

兰州大学第一届“飞马杯”程序设计竞赛

题解

2021.05.29



规则	ACM 个人赛
语言	C/C++、Java、Python
题目	12
时长	5 小时

为了响应教育部提出“新工科”教育战略，培养和展示我大学生分析、解决问题和计算机编程的能力，鼓励和培养创新思维，丰富校园学术气氛，造就具有综合素质的面向 21 世纪的计算机人才，兰州大学将于 2021 年 05 月 29 日举办兰州大学第一届“飞马杯”程序设计竞赛，面向兰州大学全体学生。

(感谢兰州大学信息科学与工程学院马俊老师提供的技术支持)

Accepted

Wrong Answer

Runtime Error

Time Limit Exceeded

Memory Limit Exceeded

Compile Error

目录

- A - ★★比赛新机制★★
- B - ★★体育课排队★★
- C - ★★生命的游戏★★
- D - ★★飞马祝福语★★
- E - ★★序列大团结★★
- F - ★★飞马分隔符★★
- G - ★★糖果魔法阵★★
- H - ★★温暖的力量★★
- I - ★★平形四边行★★
- J - ★★翻滚吧硬币★★
- K - ★★快乐苹果树★★
- L - ★★星星收集者★★

A - ★★比赛新机制★★

时间限制: 1000ms

空间限制: 256MB

难度:

★★★★☆

☆☆☆☆☆

标签: 递推 前缀和

题解

记 $sum = \sum_{i=1}^n a_i$, 先考虑正序的情况:

若答题顺序为 $a_1, a_2, \dots, a_{n-1}, a_n$, 那么总罚时为 $S_1 = na_1 + (n-1)a_2 + \dots + 2a_{n-1} + a_n$;

若答题顺序为 $a_2, a_3, \dots, a_n, a_1$, 那么总罚时为 $S_2 = a_1 + na_2 + \dots + 3a_{n-1} + 2a_n$ 。

不难发现, $S_2 = S_1 - na_1 + sum$, 以此类推, 可得 $S_{i+1} = S_i - na_i + sum$, 反序同理。

于是, 在每次递推时更新最小值即可。

时间复杂度 $O(n)$ 。

参考代码

C++

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define maxn 500005
5  #define INF 123456789000000000
6  ll T, n, sum, isum, risum, a[maxn], ans;
7  int main()
8  {
9      scanf("%lld", &T);
10     while (T--) {
11         scanf("%lld", &n);
12         sum = isum = risum = 0;
13         ans = INF;
14         for (ll i = 1; i <= n; i++) {
15             scanf("%lld", &a[i]);
16             sum += a[i];
17             isum += i * a[i];
18             risum += (n - i + 1) * a[i];
19         }
20         for (ll i = n; i >= 1; i--) {
21             isum += sum - n * a[i];
22             ans = min(ans, isum);
23         }
24         for (ll i = 1; i <= n; i++) {
25             risum += sum - n * a[i];
26             ans = min(ans, risum);
27         }
28     }
```

```
28     printf("%lld\n", ans);
29 }
30 return 0;
31 }
32
```

B - ★★体育课排队★★

时间限制: 4000ms

空间限制: 256MB

难度:

★★★★★

★★☆☆☆

标签: [二分图](#) [网络流](#) [二分答案](#) [Special Judge](#)

题解

根据题意，考虑将 n 位同学视为二分图的左部，将 n 个位置视为二分图的右部，将该问题转化为二分图匹配问题。

由于匹配是同时开始进行的，因此每次二分图匹配的时间只与最长的那组匹配有关，要求该时间的最小值，使用 KM 等方法难以解决。于是考虑二分答案，每次只将所有范围内的边加入二分图，然后进行二分图匹配。若能够完全匹配，说明在当前时间约束下可以完成排队任务；否则应扩大时间约束。

使用匈牙利 *Hungary* 算法，总时间复杂度 $O(n^3 \log n)$ ，无法通过。

注意到该图为二分图，可以用优化过的网络流算法进行二分图匹配，单次复杂度只有 $O(n^{\frac{5}{2}})$ ，相比匈牙利 *Hungary* 算法的 $O(n^3)$ 具有明显优势。匹配完成后，利用残余网络按要求输出匹配情况。

时间复杂度 $O(n^{\frac{5}{2}} \log n)$ 。

参考代码

C++

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define maxn 2005
4  #define maxm 2004005
5  #define INF 1234567890
6  int T, n, m, ss[maxn], sx[maxn], sy[maxn], sum[maxn], l, r, s, t, maxflow, cnt = 1,
    x[maxn], y[maxn], head[maxn], dis[maxn], cur[maxn], gap[maxn], ans[maxn];
7  struct Edge {int u, v, w, pre, next;} edge[maxm];
8  inline void add(int u, int v, int w) {
9      edge[++cnt].u = u;
10     edge[cnt].v = v;
11     edge[cnt].w = w;
12     edge[cnt].pre = w;
13     edge[cnt].next = head[u];
14     head[u] = cnt;
15     return;
16 }
17 inline int Dfs(int u, int lim) {
18     if (!lim || u == t) return lim;
19     int flow = 0, f;
20     for (int i = cur[u]; i; i = edge[i].next) {
21         int v = edge[i].v, w = edge[i].w;
22         cur[u] = i;
23         if (dis[v] + 1 == dis[u] && (f = Dfs(v, min(lim, w)))) {
24             flow += f;
```

```

25         lim -= f;
26         edge[i].w -= f;
27         edge[i ^ 1].w += f;
28         if (!lim) return flow;
29     }
30 }
31 gap[dis[u]]--;
32 if (!gap[dis[u]]) dis[s] = t + 1;
33 dis[u]++;
34 gap[dis[u]]++;
35 return flow;
36 }
37 inline void Bfs() {
38     queue<int>q;
39     for(int i = 1; i <= t; i++) {
40         dis[i] = INF;
41         gap[i] = 0;
42     }
43     dis[t] = 0;
44     gap[0] = 1;
45     q.push(t);
46     while (!q.empty()) {
47         int u = q.front();
48         q.pop();
49         for (int i = head[u]; i; i = edge[i].next) {
50             int v = edge[i].v;
51             if (dis[v] >= INF) {
52                 dis[v] = dis[u] + 1;
53                 gap[dis[v]]++;
54                 q.push(v);
55             }
56         }
57     }
58     return;
59 }
60 inline void ISAP() {
61     Bfs();
62     while (dis[s] < t) {
63         for (int i = 1; i <= t; i++)
64             cur[i] = head[i];
65         maxflow += Dfs(s, INF);
66     }
67 }
68 inline bool Check(int mid) {
69     cnt = 1;
70     maxflow = 0;
71     for (int i = 1; i <= t; i++)
72         head[i] = dis[i] = cur[i] = gap[i] = 0;
73     for (int i = 1; i <= n; i++)
74         for (int j = 1; j <= m; j++)
75             for (int k = 1; k <= ss[j]; k++) {
76                 int w = abs(sx[j] + k - 1 - x[i]) + abs(sy[j] - y[i]);
77                 if (w > mid) continue;
78                 add(i, n + sum[j-1] + k, 1);
79                 add(n + sum[j-1] + k, i, 0);

```

```

80     }
81     for(int i = 1; i <= n; i++) {
82         add(s, i, 1);
83         add(i, s, 0);
84         add(i + n, t, 1);
85         add(t, i + n, 0);
86     }
87     ISAP();
88     if (maxflow == n) return 1;
89     else return 0;
90 }
91 int main()
92 {
93     cin >> T;
94     while (T--) {
95         cin >> n >> m;
96         l = 0, r = 5000;
97         for (int i = 1; i <= n; i++)
98             cin >> x[i] >> y[i];
99         for (int i = 1; i <= m; i++) {
100             cin >> ss[i] >> sx[i] >> sy[i];
101             sum[i] = sum[i-1] + ss[i];
102         }
103         s = 2 * n + 1;
104         t = 2 * n + 2;
105         while (l < r) {
106             int mid = (l + r) >> 1;
107             if (Check(mid)) r = mid;
108             else l = mid + 1;
109         }
110         cout << l << endl;
111         Check(l);
112         for (int i = 1; i <= n; i++)
113             for (int j = head[i]; j; j = edge[j].next) {
114                 int v = edge[j].v, w = edge[j].w, pre = edge[j].pre;
115                 if (pre > w) ans[v - n] = i;
116             }
117         for(int i = 1; i <= m; i++)
118             for(int j = 1; j <= ss[i]; j++)
119                 cout << ans[sum[i-1] + j] << " \n"[j == ss[i]];
120     }
121     return 0;
122 }
123

```

C - ★★生命的游戏★★

时间限制: 1000ms

空间限制: 256MB

难度:

★★★★☆

☆☆☆☆☆

标签: 暴力 模拟

题解

按要求暴力模拟即可。

时间复杂度 $O(n^2k)$ 。

参考代码

C++

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define maxn 105
4  int T, n, k, ans, a[maxn][maxn], now[maxn][maxn], s[maxn][maxn], di[] = {-1, -1, -1,
5  0, 0, 1, 1, 1}, dj[] = {-1, 0, 1, -1, 1, -1, 0, 1};
6  inline bool Check() {
7      for (int i = 0; i < n; i++)
8          for (int j = 0; j < n; j++)
9              if (a[i][j] != now[i][j]) return 0;
10     return 1;
11 }
12 inline void Solve() {
13     for (int i = 0; i < n; i++)
14         for (int j = 0; j < n; j++)
15             s[i][j] = 0;
16     for (int i = 0; i < n; i++)
17         for (int j = 0; j < n; j++)
18             if (now[i][j]) {
19                 for (int k = 0; k < 8; k++) {
20                     int ni = (i + di[k] + n) % n, nj = (j + dj[k] + n) % n;
21                     s[ni][nj]++;
22                 }
23             }
24     for (int i = 0; i < n; i++)
25         for (int j = 0; j < n; j++)
26             if (s[i][j] == 3) now[i][j] = 1;
27             else if (s[i][j] < 2 || s[i][j] > 3) now[i][j] = 0;
28 }
29 int main()
30 {
31     cin >> T;
32     while (T--) {
33         cin >> n >> k;
34         ans = 0;
35         for (int i = 0; i < n; i++)
```



```
35         for (int j = 0; j < n; j++) {
36             cin >> a[i][j];
37             now[i][j] = a[i][j];
38         }
39     do {
40         Solve();
41         ans++;
42     }
43     while (!Check() && --k);
44     if (k) cout << "YES" << endl << ans << endl;
45     else cout << "NO" << endl;
46 }
47 return 0;
48 }
49
```

D - ★★飞马祝福语★★

时间限制: 4000ms

空间限制: 256MB

难度:

★★★★★

★★★★☆

标签: 递推 动态规划 区间动态规划 矩阵 线段树

题解

设祝福信为 s , FeiMa 为 x , 下标均从 1 开始。

令 $dp[i][j]$ 表示 s 的前 i 位以 FeiMa 中第 j 个字符结尾的不同子序列的数量 ($j = 0$ 表示没有出现其中任何一个字符), 那么状态转移方程为:

$$dp[i][j] = \begin{cases} dp[i-1][j-1] + dp[i-1][j], & s[i] = x[j] \\ dp[i-1][j], & else \end{cases}$$

答案即为 $dp[n][5]$ 。

暴力求解, 时间复杂度 $O(qn)$, 无法通过。

考虑用矩阵进行状态转移, 设 $D[i] = [dp[i][0] \ dp[i][1] \ dp[i][2] \ dp[i][3] \ dp[i][4] \ dp[i][5]]$,

若 $s[i] = \text{F}$, 那么有:

$$D[i] = D[i-1] \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

同理, 若 $s[i] = \text{e}$, 那么有:

$$D[i] = D[i-1] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

以此类推, 根据 $s[i]$ 的不同取值, 共有 6 种转移矩阵。

于是将问题转化为矩阵的区间乘积维护, 考虑将矩阵作为线段树结点的元素。

进行区间修改使用矩阵快速幂求值然后赋值, 总时间复杂度 $O(6^3 q \log^2 n)$, 无法通过。

考虑预处理出 6 种转移矩阵的 $0 \sim n$ 次方, 在每次区间修改时可以直接 $O(6^2)$ 赋值。

时间复杂度 $O(6^3 q \log n)$ 。

另外, 此题还有复杂度更优的算法: 线段树维护区间动态规划。

设祝福信为 s , FeiMa 为 x , s 的下标从 1 开始, x 的下标从 0 开始。

使用线段树维护区间动态规划。令 $dp[rt][i][j]$ 表示线段树的 rt 节点包含的字符串范围内，拥有字符串 x 的 i 位到 j 位区间的子序列数量（例如， $i = 0, j = 0$ 表示 rt 节点中 **F** 的数量， $i = 1, j = 1$ 表示节点中 **e** 的数量， $i = 2, j = 3$ 表示节点中子序列为 **im** 的数量），那么状态转移方程为：

$$dp[rt][i][j] = dp[rt \ll 1][i][j] + dp[rt \ll 1][i][j] + \sum_{k=i}^{j-1} dp[rt \ll 1][i][k] * dp[rt \ll 1][k+1][j]$$

答案即为 $dp[1][0][4]$ 。

时间复杂度 $O(5^3 q \log n)$ 。

参考代码

C++ - 1

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define maxn 100005
4  #define p 998244353
5  struct Matrix {
6      int a[6][6], n = 5;
7      Matrix(const int & N = 5) : n(N) {memset(a, 0, sizeof(a));}
8      void operator = (const Matrix & x) {
9          n = x.n;
10         for(int i = 0; i <= n; i++)
11             for(int j = 0; j <= n; j++)
12                 a[i][j] = x.a[i][j];
13     }
14     Matrix operator * (const Matrix & x) const {
15         Matrix ans = Matrix(n);
16         for(int i = 0; i <= n; i++)
17             for(int j = 0; j <= n; j++)
18                 for(int k = 0; k <= n; k++)
19                     ans.a[i][j] = (ans.a[i][j] + 1ll * a[i][k] * x.a[k][j]) % p;
20         return ans;
21     }
22 };
23 int T, n, q, a[maxn];
24 char x;
25 map<char, int> m;
26 Matrix pre[6][maxn];
27 struct Tree {Matrix mul; int tag;} t[maxn << 2];
28 inline int ls(int k) {return k << 1;}
29 inline int rs(int k) {return k << 1 | 1;}
30 inline void push_up(int l, int r, int k) {
31     t[k].mul = t[ls(k)].mul * t[rs(k)].mul;
32 }
33 inline void push_down(int l, int r, int k) {
34     if (t[k].tag != -1) {
35         t[ls(k)].tag = t[k].tag;
36         t[rs(k)].tag = t[k].tag;
37         int mid = (l + r) >> 1;
38         t[ls(k)].mul = pre[t[k].tag][mid - 1 + 1];
39         t[rs(k)].mul = pre[t[k].tag][r - mid];
40         t[k].tag = -1;

```

```

41     }
42 }
43 inline void build(int l, int r, int k) {
44     t[k].tag = -1;
45     if (l == r) {
46         t[k].mul = pre[a[l]][1];
47         return;
48     }
49     int mid = (l + r) >> 1;
50     build(l, mid, ls(k));
51     build(mid + 1, r, rs(k));
52     push_up(l, r, k);
53 }
54 inline void update(int nl, int nr, int l, int r, int k, int x) {
55     if (nl <= l && r <= nr) {
56         t[k].mul = pre[x][r - l + 1];
57         t[k].tag = x;
58         return;
59     }
60     push_down(l, r, k);
61     int mid = (l + r) >> 1;
62     if (nl <= mid) update(nl, nr, l, mid, ls(k), x);
63     if (nr > mid) update(nl, nr, mid + 1, r, rs(k), x);
64     push_up(l, r, k);
65 }
66 int main()
67 {
68     ios::sync_with_stdio(false);
69     cin.tie(0);cout.tie(0);
70     for (int k = 0; k <= 5; k++) {
71         for(int i = 0; i <= 5; i++)
72             pre[k][0].a[i][i] = pre[k][1].a[i][i] = 1;
73         if (k) pre[k][1].a[k-1][k] = 1;
74     }
75     for(int k = 0; k <= 5; k++)
76         for(int i = 2; i < maxn; i++)
77             pre[k][i] = pre[k][i - 1] * pre[k][1];
78     m['F'] = 1, m['e'] = 2, m['i'] = 3, m['M'] = 4, m['a'] = 5;
79     cin >> T;
80     while (T--) {
81         cin >> n >> q;
82         for (int i = 1; i <= n; i++) {
83             cin >> x;
84             a[i] = m[x];
85         }
86         build(1, n, 1);
87         int l, r;
88         while (q--) {
89             cin >> l >> r >> x;
90             update(l, r, 1, n, 1, m[x]);
91             cout << t[1].mul.a[0][5] << endl;
92         }
93     }
94     return 0;
95 }

```

C++ - 2

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int mod = 998244353;
5  const int maxn = 1e5+100;
6  int T,n,m,q;
7  char ss[maxn];
8  char pp[] = "FeiMa";
9  ll pro[maxn<<2][5][5];
10 char lazy[maxn<<2];
11 void build(int rt,int l,int r)
12 {
13     lazy[rt] = 0;
14     if(l==r)
15     {
16         for(int i=0; i<5; i++) for(int j=i; j<5; j++) pro[rt][i][j] = 0;
17         for(int i=0; i<5; i++) pro[rt][i][i] = ss[l]==pp[i];
18         return;
19     }
20     int mid = (l+r)>>1;
21     build(rt<<1,l,mid);
22     build(rt<<1|1,mid+1,r);
23     for(int i=0; i<5; i++)
24     {
25         for(int j=i; j<5; j++)
26         {
27             pro[rt][i][j] = (pro[rt<<1][i][j]+pro[rt<<1|1][i][j])%mod;
28             for(int k=i; k<j; k++) pro[rt][i][j] = (pro[rt][i][j]+pro[rt<<1][i]
[k]*pro[rt<<1|1][k+1][j])%mod;
29         }
30     }
31 }
32 void push_down(int rt,int l,int r)
33 {
34     if(!lazy[rt]) return;
35     lazy[rt<<1] = lazy[rt<<1|1] = lazy[rt];
36     int mid = (l+r)>>1;
37     int sz = mid-l+1;
38     for(int i=0; i<5; i++) for(int j=i; j<5; j++) pro[rt<<1][i][j] = 0;
39     for(int i=0; i<5; i++) pro[rt<<1][i][i] = (lazy[rt]==pp[i])*sz;
40     sz = r-mid;
41     for(int i=0; i<5; i++) for(int j=i; j<5; j++) pro[rt<<1|1][i][j] = 0;
42     for(int i=0; i<5; i++) pro[rt<<1|1][i][i] = (lazy[rt]==pp[i])*sz;
43     lazy[rt] = 0;
44 }
45 void update(int rt,int l,int r,int fr,int to,char ch)
46 {
47     if(fr<=l && r<=to)
48     {
49         for(int i=0; i<5; i++) for(int j=i; j<5; j++) pro[rt][i][j] = 0;

```

```

50     lazy[rt] = ch;
51     int sz = r-l+1;
52     for(int i=0; i<5; i++) pro[rt][i][i] = (ch==pp[i])*sz;
53     return;
54 }
55 push_down(rt,l,r);
56 int mid = (l+r)>>1;
57 if(mid>=fr) update(rt<<1,l,mid,fr,to,ch);
58 if(mid+1<=to) update(rt<<1|1,mid+1,r,fr,to,ch);
59 for(int i=0; i<5; i++)
60 {
61     for(int j=i; j<5; j++)
62     {
63         pro[rt][i][j] = (pro[rt<<1][i][j]+pro[rt<<1|1][i][j])%mod;
64         for(int k=i; k<j; k++) pro[rt][i][j] = (pro[rt][i][j]+pro[rt<<1][i]
[k]*pro[rt<<1|1][k+1][j])%mod;
65     }
66 }
67 }
68 char ch[2];
69 int main()
70 {
71     scanf("%d",&T);
72     while(T--)
73     {
74         scanf("%d%d%s",&n,&q,ss + 1);
75         build(1,1,n);
76         int l,r;
77         while(q--)
78         {
79             scanf("%d%d%s",&l,&r,ch);
80             update(1,1,n,l,r,ch[0]);
81             printf("%lld\n",pro[1][0][4]);
82         }
83     }
84     return 0;
85 }
86

```

E - ★★序列大团结★★

时间限制: 1000ms

空间限制: 256MB

难度:

★★★★★

☆☆☆☆☆

标签: 哈希Hash

题解

先考虑序列中无法被 k 整除的数, 设 $s[i][v]$ 为序列前 i 项中数值 v 出现的次数模 k 的余数, 当 $s[i]$ 和 $s[j]$ ($i < j$) 完全相同时 (即每个数值 v 出现的次数模 k 的余数都相同), 子序列 $E_{i+1}, E_{i+2}, \dots, E_j$ 就是“团结”的。

考虑使用滚动数组, 将 $s[i][v]$ 优化至 $s[v]$, 从首项开始遍历并维护 s , 同时对 s 的内容进行 *Hash* 处理, 将处理后的结果保存, 记 $num[hash(s)]$ 为到目前为止 $hash(s)$ 出现的次数, 那么以当前项为右端点的团结的子序列的数量为 $num[hash(s)] - 1$. $hash(s)$ 可用 *map* 存储。

而能够被 k 整除的数, 不影响 *hash* 值, 但也同样需要计算答案。

时间复杂度 $O(n \log n)$ 。

参考代码

C++

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  typedef unsigned long long ull;
4  #define p 998244353
5  ull T, n, k, x, ans, now, hs;
6  map<ull, int>num;
7  map<int, int>s;
8  inline ull ksm(ull x, ull k) {
9      ull res = 1;
10     for(; k >>= 1, x = x * x)
11         if (k & 1) res = res * x;
12     return res;
13 }
14 int main(){
15     cin >> T;
16     while (T--) {
17         cin >> n >> k;
18         num.clear();
19         s.clear();
20         ans = now = 0;
21         num[0] = 1;
22         for (int i = 1; i <= n; i++) {
23             cin >> x;
24             if (x % k) {
25                 hs = ksm(p, x);
26                 now -= hs * s[x];
27                 s[x] = (s[x] + 1) % k;
```

```
28         now += hs * s[x];
29     }
30     ans += num[now];
31     num[now]++;
32 }
33 cout << ans << endl;
34 }
35 return 0;
36 }
37
```


F - ★★飞马分隔符★★

时间限制: 1000ms

空间限制: 256MB

难度:

★★★★☆

☆☆☆☆☆

标签: 模拟 字符串 贪心

题解

从头到尾遍历，每次遇到字符 `a`，判断前面是否顺次出现过一组 `F`、`e`、`i`、`M`，若出现过，则在此处分隔，更新答案，并清空记录。

时间复杂度 $O(n)$ 。

参考代码

C++

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int T, n, ans, flag;
4  string s;
5  int main()
6  {
7      cin >> T;
8      while (T--) {
9          cin >> n >> s;
10         ans = flag = 0;
11         for (int i = 0; i < n; i++) {
12             if (s[i] == 'F' && flag == 0) flag++;
13             else if (s[i] == 'e' && flag == 1) flag++;
14             else if (s[i] == 'i' && flag == 2) flag++;
15             else if (s[i] == 'M' && flag == 3) flag++;
16             else if (s[i] == 'a' && flag == 4) {
17                 ans++;
18                 flag = 0;
19             }
20         }
21         cout << ans << endl;
22     }
23     return 0;
24 }
25
```

G - ★★糖果魔法阵★★

时间限制: 2000ms

空间限制: 256MB

难度:

★★★★★

★★★★☆

标签: 数学 数论 递推 二次剩余 光速幂 扩展欧拉定理 逆元

题解

根据题意, 定义 $G_n = \frac{G_{n-1}^4 + 12G_{n-1}^2 + 4}{4G_{n-1}(G_{n-1}^2 + 2)}$, 求出两个实数不动点为 $\sqrt{2}, -\sqrt{2}$ 。

构造 $\frac{G_n - \sqrt{2}}{G_n + \sqrt{2}} = \frac{G_{n-1}^4 - 4\sqrt{2}G_{n-1}^3 + 12G_{n-1}^2 - 8\sqrt{2}G_{n-1} + 4}{G_{n-1}^4 + 4\sqrt{2}G_{n-1}^3 + 12G_{n-1}^2 + 8\sqrt{2}G_{n-1} + 4} = \left(\frac{G_{n-1} - \sqrt{2}}{G_{n-1} + \sqrt{2}}\right)^4 = \left(\frac{G_0 - \sqrt{2}}{G_0 + \sqrt{2}}\right)^{4^n}$

推出通项为 $G_n = \sqrt{2} \frac{(G_0 + \sqrt{2})^{4^n} + (G_0 - \sqrt{2})^{4^n}}{(G_0 + \sqrt{2})^{4^n} - (G_0 - \sqrt{2})^{4^n}}$

由于 2 是模 998244353 的二次剩余, 将 $G_0 = 1$ 与 $\sqrt{2} \equiv 116195171 \pmod{998244353}$ 代入得:

$$G_n = 116195171 \frac{116195172^{4^n} + 882049183^{4^n}}{116195172^{4^n} - 882049183^{4^n}} \pmod{998244353}$$

考虑指数 4^n , 根据欧拉定理, 可对其取模, 即 $4^n \iff 4^n \pmod{\varphi(998244353)} \iff 4^n \pmod{998244352}$

考虑指数 n , 根据扩展欧拉定理, 当 $n \geq \varphi(998244352) = 402653184$ 时可对其取模, 即

$n \iff n \pmod{402653184 + 402653184}$ 。

使用快速幂求出每天结束时的 G_k , 进而求出下一天的魔力值, 时间复杂度 $O(n \log P)$, 无法通过。

考虑对 4^n 进行模 998244352 的光速幂处理, 对 116195172^n 和 882049183^n 进行模 998244353 的光速幂处理。

在每天结束时, 用光速幂求出相应的 $(116195172^{4^k} - 882049183^{4^k})mG_k \pmod{998244353}$, 便可得知下一天的情况。

直到最后一天结束时, 用光速幂和费马小定理求出 G_n 。

时间复杂度 $O(n + q \log P)$ 。

参考代码

C++

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define maxn 31625
5 #define two 116195171
6 #define ka 116195172
7 #define kb 882049183
8 #define phi 402653184
9 #define p 998244353
10 ll s, d, q, n, a[maxn + 5][2], b[maxn + 5][2], c[maxn + 5][2], k, t, sum, ans;
```

```

11 bool flag;
12 inline void Init() {
13     a[0][0] = a[0][1] = b[0][0] = b[0][1] = c[0][0] = c[0][1] = 1;
14     for (ll i = 1; i <= maxn; i++) {
15         a[i][0] = a[i - 1][0] * ka % p;
16         b[i][0] = b[i - 1][0] * kb % p;
17         c[i][0] = c[i - 1][0] * 4 % (p - 1);
18     }
19     for (ll i = 1; i <= maxn; i++) {
20         a[i][1] = a[i - 1][1] * a[maxn][0] % p;
21         b[i][1] = b[i - 1][1] * b[maxn][0] % p;
22         c[i][1] = c[i - 1][1] * c[maxn][0] % (p - 1);
23     }
24 }
25 inline ll ksm(ll x, ll k) {
26     ll res = 1 % p; x %= p;
27     for (; k; k >>= 1, x = x * x % p)
28         if (k & 1) res = res * x % p;
29     return res;
30 }
31 int main()
32 {
33     Init();
34     cin >> s >> d >> q;
35     while (q--) {
36         cin >> n;
37         sum = s;
38         k = flag = 0;
39         for (ll i = 1; i <= n; i++) {
40             k += sum / d;
41             if (k >= phi) flag = 1;
42             k %= phi;
43             if(flag) k += phi;
44             t = c[k % maxn][0] * c[k / maxn][1] % (p - 1);
45             sum = (((a[t % maxn][0] * a[t / maxn][1] % p) + (b[t % maxn][0] * b[t /
maxn][1] % p)) * two % p) * i % p;
46         }
47         ans = (sum * ksm(n, p - 2) % p) * ksm((((a[t % maxn][0] * a[t / maxn][1] % p)
- (b[t % maxn][0] * b[t / maxn][1] % p) + p) % p) % p, p - 2)%p;
48         cout << ans << endl;
49     }
50     return 0;
51 }
52

```

H - ★★温暖的力量★★

时间限制: 1000ms

空间限制: 256MB

难度:

★★★★★

★★★★★

标签: 数学

题解

要求将 n 分为尽可能多的质数的和。

当 n 为大于 3 的奇数时, 可以分为若干个 2 和一个 3; 当 n 为大于 3 的偶数时, 可以分为若干个 2。

不难得出答案:

$$Answer = \begin{cases} -1, & n \leq 3 \\ \left\lfloor \frac{n}{2} \right\rfloor, & n > 3 \end{cases}$$

时间复杂度 $O(T)$ 。

参考代码

C++

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int T, n;
4  int main()
5  {
6      cin >> T;
7      while (T--) {
8          cin >> n;
9          if (n <= 3) cout << -1 << endl;
10         else cout << n / 2 << endl;
11     }
12     return 0;
13 }
14
```

I - ★★平形四边行★★

时间限制: 1000ms

空间限制: 256MB

难度:

★★★★★

☆☆☆☆☆

标签: 暴力 Special Judge

题解

四个点能形成“平形四边行”的充要条件是，存在一种方案，将四个点均分为两组，每组的两个点形成一条线段，这两条线段的中点重合。

注意到桶的大小只有 4000×4000 ，即第 $4000 \times 4000 + 1$ 个点落下时，一定存在重合的中点，从而构成一组解。另外，桶不可能被完全填满，因此复杂度远远小于预期。

于是 n^2 暴力枚举每组点，存储所形成线段的中点，直到存在重合的中点，同时应注意过滤掉存在交集的两组点。

时间复杂度 $O(4000^2)$ 。

参考代码

C++

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define maxn 100005
4  int n, x[maxn], y[maxn];
5  pair<int, int>v[4005][4005];
6  int main()
7  {
8      cin >> n;
9      for (int i = 1; i <= n; i++) {
10         cin >> x[i] >> y[i];
11         for (int j = 1; j < i; j++) {
12             int xx = x[i] + x[j] + 2000;
13             int yy = y[i] + y[j] + 2000;
14             if (i == v[xx][yy].first || j == v[xx][yy].first) continue;
15             if (i == v[xx][yy].second || j == v[xx][yy].second) continue;
16             if (v[xx][yy].first) {
17                 cout << "YES" << endl << i << ' ' << j << ' ' << v[xx][yy].first << '
18                 << v[xx][yy].second << endl;
19                 return 0;
20             }
21             v[xx][yy] = make_pair(i,j);
22         }
23     }
24     cout << "NO" << endl;
25     return 0;
26 }
```

J - ★★翻滚吧硬币★★

时间限制: 1000ms

空间限制: 256MB

难度:

★★★★☆

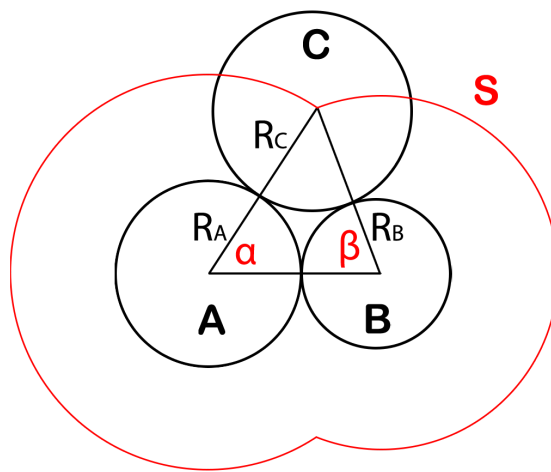
☆☆☆☆☆

标签: 数学 计算几何 Special Judge

题解

硬币悖论问题。

如图, 三枚硬币 A, B, C , 半径分别为 R_A, R_B, R_C 。



现在, 让硬币 C 沿着硬币 A, B 的边界翻滚, 硬币 C 的圆心的运动轨迹为曲线 S , 记其长度为 s , 那么翻滚的圈数即为 $\frac{s}{2\pi R_C}$ 。

由弧长公式可得 $s = 2(\pi - \alpha)(R_A + R_C) + 2(\pi - \beta)(R_B + R_C)$, 其中 α, β 可由余弦定理推出:

$$\begin{cases} \alpha = \arccos\left(\frac{(R_A + R_B)^2 + (R_A + R_C)^2 - (R_B + R_C)^2}{2(R_A + R_B)(R_A + R_C)}\right) \\ \beta = \arccos\left(\frac{(R_B + R_A)^2 + (R_B + R_C)^2 - (R_A + R_C)^2}{2(R_B + R_A)(R_B + R_C)}\right) \end{cases}$$

要想使得圈数最小, 只需固定半径较小的两个硬币, 让半径最大的硬币沿其边界翻滚即可。

时间复杂度 $O(T)$ 。

参考代码

C++

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 const long double pi=acosl(-1);
4 int T, r[5];
5 long double ans, a, b, c, x, y;
6 int main()
7 {
```

```
8     cin >> T;
9     while (T--) {
10         cin >> r[1] >> r[2] >> r[3];
11         sort(r + 1, r + 3 + 1);
12         a = r[1] + r[2], b = r[1] + r[3], c = r[2] + r[3];
13         x = acosl((a * a + b * b - c * c) / (2.01 * a * b));
14         y = acosl((a * a + c * c - b * b) / (2.01 * a * c));
15         ans = ((pi - x) * b + (pi - y) * c) / (pi * r[3]);
16         cout << fixed << setprecision(15) << ans << endl;
17     }
18     return 0;
19 }
20
```

K - ★★快乐苹果树★★

时间限制: 1000ms

空间限制: 256MB

难度:

★★★★★

★★★★☆

标签: 数学 逆元 概率论 数学期望 树链剖分 树状数组 线段树

题解

定义 $size[i]$ 表示以结点 i 为根的子树的大小, $son[i]$ 表示结点 i 的所有直接子结点(距离为 1)组成的集合, $sons[i]$ 表示以结点 i 为根的子树内所有节点组成的集合。

对于操作一, 可分为以下两种情况:

- 当选取结点 $u \in sons[w], w \in son[F]$, 那么点集 $\{v \mid v \in T, lca(u, v) = F\}$ 表示的是 $sons[F] - sons[w]$, 此时该操作与 u 无关, 只与 w 有关。结点 $u \in sons[w]$ 的概率为 $\frac{size[w]}{size[F]}$, 于是点集 $sons[F] - sons[w]$ 中所有结点的快乐值增加 $\frac{size[w]}{size[F]} \frac{1}{size[F] - size[w]} k$ 。
- 当选取结点 $u = F$ 时, 概率为 $\frac{1}{size[F]}$, 点集 $\{v \mid v \in T, lca(u, v) = F\}$ 表示的是 $sons[F]$, 于是点集 $sons[F]$ 中所有结点的快乐值增加 $\frac{1}{size[F]} \frac{1}{size[F]} k$ 。

预处理出各子树大小以及对应逆元, 暴力求解, 总时间复杂度 $O(n + qn^2)$, 无法通过。

考虑使点集 $sons[F]$ 中所有结点的快乐值都加上该值, 然后再使点集 $sons[w]$ 中所有结点的快乐值减去该值。

如此一来, 对于每次操作一, 使点集 $sons[F]$ 中所有结点的快乐值增加

$$\frac{1}{size[F]} \frac{1}{size[F]} k + \sum_{w \in son[F]} \frac{size[w]}{size[F]} \frac{1}{size[F] - size[w]} k,$$

随后, 遍历 $w \in son[F]$, 使点集 $sons[w]$ 中所有结点的快乐值减去 $\frac{size[w]}{size[F]} \frac{1}{size[F] - size[w]} k$ 。

总时间复杂度 $O(n + qn)$, 无法通过。

考虑将子树按大小分块, 在每次进行子树更新时使用线段树或树状数组维护, 总时间复杂度

$O(n \log n + q\sqrt{n} \log n)$, 可以通过, 但不稳定且代码实现较为困难。

考虑进行树链剖分, 将轻儿子 x 懒标记的值放置在 F 结点即 $top[fa[x]]$ 结点, 重儿子 $\log n$ 更新。查询时, 遍历 $top[fa[x]]$ 节点, 减去对应的懒标记的值。

时间复杂度 $O(n + q \log n)$ 。

参考代码

C++

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define maxn 100005
5 #define maxm 200005
```



```

6  #define p 998244353
7  ll T, n, q, inv[maxn], t[maxn], tag[maxn], fa[maxn], se[maxn], son[maxn], dfn[maxn],
   top[maxn], id, cnt, head[maxn], s1[maxn], s2[maxn], s3[maxn];
8  struct Edge {ll u, v, next;} edge[maxn];
9  inline void add(ll u, ll v) {
10     edge[++cnt].u = u;
11     edge[cnt].v = v;
12     edge[cnt].next = head[u];
13     head[u] = cnt;
14 }
15 inline void Get_Inv() {
16     inv[1] = 1;
17     for (ll i = 2; i < maxn; i++)
18         inv[i] = (p - p / i) * inv[p % i] % p;
19 }
20 inline ll lowbit(ll x) {return x & (-x);}
21 inline void change(ll x, ll val)
22 {
23     for (ll i = x; i <= n; i += lowbit(i))
24         t[i] = (t[i] + val) % p;
25 }
26 inline ll query(ll x) {
27     ll res = 0;
28     for (ll i = x; i; i -= lowbit(i))
29         res = (res + t[i]) % p;
30     return res;
31 }
32 inline void update(ll l, ll r, ll val) {
33     change(l, val);
34     change(r + 1, p - val);
35 }
36 inline void Dfs1(int u, int f) {
37     fa[u] = f;
38     se[u] = 1;
39     son[u] = s3[u] = 0;
40     for (ll i = head[u]; i; i = edge[i].next) {
41         ll v = edge[i].v;
42         if (v == f) continue;
43         Dfs1(v, u);
44         se[u] += se[v];
45         if (se[v] > se[son[u]]) son[u] = v;
46     }
47     s1[u] = inv[se[u]];
48     for (ll i = head[u]; i; i = edge[i].next) {
49         ll v = edge[i].v;
50         if (v == f) continue;
51         s2[v] = (se[v] * s1[u] % p) * inv[se[u] - se[v]] % p;
52         s3[u] = (s3[u] + s2[v]) % p;
53     }
54 }
55 inline void Dfs2(int u, int t) {
56     top[u] = t;
57     dfn[u] = ++id;
58     if (!son[u]) return;
59     Dfs2(son[u], t);

```

```

60     for (ll i = head[u]; i; i = edge[i].next) {
61         ll v = edge[i].v;
62         if (v == son[u] || v == fa[u]) continue;
63         Dfs2(v, v);
64     }
65 }
66 int main()
67 {
68     Get_Inv();
69     cin >> T;
70     while (T--) {
71         cin >> n >> q;
72         cnt = id = 0;
73         for (ll i = 0; i <= n; i++)
74             head[i] = t[i] = tag[i] = 0;
75         ll u, v;
76         for (ll i = 1; i < n; i++) {
77             cin >> u >> v;
78             add(u, v);
79             add(v, u);
80         }
81         Dfs1(1, 0);
82         Dfs2(1, 1);
83         ll op, f, k;
84         while (q--) {
85             cin >> op;
86             if (op == 1) {
87                 cin >> f >> k;
88                 update(dfn[f], dfn[f] + se[f] - 1, (s3[f] * k % p + (s1[f] * s1[f] %
p) * k % p) % p);
89                 if (son[f])
90                     update(dfn[son[f]], dfn[son[f]] + se[son[f]] - 1, (p - s2[son[f]]
* k % p) % p);
91                 tag[f] = (tag[f] + k) % p;
92             }
93             else {
94                 cin >> f;
95                 ll res = query(dfn[f]);
96                 for (ll i = top[f]; fa[i]; i = top[fa[i]])
97                     res = (res + (p - s2[i] * tag[fa[i]] % p) % p) % p;
98                 cout << res << endl;
99             }
100         }
101     }
102     return 0;
103 }
104

```

L - ★★星星收集者★★

时间限制: 1000ms

空间限制: 256MB

难度: ★★★★★

☆☆☆☆☆

标签: 博弈论 动态规划

题解

令星星的总和为 sum 。

首先, 因为双方的星星总和一定是 sum , 如果 $x \geq \frac{sum}{2}$, 那么当 $a \geq \frac{sum}{2}$ 时 $L.St$ 就获胜了。

反之, 如果 $x < \frac{sum}{2}$, 当 $a < \frac{sum}{2}$ 时 $L.St$ 获胜。

其余情况 $L.Ts$ 获胜。

那么我们从 $L.St$ 的角度考虑, 他的策略根据 sum 与 x 的关系, 只有两种:

- 抓尽可能少的星星
- 抓尽可能多的星星 (也就是全部抓取)

$L.Ts$ 需要让 $L.St$ 在这两种策略下都不能获胜。

令添加星星数为 k , 首先可以发现, 当 $x \geq \frac{sum}{2}$ 时, $L.St$ 必胜 (全部抓取), 所以可以得到

$k > x + x - sum$ 。

当 $x \geq \frac{sum}{2}$ 时, 考虑 $L.Ts$ 在添加星星时的策略。

我们发现, 在此情况下, 双方都想最小化自己的星星数。那么 $L.Ts$ 会尽可能的将 k 分配给 $L.St$, 只要将 k 全部分配给第一个盒子就能做到。

我们用动态规划可以求出在原先状态下, a 最少可以抓取多少星星。

令这个值为 v , 则能得到 $\frac{k + sum}{2} < v + k$ 。

综合两种情况就能得到 k 的最小值

参考代码

C++

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 #define maxn 200005
5 ll n, x, a[maxn], sum[maxn], dp[maxn], last;
6 int main()
7 {
8     cin >> n >> x;
9     for (int i = 1; i <= n; i++) {
10         cin >> a[i];
11         sum[i] = sum[i - 1] + a[i];
12     }
13     last = n;
14     dp[n] = a[n];
15     for (ll i = n - 1; i; i--) {
16         dp[i] = sum[n] - sum[i - 1] - dp[last];
```

```
17         if (dp[i] > dp[last]) last = i;
18     }
19     cout << max(max(0ll, sum[n] - dp[1] - dp[1] + 1), max(0ll, 2 * x + 1 - sum[n])) <<
endl;
20     return 0;
21 }
22
```

