「杂题记录」出题人

Jiayi Su (ShuYuMo)

2021-01-19 15:35:06

给出长度为 n 的序列 a_i ，需要构造一个序列 b_i ，使得 $\forall i, \exists x, y \in [1, n], s.t. b_x + b_y = a_i$ 要求输出序列 b_i 和方案（a_i 由哪两个 b_j 构成）。无解输出 -1. d3A

$n \leq 30$

分析

一道搜索模板题。

考虑如果能够构造出一个 a_i 的一个子序列 a' 的对应长度的序列 b' 那么每次增加一个元素 a_k 都可以直接构造 b_k。

如果 a_i 存在偶数，可以直接构造 $b_i = \frac{a_i}{2}$ 。其他的元素直接构造即可。

如果 a_i 全部为奇数，考虑将 b_i 中的元素看成点，a_i 看成边，这样就有 n 个点 n 条边，一定存在一个环，只要找到这个环，剩下的元素就可以直接构造了

设：环上一个元素 b_k，那么对于任意一个环外的元素 i ，可以直接构造成 $b_i = a_i - b_k$。

不妨设 环的大小为 k，$a_i = b_i + b_{i+1}$，$a_k = b_k + b_1$ 。则：$\sum_{i=1}^{k} a_i = 2\sum_{i=1}^{k} b_i$，易知：$2|k$。

可以直接拆成 $a_1 + a_3 + \cdots + a_{k-1} = \sum_{i=1}^{k} b_i = a_2 + a_4 + \cdots a_k$

传 统 艺 能。直接搜索每个点 a_i 是属于上式中的左边、右边 或者 不属于环上的点。要求满足左边的点权和等于右边的点权和。复杂度 $\mathcal{O}(3^n)$.

考虑把 a_i 拆成两段，分别搜索，记录最终方案，然后线性合并。即：折半搜索。

复杂度 $\mathcal{O}(3^{n/2})$ 。

赛后因为太菜不会线性合并，带了一个 log 卡成了 30 ~~体验很好~~。后来写 hash 挂链优化掉了一个 log ，内存占用 2 GiB，然后人就没了。至今任然 30。

```cpp
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
#include <cmath>
#include <climits>
#include <map>
#include <vector>
#define GG() return (void)puts("No")
using namespace std;
#define LL long long
#define int long long
const int _ = 1e2 + 5;
```

```
int n, A[_];

namespace subtask0{void work(){
    int target = 0;
    for(int i = 1; i <= n; i++) if(A[i] % 2) ; else { target = i; break; }
    cout << ("Yes") << endl;
    for(int i = 1; i <= n; i++){
        if(i == target) {
            cout << (A[i] / 2) << " ";
        } else {
            cout << (A[i] - (A[target] / 2)) << " ";
        }
    } cout << endl;
    for(int i = 1; i <= n; i++){
        if(i == target) {
            cout << target << " " << target << endl;
        } else {
            cout << target << " " << i << endl;
        }
    }
}}

namespace subtask1{


struct status_t{
    LL S;
    int A0; // the sum of.
    int A1; // the sum of.
    int sum0; // the number of.
    int sum1; // the number of.
    status_t() { sum0 = sum1 = S = A0 = A1 = 0; }
} ;

vector<status_t> reta, retb;

void d( int now, status_t st, const int & ed, vector<status_t> & ret){
    if(now > ed) {
        ret.push_back(st);
        return ;
    }
    status_t ext;
    ext = st; ext.S *= 3; ext.S += 0; ext.A0 += A[now]; ext.sum0 ++;
    d(now + 1, ext, ed, ret);

    ext = st; ext.S *= 3; ext.S += 1; ext.A1 += A[now]; ext.sum1 ++;
    d(now + 1, ext, ed, ret);
```

```
    ext = st; ext.S *= 3; ext.S += 2;
    d(now + 1, ext, ed, ret);
}



map<pair<int, int>, status_t> M;
#define none (-20031006)
int mid;

void output(status_t A, status_t B){
    vector<int> Node0, Node1, nodes;
    for(int i = 0, id = mid, tmp = A.S;       i < mid; i++, id--, tmp /= 3) { if(tmp % 3 == 0) Node0.pu
    for(int i = 0, id = n  , tmp = B.S; i < (n - mid); i++, id--, tmp /= 3) { if(tmp % 3 == 0) Node0.pu
    for(int i = 0; i < (int)Node0.size(); i++) nodes.push_back(Node0[i]), nodes.push_back(Node1[i]);
    static int Ans[_];  for(int i = 1; i <= n; i++) Ans[i] = none;
    static pair<int, int> Out[_];
    Ans[nodes[0]] = 0;
    for(int i = 0; i < (int)nodes.size(); i++){
        Ans[nodes[(i + 1) % nodes.size()]] = ::A[nodes[i]] - Ans[nodes[i]];
        Out[nodes[i]] = make_pair(nodes[i], nodes[(i + 1) % nodes.size()]);
    }
    static vector<int> ext;
    for(int i = 1; i <= n; i++){ if(Ans[i] != none) continue;
        Ans[i] = ::A[i];
        Out[i] = make_pair(nodes[0], i);
    }
    cout << ("Yes") << endl;
    for(int i = 1; i <= n; i++) cout << Ans[i] << " "; cout << endl;
    for(int i = 1; i <= n; i++) cout << Out[i].first << " " << Out[i].second << endl;
}



void work(){
    mid = (n >> 1);
    reta.clear(); retb.clear();
    d(1, status_t(), mid, reta);
    d(mid + 1, status_t(), n, retb);
    if(reta.size() == 0 || retb.size() == 0) GG();
    for(int i = 0; i < (int)reta.size(); i++) { pair<int, int> Key; Key = make_pair(reta[i].sum0 - reta
    for(int i = 0; i < (int)retb.size(); i++) {
        pair<int, int> Key; Key = make_pair(retb[i].sum1 - retb[i].sum0, retb[i].A1 - retb[i].A0);
        if(M.count(Key)) ; else continue;
        status_t A = M[Key], B = retb[i];
        if(A.sum0 + A.sum1 == 0 || B.sum0 + B.sum1 == 0) continue;
        if((A.sum0 + B.sum0 + A.sum1 + B.sum1) % 2) ;
```

```
        else return output(A, B);
    }
    GG();
} }
signed main(){
    // freopen(".in", "r", stdin);
    ios::sync_with_stdio(false);
    cin >> n; for(int i = 1; i <= n; i++) cin >> A[i];
    int pass = false;
    for(int i = 1; i <= n; i++) if(A[i] % 2) ; else { pass = true; break; }
    if(pass) subtask0::work();
    else subtask1::work();
    return 0;
}
```