6007. Maximum AND Sum of Array

My Submissions (/contest/weekly-contest-280/problems/maximum-and-sum-of-array/submissions/) Back to Contest (/contest/weekly-contest-280/)

You are given an integer array `nums` of length `n` and an integer `numSlots` such that `2 * numSlots >= n`. There are `numSlots` slots numbered from `1` to `numSlots`.

You have to place all `n` integers into the slots such that each slot contains at **most** two numbers. The **AND sum** of a given placement is the sum of the **bitwise** `AND` of every number with its respective slot number.

- For example, the **AND sum** of placing the numbers `[1, 3]` into slot `1` and `[4, 6]` into slot `2` is equal to `(1 AND 1) + (3 AND 1) + (4 AND 2) + (6 AND 2) = 1 + 1 + 0 + 2 = 4`.

Return *the maximum possible* ***AND sum*** *of* `nums` *given* `numSlots` *slots.*

User Accepted:	0
User Tried:	0
Total Accepted:	0
Total Submissions:	0
Difficulty:	Hard

Example 1:

```
Input: nums = [1,2,3,4,5,6], numSlots = 3
Output: 9
Explanation: One possible placement is [1, 4] into slot 1, [2, 6] into slot 2, and [3, 5] into slot 3.
This gives the maximum AND sum of (1 AND 1) + (4 AND 1) + (2 AND 2) + (6 AND 2) + (3 AND 3) + (5 AND 3) = 1 + 0 + 2 + 2 + 3
```

Example 2:

```
Input: nums = [1,3,10,4,7,1], numSlots = 9
Output: 24
Explanation: One possible placement is [1, 1] into slot 1, [3] into slot 3, [4] into slot 4, [7] into slot 7, and [10] into
This gives the maximum AND sum of (1 AND 1) + (1 AND 1) + (3 AND 3) + (4 AND 4) + (7 AND 7) + (10 AND 9) = 1 + 1 + 3 + 4 + 7
Note that slots 2, 5, 6, and 8 are empty which is permitted.
```

Constraints:

- `n == nums.length`
- `1 <= numSlots <= 9`
- `1 <= n <= 2 * numSlots`
- `1 <= nums[i] <= 15`

JavaScript

```
function edge(to, cap, cost, from) {
    this.from = from;
    this.to = to;
    this.cost = cost;
    this.cap = cap;
}

function MCMF(n) {
    const initializeGraph = (n) => { let g = []; for (let i = 0; i < n; i++) { g.push([]); } return g; };
    let g = initializeGraph(n), h = Array(n).fill(0), dis = Array(n).fill(0), prev_v = Array(n).fill(0), prev_e = Array(n).fill(0);
    return { addEdge, minCostFlow }
    function addEdge(from, to, cap, cost) {
        g[from].push(new edge(to, cap, cost, g[to].length));
        g[to].push(new edge(from, 0, -cost, g[from].length - 1));
    }
    function minCostFlow(from, to, flow) {
        let res = 0;
        while (flow > 0) {
            let pq = new MinPriorityQueue({
                compare: (x, y) => {
                    if (x[0] != y[0]) return x[0] - y[0];
```

```
                    return x[1] - y[1];
                }
            });
            dis.fill(Number.MAX_SAFE_INTEGER);
            dis[from] = 0;
            pq.enqueue([0, from]);
            while (pq.size()) {
                let [curDis, cur] = pq.dequeue();
                if (dis[cur] < curDis) continue;
                for (let i = 0; i < g[cur].length; i++) {
                    let child = g[cur][i];
                    if (child.cap > 0 && dis[child.to] > dis[cur] + child.cost + h[cur] - h[child.to]) {
                        dis[child.to] = dis[cur] + child.cost + h[cur] - h[child.to];
                        prev_v[child.to] = cur;
                        prev_e[child.to] = i;
                        pq.enqueue([dis[child.to], child.to]);
                    }
                }
            }
            if (dis[to] == Number.MAX_SAFE_INTEGER) return -1;
            for (let i = 0; i < n; i++) h[i] += dis[i];
            let d = flow;
            for (let i = to; i != from; i = prev_v[i]) {
                d = Math.min(d, g[prev_v[i]][prev_e[i]].cap);
            }
            flow -= d;
            res += d * h[to];
            for (let i = to; i != from; i = prev_v[i]) {
                let edge = g[prev_v[i]][prev_e[i]];
                edge.cap -= d;
                g[i][edge.from].cap += d;
            }
        }
        return res;
    }
}

const maximumANDSum = (a, m) => {
    let n = a.length, mcmf = new MCMF(n + m + 2), from = n + m, to = from + 1;
    for (let i = 0; i < n; i++) {
        mcmf.addEdge(from, i, 1, 0);
        for (let j = 0; j < m; j++) {
            mcmf.addEdge(i, j + n, 1, -(a[i] & (j + 1)));
        }
    }
    for (let i = 0; i < m; i++) mcmf.addEdge(i + n, to, 2, 0)
    return -mcmf.minCostFlow(from, to, n);
};
```

Custom Testcase | Use Example Testcases

Run | Submit

Submission Result: Accepted (/submissions/detail/640378421/) | More Details > (/submissions/detail/640378421/)

Share your acceptance!