node i has a supply (if $a_i > \bar{a}$) or demand (if $a_i < \bar{a}$). Supply $s_i = a_i - \bar{a}$:

- If $s_i > 0$: node i has s_i apples to give away.
- If $s_i < 0$: node i needs $|s_i|$ apples.

For each pair of adjacent children (i and (i mod n) + 1), created two directed edges:

1. From i to (i mod n) + 1
2. From (i mod n) + 1 to i

Each edge has:

- Capacity: infinite (or at least large enough for all apples)
- Cost: 1 per apple (because one move = moving one apple to an adjacent node)

Sending flow along an edge represents moving apples from one child to an adjacent child. The cost of the flow is the total number of moves. This is a minimum cost flow problem, which can be solved in polynomial time using algorithms like the Successive Shortest Path algorithm or the Cycle-Canceling algorithm. The minimum cost flow will give us the minimum number of moves needed to balance the apples among the children.

Problem 2

Explicitly construct a bijection between each of the following pairs of sets and therefore show that they have the same cardinality.

1. Natural numbers and perfect squares.
2. $(0, 1)$ and $(1, +\infty)$.
3. $(-1, 1)$ and $\mathbb{R}$.
4. $[0, 1]$ and $[0, 1)$.

Solution.

1. We map every natural number n to its square n^2. This is injective, since each natural number has a unique square, and it is surjective, as for each perfect square $p = q^2$, we can take its square root and get the natural number $|q|$ which maps to it. Therefore this mapping is bijective, and the two sets have the same cardinality.
2. For every number n in the set $(1, +\infty)$, we map it to its multiplicative inverse $\frac{1}{n}$. This is injective, as every n maps to a unique number, and surjective, as for every number $p \in (0, 1)$, we can find its corresponding number $n \in (1, +\infty)$ by doing $n = \frac{1}{p}$. Thus the mapping is bijective.
3. Similarly to the last problem, for every number p in the subset $(-1, 1) \setminus 0$ of the first set $(-1, 1)$, we map it to its multiplicative inverse $\frac{1}{p}$ in the subset $\mathbb{R} \setminus 0$ of the second set $\mathbb{R}$. We additionally map 0 in the first set to 0 in the second set. This is injective, as each n maps to its unique multiplicative inverse, except 0, which does not have a multiplicative inverse. However, we have mapped 0 in the first set to 0 in the second set, and since no number has multiplicative inverse 0, 0 uniquely maps to 0, satisfying injectiveness. This is surjective, as for every $q \in (R)$ we can find its corresponding element in the first set by performing $\frac{1}{q}$. Again, the exception is 0, which we know maps to 0. Therefore we have a bijection.
4. Map every number from the first set to the same number in the second set, except for numbers that can be expressed in the form $\frac{1}{n}$, where n is an integer greater than 0, which we instead map to $\frac{1}{n+1}$. For example, $1 = \frac{1}{1}$ in the first set maps to $\frac{1}{2}$ in the second, $\frac{1}{2}$ in the first set maps to $\frac{1}{3}$ in the second, and so on. This means no number in the first set maps to 1 in the second set. This is injective, as each number a in the first set either cannot be written in the form $\frac{1}{n}$ for some natural number n, in which case it maps to itself uniquely, or it can be written in that form, in which case it maps uninquely to $\frac{1}{n+1}$. This is injective, as for a number b in the second set that does not follow the form $\frac{1}{n}$, it will be mapped to by themselves, and for b that does follow the form, we can find the number that maps to it as $\frac{1}{n-1}$. Therefore the mapping is bijective.

Problem 3

Review the halting problem, and prove that the following problem is also undecidable: Given a Turing machine, decide whether it always correctly decides the set cover problem. (Note that the given Turing machine may not halt on a valid input, in which case it does not correctly decide the set cover problem.)

Solution.
We reduce from the halting problem, which is known to be undecidable.
Assume for contradiction: Suppose we have a decider D that takes as input a Turing machine M and determines whether M always correctly decides the set cover problem.We will construct a Turing machine M' based on any input M and w, such that:M' behaves like this:
On input x:
If x is a valid instance of the set cover problem, then: Simulate $M(w)$.
If $M(w)$ halts, solve the set cover problem correctly on input x.
If $M(w)$ does not halt, loop forever.
Thus, M' behaves as follows:
If $M(w)$ halts, then M' always halts and gives the correct answer on all inputs to the set cover problem.
If $M(w)$ does not halt, then M' loops forever on all set cover inputs, hence does not "correctly decide" the problem.
Use the assumed decider D on M':
If $D(M')$ says "yes", then $M(w)$ halts.
If $D(M')$ says "no", then $M(w)$ does not halt.
Therefore, using D, we could decide whether $M(w)$ halts — which contradicts the undecidability of the halting problem.‘

Problem 4

Prove that the following problem is NP-complete: Given n positive integers that sum to $2W$, decide whether we can partition the integers into two parts with equal sum (W each).

Solution.
To prove a problem is NP-complete, we first need to show that it is in NP. We can easily check if a partition is a solution by adding all integers in each partition together and checking if the sum for the first partition equals the sum for the second partition. This certifier is clearly polynomial time (it would be $O(n)$), and thus our problem is in NP.

We now have to show that our problem is NP-hard. We can do this by reducing a problem we know is NP-complete (and thus NP-hard) to this problem. Here, we will reduce the subset sum Problem, which we know is NP-complete, to this problem.

The subset sum problem states that we have a set of positive integers $A = \{a_1, a_2, ...a_n\}$ and we must decide whether there is a subset that sums to a number S. We must now find a polynomial-time algorithm to convert an instance of the subset sum problem into this problem.

Take an instance of the subset sum problem where we have a set A and target sum S, and the sum of the integers in the set A is A_{tot}. We construct a new set $A' = \{a_1, ...a_n, 2S - A_{tot}\}$, with sum $A'_{tot} = A_{tot} + 2S - A_{tot} = 2S$.

Now if there is a subset $B \subseteq A$ which sums to S, then the subset $A' \setminus B$ would have the sum $2S - S = S$, and we therefore have two subsets of equal sum, which is our problem. Therefore any instance of the subset sum problem has an equivalent instance in our problem.

Similarly, for an instance of our problem, if we can partition A' into two subsets with equal sum S, then one of the subsets has to have the element $2S - A_{tot}$, which means the rest of the subset sums to $S - 2S + A_{tot} = A_{tot} - S$. Then since all the elements of A' are in A except this $2S - Atot$ element, which is already in the first subset, and the set A summed to A_{tot}, the other subset must sum to $A_{tot} - (A_{tot} - S) = S$. Therefore our instance is equivalent to an instance of the subset sum problem.

Our reduction is clearly polynomial time because it is simply adding another element. We have therefore shown that this problem is in NP and can be reduced to from an NP-hard problem in polynomial time, so it is NP-complete.

Problem 5

Prove that the following problem is `NP`-complete: Given a weighted directed graph with n vertices and m edges, decide whether the graph has a simple cycle with sum of weights equal to 344. The weight of each edge must be an integer, but can be positive, negative, or zero.

Solution. To prove a problem is NP-complete, we first need to show the problem is in NP. To do this, we can check that given a cycle as a certificate, we can check if it's simple and that the sum of edge weights along the cycle is exactly 344. Both of these can be done in polynomial time, thus the problem is in NP.

Next, to show that the problem is NP-hard, we need to reduce a known NP-complete problem to it. The Subset Sum problem says that:

- given integers $a_1, a_2, ..., a_n$ and integer K does there exist a subset whose sum is exactly K?

We can reduce the Subset Sum problem (which is known NP-complete) to our cycle problem. We start by building the graph:

1. Create n + 1 vertices $v_0, v_1, v_2, ..., v_n$.

2. For each i = 1, ...,n:
    - Add a directed edge from v_{i-1} to v_i with weight 0.
    - Add a directed edge from v_{i-1} to v_i with weight a_i.
3. Add a directed edge from v_n to v_0 with weight 0.

Next is the cycle correspondence:

1. Any simple cycle starting at v_0, going through to v_n, and back to v_0 corresponds to a selection of edges.
2. At each step i, you choose either:
    - The 0-weight edge (meaning don't include a_i), or
    - The a_i-weight edge (meaning include a_i in the sum).

Thus, the total weight of the cycle is the sum of the selected a_i values.
The next step is to set the target weight:

1. Set the target weight to K.
2. The cycle has weight K $\iff$ there is a subset of the a_i summing to K.

Finally, we map to 344:

- Since the cycle problem has a fixed target of 344, we can transform any instance of Subset Sum to an instance of our problem by shifting the target appropriately (e.g., for any Subset Sum instance, ask whether there is a cycle of weight $K' = 344$ by adjusting edge weights by $(344 - K)$ if needed).

Therefore, the problem is NP-complete.