R867_Div3

ZeitHaum

2023 年 4 月 27 日

目录

1 D Super-Permutation

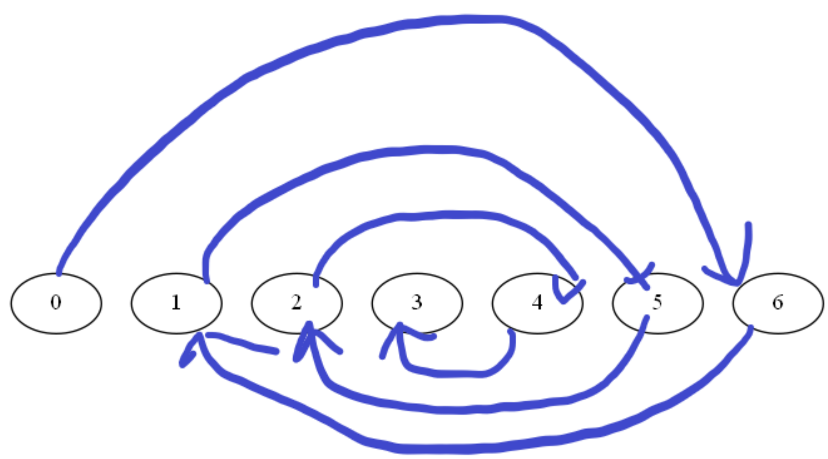
对于输入的 n, 构造一个排列 $a_1, a_2, ..., a_n$, 计算其前缀和 (模 n 意义下), 使得其前缀和序列两两不重复。

1.1 排列的图论模型

构造题。思考一个 $1-n$ 排列的图论模型 (这里取 $n=6$),

构造结点 0 - 6, 其中 0 表式起点，我们每次从 0 开始，不重不漏地将每个结点遍历恰好一次，那么我们的遍历序列 (除去 0 刚好是一个 $1-n$ 的排列).

如下图



此图遍历序列即为 $6->1->5->2->4->3$, 考虑每步移动的距离，可得序列 $6->(-5)->4->(-3)->2->(-1)$.

在模 6 意义下即为 $0->1->4->3->2->5$.

显然如果令 $a=(6,1,4,3,2,5)$, 如此得到的 b 等价于每次遍历的结点编号序列，即 $(6,1,5,2,4,3)$.

因此我们只需构造不重的序列 a, 使得按照 a 的方式遍历刚好可以遍历每个结点恰好一次即可。

1.2 Solution

当 n 为奇数时，$b_n = \frac{n(n+1)}{2} \% n = 0$.

因此 $b_n = 1$, 考虑 $a_i = n(i \neq 1)$ 的情况，此时必有 $b_i = b_{i-1}$。

所以 $b_1 = 1$.

因此只要 $n \neq 1$,b 必有重复元素。

当 n 为偶数时，有两种方法构造:

1.2.1 方法一: 观察样例法

仔细观察样例 (或者自己多构造几组样例), 发现其形如 $0, -1, 2, -3, ..., (-1)^{n-1}(n-1)$.

其前缀和为 $0, -1, 1, -2, 2, ..., -\frac{n}{2}$.

满足条件。

方法二: 图论模型法

另外一种方法便是观察图论模型，可以发现 $0, -(n-1), n-2, -(n-3), ..., -1$ 也是一组解 (见1.1)。

其等价于 $0, 1, -2, 3, ..., (-1)^n(n-1)$.

本质上是方法一的对称情况，证明略。

1.3 Code

1.3.1 方法一

```cpp
#include<bits/stdc++.h>
using namespace std;

void solve(){
    int n;
    cin>>n;
    if(n%2==1 && n>1){
        cout<<-1<<"\n";
    }
    else{
        vector<int> ans(n+1);
        for(int i = 1;i<=n;i++){
            ans[i] = i-1;
```

```
        }
        ans[1] = n;
        for(int i = 2;i<=n;i+=2){
            ans[i] = n-ans[i];//-1^(i-1)*(i-1)
        }
        for(int i  = 1;i<=n;i++){
            cout<<ans[i]<<(i!=n?" ":"\n");
        }
    }
}

signed main(){
    //fastio. IO's constant is very large(5+).
    ios::sync_with_stdio(false);
    cin.tie(0);
    int t;
    cin>>t;
    for(int i = 0;i<t;i++){
        solve();
    }
}
```

1.3.2 方法二

```
#include<bits/stdc++.h>
using namespace std;

void solve(){
    int n;
    cin>>n;
    if(n%2==1 && n>1){
        cout<<-1<<"\n";
    }
    else{
        vector<int> ans(n+1);
        for(int i = 1;i<=n;i++){
            ans[i] = i-1;
        }
```

```
        ans[1] = n;
        for(int i = 3;i<=n;i+=2){
            ans[i] = n-ans[i];//-1^(i)*(i-1)
        }
        for(int i  = 1;i<=n;i++){
            cout<<ans[i]<<(i!=n?" ":"\n");
        }
    }
}

signed main(){
    //fastio. IO's constant is very large(5+).
    ios::sync_with_stdio(false);
    cin.tie(0);
    int t;
    cin>>t;
    for(int i = 0;i<t;i++){
        solve();
    }
}
```

1.4 E. Making Anti-Palindromes

给定一个字符串 s，定义反回文串：如果一个字符串对于任意 i，都有 $s[i] \neq s[n-i] (i \in \{1, 2, ..., n\})$，那么就称字符串 s 为反回文串。现在你可以进行一个交换操作，也即交换字符串两个位置上的字符，求最少的交换次数使得 s 变为反回文串。

1.5 Solution

定义集合 $S = \{(i, n-i) | s[i] = s[n-i] 且 i < n-i\}$，我们的目的便是将 S 清空。

可以将消除分为以下两个步骤:

步骤 1: 观察到一个交换操作最多可以消除 2 个 S 中的元素 $(i, n-i), (i', n-i')$，只需满足 $s[i] \neq s[i']$ 即可。

反复重复上述操作，最终 S 下标对应的字符将会全部相同，记此时的集合为 S'.

步骤 2: 对于这些字符全相同的字符，1 次交换只能消除 1 个 S 中的元素。具体操作如下:

对于 $(i, n-i)$, 选择一对索引 $(j, n-j)$, 其中 $s[j] \neq s[i], s[n-j] \neq s[i]$ 且 $s[j] \neq s[n-j]$.

交换 (i, j) 即可。

现在考虑最小化操作数量，显然**步骤 2** 需要的操作数为 $|S'|$, 因此我们需要在步骤 1 中最小化 $|S'|$.

记 $\text{count}(i)$ 为 S 中满足 $s[i] = s[i']$ 的索引数量,$(i, n-i), (i', n-i') \in S$。

考虑 S 中任意使得 count 函数最大的 i, 记为 i_{max}.

显然可以对于 $\text{count}(i_{max})$ 进行分类讨论，

如果 $\text{count}(i_{max}) \geq \sum_{\text{count}(i') \neq \text{count}(i)} \text{count}(i')$,

观察到此时最佳方案为每个交换 i 和 i', 其中 $\text{count}(i) = \text{count}(i_max), \text{count}(i') \neq \text{count}(i_{max})$.

此时 $\min(|S'|) = \text{count}(i_{max}) - \sum_{\text{count}(i') \neq \text{count}(i)} \text{count}(i')$.

答案为 $|S'| + \sum_{\text{count}(i') \neq \text{count}(i)} \text{count}(i') = \text{count}(i_{max})$.

如果 $\text{count}(i_{max}) < \sum_{\text{count}(i') \neq \text{count}(i)} \text{count}(i')$,