Homework 1 (Due Date: Friday 15 September)

Please write the following on your homework:

- Name
- Collaborators (write none if no collaborators)
- Source, if you obtained the solution through research, e.g. through the web.

While you may collaborate, you *must write up the solution yourself* and you are asked to try doing the questions yourself first before discussing with others. There will be a box outside COM2-02-06. Please submit your answers on a separate piece of paper into the box by the deadline. You do NOT need to submit the question paper.

If you are asked to describe an algorithm, you should first provide a short description of the problem and what your main results are. Then you should provide a description of the algorithm in English, and if helpful, in pseudo-code. An example, or diagram can often be helpful in helping describe the algorithm. You should provide a proof (or indication) of correctness of the algorithm, and an analysis of the running time of the algorithm. Remember that you need to communicate clearly. Grades cannot be awarded if the grader is unable to understand what you are writing.

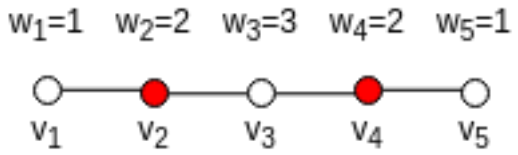
---

1. **Minimum Weighted Vertex Cover**
   Given an undirected, unweighted graph $G = (V, E)$, a vertex cover of G is a subset S of V such that for every edge $e = (u, v)$, either $u \in S$, $v \in S$ or both. Suppose also that each vertex v in G has a weight w_v. One interesting problem that arises is to find the vertex cover S such that the total weight of the vertices in S is minimised. This is called the Minimum Weighted Vertex Cover (MWVC) problem.

   This problem arises in many places such as in communication networks with nodes and links between the nodes. Someone might want to find a set of nodes that can listen to communication between every link in the network and this set of nodes must be a vertex cover of the network. Listening to each node might require different costs due to factors such as distance so minimising the total cost is important.

   As you will learn later in the course, this problem is NP-hard which means there is currently no known algorithm that solves this problem in polynomial time on general graphs. However, there are some special classes of graphs where this problem is solvable in polynomial time and in this question we will discuss one of them.

First, we will look at a special class of graphs called paths. If G has n vertices and we label the vertices $v_1, v_2 ... v_n$ then there is an edge between vertex v_i and v_j if and only if $|i-j| = 1$. One example for $n = 5$ is the shown below.



The vertices in red shows the minimum weighted vertex cover with a total weight of 4. You can verify that there is no other vertex cover with a smaller total weight. For example, $S = \{v_1, v_4\}$ is not a valid vertex cover because the edge (v_2, v_3) is not covered and $S = \{v_1, v_3, v_5\}$ is a valid vertex cover, but its total weight is 5 which is not the minimum.

(a) Consider the following algorithm that returns minimum total weight required to cover a path of n vertices and $W[1...n]$ is an array of integers such that $W[i]$ is the weight of the i-th vertex.

GREEDYMWVC($W[1...n]$)

1 $even\_total \leftarrow$ sum of weights of even-numbered vertices
2 $odd\_total \leftarrow$ sum of weights of odd-numbered vertices
3 **return** $\min(odd\_total, even\_total)$

Is this algorithm correct? If it is, prove its correctness. Otherwise, provide a counter-example of when this algorithm will give the wrong answer.

(b) We now want to design a divide and conquer algorithm to solve this problem. Let's focus on the edge $(v_{\lfloor \frac{n}{2} \rfloor}, v_{\lfloor \frac{n}{2} \rfloor + 1})$ in the middle. To cover this edge, either $v_{\lfloor \frac{n}{2} \rfloor}$ or $v_{\lfloor \frac{n}{2} \rfloor + 1}$ must be in the vertex cover. Based on that, answer the following questions by using MWVC(i, j) to denote the minimum total weight required to cover the edges between vertices $v_i, v_{i+1}, ..., v_j$.

i. If $v_{\lfloor \frac{n}{2} \rfloor}$ is taken, what is the minimum total weight required to cover the remaining edges?

ii. If $v_{\lfloor \frac{n}{2} \rfloor + 1}$ is taken, what is the minimum total weight required to cover the remaining edges?

Using the previous 2 formulae, complete the following divide and conquer algorithm to solve the MWVC problem and analyse its running time.

SLOWMWVC($W[1...n]$) returns MWVC$(1, n)$

1 **if** $n == 1$
2 **return** _____
3 $ans1 \leftarrow$ _____
4 $ans2 \leftarrow$ _____
5 **return** _____

(c) While SLOWMWVC is simple, it is quite slow because it solves several overlapping subproblems multiple times. We can improve on SLOWMWVC to significantly improve the running time by combining the overlapping subproblems. Complete the algorithm below using ideas similar to SLOWMWVC and analyse its running time. For simplicity, you may assume that $n \geq 3$.

FASTMWVC($W[1...n]$)
returns (MWVC($1, n$),MWVC($1, n-1$),MWVC($2, n$),MWVC($2, n-1$))

```
1  if n == 3
2      return (___, ___, ___, ___)
3  (s_1, s_2, s_3, s_4) ← FASTMWVC(W[1...⌊n/2⌋])
4  (t_1, t_2, t_3, t_4) ← FASTMWVC(W[(⌊n/2⌋ + 1)...n)
5  return (_______, _______, _______, _______)
```

2. **In-place Sorting**
An in-place algorithm is an algorithm that transforms an input into the required output without using any extra data structures. It usually updates the input sequence by replacing individual elements or swapping elements within the input. However, a small amount of extra storage space is allowed for variables.

For this problem, we will ignore the stack memory required for recursive function calls. Thus for example, bubble sort and Quicksort are both considered in-place algorithms as they only involve swapping elements within the array without declaring a new array to store any data.

You are given an array $A[1...n]$ where each element is an integer between 1 and k inclusive and $\log n < k < n$. However, you do not know what is the value of k.

Design an in-place sorting algorithm that sorts A in worst case $O(n \log k)$ time.

Hint: Find a way to get the value of k first