Algorithm explanation:

In this case, a dynamic programming algorithm will be used to find the minimum spin. For that you must get the recursive equation that solves the problem.

The recursive function is the following:

$$\phi(u) = \begin{cases} S_u * H_u \ \ if\ child(u) = \emptyset \\ \downarrow\ (S_u * H_u + \sum_{v\ \epsilon child(u)} S_v * S_u * J_{u,v} + \phi(v)\ for\ S_u = -1, \\ S_u * H_u + \sum_{v\ \epsilon child(u)} S_v * S_u * J_{u,v} + \phi(v)\ for\ S_u = 1) \end{cases}$$

to build the tree from the graph we use the depth-first search algorithm, launched from any node, since the result is the same regardless of the root node.

the DFS algorithm runs in O(V+E) time, being a tree, it is known that E=2*(V-1) (Because is an undirected graph) therefore the DFS algorithm runs in O(V) =O(n).

To represent the weight of the arcs given the restriction, a dictionary was used for the sparse matrix, accessing its elements is achieved in constant time.

using only recursion, it is easy to see that the complexity is of exponential order O(2^n), since in each call two recursions are made, creating a binary tree of height n, the number of recursive calls is 2^n and there apply constant operations, therefore it is O(2^n).

For the DP algorithm we only store one element which is the node, therefore the temporal complexity is O(n) and the spatial complexity the dictionary stores O(n) elements, the tree is stored in O(n) space, the weights t of the solution S is also stored in time O(n), finally memorization stores spin value answer vector S therefore wastes O(n), therefore the time complexity and the spatial complexity is O(n) which belongs to O(n log n).

Thanks to dynamic programming, we can avoid repeating calculations, since given the node where you go, the result of the least spin will always be the same no matter when it is called, thanks to this we can reduce the calculations of O(2^n) to O(n)

To solve the DP problem, the memorization technique is used.

Finally, 9 test cases were achieved by creating random trees using the networkx library that allows generating random graphs and trees to obtain these.

Therefore, it is possible to solve this particular case of the general problem. It is impossible to do it in a general graph, since there are cycles in a graph, that implies the recursive call would remain in an infinite recursion. For that, it is necessary to look for better approximations, any way to solve is at least O(2^n) that is, it is of exponential order, but the solution can be verified in polynomial time, therefore the algorithm is class NP, but it is not safe let it be class P