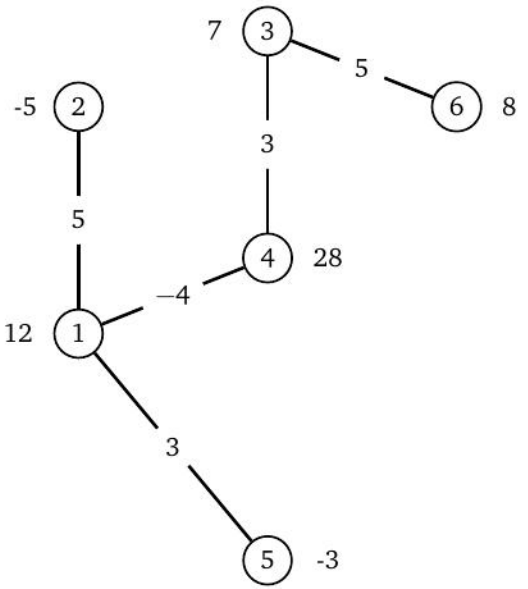
Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a **weighted, undirected, acyclic, connected** graph with weight function $w : \mathcal{V} \longrightarrow \mathbb{Z}$. Let n be the number of vertices in $\mathcal{G}$. Observe that such a graph has exactly $n - 1$ edges. We are also given a cost $c : \mathcal{E} \longrightarrow \mathbb{Z}$ per edge. Since $\mathcal{G}$ is connected and acyclic, if we remove any edge $e \in \mathcal{E}$ from $\mathcal{G}$, we obtain two connected components, say $\mathcal{H}_1^e, \mathcal{H}_2^e$. We define the **vulnerability (γ_e)** of any edge $e \in \mathcal{E}$ as follows.

$$\gamma_e = c(e) - \left| \left(\sum_{x \text{ a vertex of } \mathcal{H}_1^e} w(x) \right) - \left(\sum_{y \text{ a vertex of } \mathcal{H}_2^e} w(y) \right) \right|$$



Edge	Vulnerability
$\{1, 2\}$	-52
$\{1, 4\}$	-43
$\{1, 5\}$	-50
$\{3, 4\}$	-14
$\{3, 6\}$	-26

Figure 1: A graph and the vulnerability of its edges.

Vertices are labeled with $\{1, 2, \ldots, n\}$. Figure 1 exhibits an example of a graph and the vulnerabilities of its edges. In this assignment, you **write a C[redacted] function to find an edge having the highest**

vulnerability. If there are more than one edge having the highest vulnerability, output any one of them. Represent the graph with an adjacency list.

Part I: Brute-force $\mathcal{O}(n^2)$-Time Algorithm

In this part, you implement the following algorithm. For each edge $e \in \mathcal{E}$, you remove the edge from the graph, obtain the two connected components $\mathcal{H}_1^e$ and $\mathcal{H}_2^e$, compute the sum of the weights of the vertices in each $\mathcal{H}_1^e$ and $\mathcal{H}_2^e$, and compute $\gamma(e)$. Finally output an edge e with the smallest $\gamma(e)$.

Part II: Linear Time Algorithm

Design an $\mathcal{O}(n)$-time algorithm for this problem. *Hint: use DFS (you are welcome to use any other approach)!*

main()

1. Read n from the user.
2. Read the weights of these n vertices.
3. Read the edges along with its weight.
4. Dynamically allocate space to store the graph in the adjacency list format using malloc/calloc/new
5. Compute an edge with the smallest vulnerability using both the methods and output.

You assume that the given graph is weighted, undirected, acyclic, connected (no need to check this in your code). **An edge is denoted by an unordered pair of vertices whose order does not matter. For example, when giving input, user can denote an edge as 2 3 or 3 2. You code should behave identically in both the cases.** Submit a single .c [redacted] file. Your code should get compiled properly by gcc [redacted] compiler.

Sample Output

```
Write n: 6
Write weights of vertices: 12 -5 7 28 -3 8
Write the edges and their weights
1 2 5
1 4 -4
1 5 3
3 4 3
3 6 5
Edge with the smallest vulnerability computed using first method: 3 4
Edge with the smallest vulnerability computed using second method: 3 4
```