综合混练

人员

刘锦轩、范家郡、牛晓晨 到课, 刘智予 线上

上周作业检查

上上周作业链接: https://vjudge.net/contest/724229

Begin: 2025-06-21 08:30 CST ☆ 2025-0621 五队上课 (综合混练) End: 2026-01-15 16:30 CST

Elapsed: 13:23:26:18 Running Remaining: 194:08:33:41

Overview Problem Status Rank (13:23:26:16) Discuss Setting Clone Update Delete

Rank	Team	Score	Penalty	A 1 / 5	B 1 / 1	C 4 / 6	D 5 / 16	E 5 / 11
1	☆ ikunTLE (方冠霖)	4	68541		13:12:31:24	13:08:17:00	13:07:47:46 (-3)	7:08:45:08
2	☆ two_tiger (卢炫佑)	3	40839	13:11:05:14 (-2)			7:08:26:01 (-5)	7:10:48:27
3	☆ longlong_int (刘锦轩)	3	48875			13:11:16:31	13:09:03:46	7:02:14:44
4	☆ misaka16384 (黄诗琦)	2	21167				7:08:26:18 (-1)	7:07:40:56 (-1)
5	☆ Hanhj (韩鸿钜)	2	25423			7:08:43:38 (-1)		10:06:39:42
6	☆ zhn123bc (张皓宁)	1	10607	(-2)		7:08:27:57 (-1)		(-3)
7	☆ FeatherCrow (许岩)	1	10638				7:08:38:17 (-2)	
8	☆ lxr123bc (刘新睿)	0	0					(-2)

上周作业链接: https://vjudge.net/contest/725693

作业

课上讲了上周比赛的 A B C D E, 本周没有留新的课后作业, 同学们需要把之前的十几场比赛中的几十道题都补一补

课堂表现

同学们课上听讲都比较认真, 今天讲题时间比较久, 给同学们课上做题时间不多。

今天的 D E 2 道题都是之前 CSP-S 组的真题, 题目都不是很简单, 同学们课下要好好做一做这两道题。

课堂内容

U564690 异或和

针对每个二进制位进行考虑, 求每个二进制位上, 对最终的答案贡献了多少值

枚举不同的二进制位, 对于一个二进制位来说, 有多少贡献就是有多少区间异或值是 1 的区间

```
#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int maxn = 2e5 + 5;
int w[maxn];
int n;

LL calc(int k) {
  int odd = 0, even = 1;
  LL sum = 0;
  for (int i = 1; i <= n; ++i) {
    sum ^= w[i];
    if ((sum>>k) % 2 == 1) ++odd;
    else ++even;
  }
  return (LL)odd * even;
}

int main()
{
  cin >> n;
  LL sum = 0;
  for (int i = 1; i <= n; ++i) cin >> w[i], sum += w[i];

  LL res = 0;
  for (int i = 0; i <= 30; ++i)
    res += calc(i)*(1<<i);
  cout << res-sum << endl;
  return 0;
}
```

CF1846G Rudolf and CodeVid-23

通过状态压缩, 可以把相应的 01 字符串映射成对应的整数值

根据每种输入的药的效果, 可以建一个 1e3 * 1e3 大小的图

然后从 起点状态 往 终点状态(0) 跑一遍 bfs, 求最短路即可

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 1024 + 5;
struct node {
  int to, value;
```

```
  bool operator < (const node& p) const { return value < p.value; }
  bool operator > (const node& p) const { return value > p.value; }
};
vector<node> vec[maxn];
int n, m;

int strRead() {
  string str; cin >> str;
  int res = 0;
  for (int i = 0; i < n; ++i) res += (str[i]-'0')*(1<<i);
  return res;
}

int f[maxn];
bool st[maxn];

int dijkstra(int s) {
  memset(f, -1, sizeof(f)); memset(st, false, sizeof(st));
  priority_queue<node, vector<node>, greater<node>>q; f[s] = 0; q.push({s,f[s]});

  while (!q.empty()) {
    node u = q.top(); q.pop();
    int to = u.to, value = u.value;
    if (st[to]) continue;
    st[to] = true;
    for (node it : vec[to]) {
      if (f[it.to]==-1 || value+it.value<f[it.to]) {
        f[it.to] = value+it.value; q.push({it.to,f[it.to]});
      }
    }
  }

  return f[0];
}

void solve() {
  cin >> n >> m;
  int limit = (1<<n);
  for (int i = 0; i < limit; ++i) vec[i].clear();

  int s = strRead();
  while (m -- ) {
    int value; cin >> value;
    int a = strRead(), b = strRead();
    for (int i = 0; i < limit; ++i) {
      int j = ((i&(~a))|b);
      vec[i].push_back({j, value});
    }
  }

//  cout << "--------------------------- ";
  cout << dijkstra(s) << endl;
}
```

```
int main()
{
  int T; cin >> T;
  while (T -- ) solve();
  return 0;
}
```

CF1272E Nearest Opposite Parity

根据题目的跳跃情况, 建一个反图

建一个连接所有奇点的 超级奇点, 再建一个连接所有偶点的 超级偶点 (这里的 奇点 和 偶点 指的是点上的值是奇数还是偶数)

从超级奇点出发跑一遍 bfs, 能求出所有偶数点的结果; 从超级偶点出发跑一遍 bfs, 能求出所有奇数点的结果

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 2e5 + 5;
int w[maxn], f[maxn], ans[maxn];
vector<int> vec[maxn];

void bfs(int s) {
  memset(f, -1, sizeof(f));
  queue<int> q; q.push(s); f[s] = 0;
  while (!q.empty()) {
    int u = q.front(); q.pop();
    for (int i : vec[u]) {
      if (f[i] == -1) q.push(i), f[i] = f[u]+1;
    }
  }
}

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> w[i];

  int odd = n+1, even = n+2;
  for (int i = 1; i <= n; ++i) {
    if (i+w[i] <= n) vec[i+w[i]].push_back(i);
    if (i-w[i] >= 1) vec[i-w[i]].push_back(i);

    if (w[i]&1) vec[odd].push_back(i);
    else vec[even].push_back(i);
  }

  bfs(odd);
  for (int i : vec[even]) ans[i] = (f[i]==-1 ? -1 : f[i]-1);
```

```
  bfs(even);
  for (int i : vec[odd]) ans[i] = (f[i]==-1 ? -1 : f[i]-1);

  for (int i = 1; i <= n; ++i) cout << ans[i] << " ";
  cout << endl;
  return 0;
}
```

P11232 [CSP-S 2024] 超速检测

对于每辆车, 可以通过 二分 确定其超速的区间是哪一段

然后, 问题转化为了 有一些区间,想选最少的点覆盖这些区间

这就转化为了一个经典的贪心问题, 可以按照右端点排序然后贪心求

```
#include <bits/stdc++.h>

using namespace std;

const int N = 1e5 + 5;
int n, m, L, V;
int d[N], v[N], a[N], p[N];
struct node {
  int l, r;
  bool operator < (const node& pp) const { return r < pp.r; }
};

int get_div(int a, int b) { return a/b+1; }

int get_div_2(int a, int b) { return (a%b==0 ? a/b-1 : a/b); }

node get_rValue(int x) {
  int l = lower_bound(p+1, p+m+1, x) - p;
  return (l<=m ? node{l,m} : node{-1, -1});
}

node calc(int i) {
  if (a[i] == 0) {
    if (v[i] <= V) return {-1, -1};
    return get_rValue(d[i]);
  }

  if (a[i] > 0) {
    if (v[i] > V) return get_rValue(d[i]);

    int dis = get_div(V*V-v[i]*v[i], 2*a[i]);
    return get_rValue(d[i]+dis);
  }

  if (v[i] <= V) return {-1, -1};
```

```
  int dis = get_div_2(V*V-v[i]*v[i], 2*a[i]);
  int l_pos = lower_bound(p+1, p+m+1, d[i]) - p;
  int r_pos = upper_bound(p+1, p+m+1, d[i]+dis) - p - 1;

  if (l_pos>r_pos) return {-1, -1};
  return {l_pos, r_pos};
}

void solve() {
  cin >> n >> m >> L >> V;
  for (int i = 1; i <= n; ++i) scanf("%d%d%d", &d[i], &v[i], &a[i]);
  for (int i = 1; i <= m; ++i) scanf("%d", &p[i]);

  vector<node> vec;
  for (int i = 1; i <= n; ++i) {
    node t = calc(i);
    if (t.l!=-1 && t.r!=-1) vec.push_back(t);
  }

  sort(vec.begin(), vec.end());
  int res = 0, last = -1;
  for (node it : vec) {
    int l = it.l, r = it.r;
    if (l > last) last = r, ++res;
  }

  cout << vec.size() << " " << m - res << endl;
}

int main()
{
  int T; cin >> T;
  while (T -- ) solve();
  return 0;
}
```

P9753 [CSP-S 2023] 消消乐

O(n^3) 做法: O(n^2) 枚举每一段字符串, 按照类似于括号匹配的方法, 把每个字母入栈、出栈, 最后看字符串是否为空

O(n^2) 做法: O(n) 枚举起点 i, 从点 i 往后跑一遍 括号匹配 过程直到点 n, 看中间 栈 空过几次即可

O(n) 做法: 从 1 号点往后跑一遍 括号匹配 过程, 如果跑到位置 i 和跑到位置 j, 栈里面剩的元素都一样时, 说明 i+1~j 是可消除的一段字符串 (想快速判断栈里面元素是否一样， 可以把栈里面的字符串, 维护一个哈希值, 记录每个不同的哈希值出现过多少次即可)

```
#include <bits/stdc++.h>

using namespace std;
```

```cpp
typedef long long LL;
typedef unsigned long long ULL;
const int maxn = 2e6 + 5;
const int P = 131;
char s[maxn];
map<ULL, int> mp;

int main()
{
  int n; scanf("%d", &n);
  scanf("%s", s+1);

  stack<ULL> stk; stk.push(1); mp[1]++;
  stack<char> stk2; stk2.push(' ');
  for (int i = 1; i <= n; ++i) {
    if (stk2.top()==s[i]) stk.pop(), stk2.pop();
    else stk.push(stk.top()*P + (int)s[i]), stk2.push(s[i]);
    mp[stk.top()]++;
  }

  LL res = 0;
  for (auto it : mp) res += (LL)it.second*(it.second-1)/2;
  cout << res << endl;
  return 0;
}
```