综合混练

人员

叶乐山、齐中磊、李政毅、于飞岚、陈瀚霄、曹承贤、方冠霖、卢炫佑 到课

上周作业检查

上周作业链接: https://vjudge.net/contest/747412

开始时间 : 2025-09-13 08:30 CST　☆ 2025-0913 五队上课 (综合混练)　结束时间 : 2026-04-09 16:30 CST

已开始 : 14:09:04:33　进行中　还剩余 : 193:22:55:26

概览　题目　提交状态　排名 (14:09:04:26)　讨论　设置　克隆　更新　删除

Rank	Team	Score	Penalty	A 17 / 25	B 8 / 10	C 9 / 35	D 12 / 12	E 4 / 7	F 4 / 7	G 2 / 2
1	lxw123bc (卢新闻)	7	77452	1:08:22:07	8:07:48:51	14:01:52:02 (-2)	14:01:25:25	14:44:03 (-1)	7:01:24:44	**8:06:14:52**
2	sty0948 (隋天翼)	6	44616	7:03:40	1:14:28:14	14:01:54:00 (-2)	14:00:55:53	**8:17:12**	**14:17:15**	
3	shijixuan (史吉轩)	6	63971	2:34:14 (-2)	**0:13:19**	14:02:12:30 (-3)	14:01:12:09	6:12:49:20 (-2)	9:12:29:38 (-1)	
4	qizhongleiiii (齐中磊)	5	49371	8:42:13 (-1)	8:19:41 (-1)	14:08:59:11 (-3)		9:12:53:56	9:13:36:15 (-2)	
5	ikunTLE (方冠霖)	4	60677	1:43:55 (-1)		14:08:54:14 (-3)	14:00:55:11			13:14:24:32
6	123zhw (战鹤文)	3	21067	1:52:17 (-1)	11:46:57 (-1)		14:00:48:45			
7	FEILANYU (于飞岚)	3	21751	8:44:12	8:51:02		14:08:56:40			
8	two_tiger (卢炫佑)	3	41583	**1:21:12**		14:08:44:08 (-6)	14:08:57:46			
9	chx123bc (_陈瀚霄)	3	41765	8:43:52		14:08:51:30 (-6)	14:04:29:48			
10	Terry_MC (叶乐山)	3	41905	8:40:23		14:08:48:33	14:08:56:11			
11	longlong_int (刘锦轩)	3	43241	6:10:01:35	9:14:01:16		**14:00:39:02**			
12	qp_an (赵广宇)	2	15491	7:10:05	10:11:00:56					
13	WangYanzhen (王彦臻)	2	40473	14:01:17:08			14:01:16:15			
14	lzy1031 (李政毅)	2	41335	14:08:53:46			14:08:01:40			
15	XingChen_MoNian (...	1	95	1:35:03						
16	niuxiaochen (牛晓晨)	1	133	1:53:12 (-1)						
17	aiyishengaiyishi (王...	1	20231			**14:00:51:00 (-1)**				
18	ccx123bc (曹承贤)	1	20585	14:06:25:25 (-2)						

本周作业

https://vjudge.net/contest/751260, (课上讲了上周比赛的 B C D E F 题, 课后作业是本周比赛的 A B C D E 题)

课堂表现

今天的 C 题和 G 题要稍微复杂一些, 但是这两个题知识点都不难, 一个是搜索, 一个是二分+前缀和, 大家要认真补一补这两道题, 不要觉得麻烦就不做了

课堂内容

CF1245D Shichikuji and Power Grid

建一个虚拟源点 n+1 号点, 他与 i 号点的边权是 c[i], i 与 j 的边权是 dis(i,j)*(k[i]+k[j])

此时, 原问题被转化为一个最小生成树问题

这个图是一个完全图, 因此建议用 prim 算法实现

最后, 需要输出选择哪些点或哪些路径时, 可以用一个数组记录一下每个点的 d 值是由谁转移而来的

如果是由 n+1 号点转移而来的, 那么就是单独点这一个点; 如果不是 n+1 号点转移而来, 就相当于连接了上一个点和这一个点

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

using namespace std;

typedef long long LL;
const int maxn = 2000 + 5;
const LL inf = 0x3f3f3f3f3f3f3f3f;
int x[maxn], y[maxn], c[maxn], k[maxn], pre[maxn];
LL w[maxn][maxn], d[maxn];
bool st[maxn];

int dis(int a, int b) { return abs(x[a]-x[b]) + abs(y[a]-y[b]); }

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> x[i] >> y[i];
  for (int i = 1; i <= n; ++i) cin >> c[i];
  for (int i = 1; i <= n; ++i) cin >> k[i];

  for (int i = 1; i <= n+1; ++i) {
    for (int j = 1; j <= n+1; ++j) {
      if (i == j) continue;
      if (i == n+1) w[i][j] = c[j];
      else if (j == n+1) w[i][j] = c[i];
      else w[i][j] = (LL)dis(i,j) * (k[i]+k[j]);
    }
  }

  LL res = 0;
```

```
  memset(d, 0x3f, sizeof(d));
  vector<int> vec1;
  vector<pair<int,int>> vec2;
  for (int i = 0; i <= n; ++i) {
    int id = n+1;
    if (i) {
      for (int j = 1; j <= n+1; ++j) {
        if (st[j]) continue;
        if (id==n+1 || d[j]<d[id]) id = j;
      }
      res += d[id];
      if (pre[id] == n+1) vec1.push_back(id);
      else vec2.push_back({pre[id], id});
    }

    st[id] = true;
    for (int j = 1; j <= n+1; ++j) {
      if (id == j) continue;
      if (w[id][j] < d[j]) d[j] = w[id][j], pre[j] = id;
    }
  }

  cout << res << endl;
  cout << vec1.size() << endl;
  for (int i : vec1) cout << i << " ";
  cout << endl;
  cout << vec2.size() << endl;
  for (pair<int,int> it : vec2) cout << it.first << " " << it.second << endl;
  return 0;
}
```

UVA1599 理想路径 Ideal Path

先从 n 号点做一遍 bfs, 求出来每个点到 n 号点的最短距离 dis[i]

然后从 1 号点往后搜索遍历, 在保证最短路径的前提下, 找到颜色字典序最小的情况

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

using namespace std;

const int maxn = 1e5 + 5;
struct node {
  int to, c;
};
vector<node> vec[maxn];
int n, m;
int dis[maxn];
vector<int> ans;
bool st[maxn];
```

```
void bfs() {
  queue<int> q; q.push(n); dis[n] = 0;
  while (!q.empty()) {
    int u = q.front(); q.pop();
    for (node it : vec[u]) {
      if (dis[it.to] == -1) q.push(it.to), dis[it.to] = dis[u]+1;
    }
  }
}

void solve() {
  queue<int> q; q.push(1); st[1] = true;
  while (!st[n]) {
    vector<int> points;
    while (!q.empty()) points.push_back(q.front()), q.pop();
    int min_color = 1e9+10;
    for (int i : points) {
      for (node it : vec[i]) {
        if (dis[it.to] == dis[i]-1) min_color = min(min_color, it.c);
      }
    }

    ans.push_back(min_color);

    for (int i : points) {
      for (node it : vec[i]) {
        if (dis[it.to]==dis[i]-1 && !st[it.to] && it.c==min_color) q.push(it.to),
st[it.to] = true;
      }
    }
  }
}

int main()
{
  ios::sync_with_stdio(false);
  cin.tie(0);

  while (cin >> n >> m) {
    for (int i = 1; i <= n; ++i) {
      vec[i].clear(); dis[i] = -1; st[i] = false;
    }
    ans.clear();

    for (int i = 1; i <= m; ++i) {
      int a, b, c; cin >> a >> b >> c;
      vec[a].push_back({b,c}), vec[b].push_back({a,c});
    }

    bfs(); solve();

    cout << dis[1] << "\n";
    for (int i = 0; i < (int)ans.size(); ++i) {
      if (i) cout << " ";
```

```
      cout << ans[i];
    }
    cout << "\n";
  }
  return 0;
}
```

CF148E Porcelain

可以先预处理, 求出来每一行分别选 1个, 2个, 3个, ... 时能获得的最大价值

接下来, 就是在每一行挑几个, 问正好选够 m 个时, 能获得的最大价值是多少, 问题转化为了一个分组背包的板子题

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

using namespace std;

const int N = 100 + 5, M = 10000 + 5;
int len[N], w[N], pre[N], c[N][N], f[M];

int get_sum(int l, int r) { return pre[r] - pre[l-1]; }

void solve(int id) {
  cin >> len[id];
  for (int i = 1; i <= len[id]; ++i) cin >> w[i], pre[i] = pre[i-1]+w[i];

  for (int i = 1; i <= len[id]; ++i) {
    for (int j = 0, k = i; j <= i; ++j, --k) {
      c[id][i] = max(c[id][i], pre[j] + get_sum(len[id]-k+1,len[id]));
    }
  }
}

int main()
{
  int n, m; cin >> n >> m;
  for (int i = 1; i <= n; ++i) solve(i);

  for (int i = 1; i <= n; ++i) {
    for (int j = m; j >= 0; --j) {
      for (int k = 1; k <= len[i]; ++k) {
        if (j-k >= 0) f[j] = max(f[j], f[j-k]+c[i][k]);
      }
    }
  }
  cout << f[m] << endl;
  return 0;
}
```

Gym - 103428G Shinyruo and KFC

最多只有 sqrt(n) 种不同的餐品数量, 每种餐品可以 O(k) 来做

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

using namespace std;

typedef long long LL;
const int maxn = 1e5 + 5;
const int mod = 998244353;
int w[maxn], fac[maxn], inv_fac[maxn], c[maxn];

int qmod(int a, int k) {
  int res = 1;
  while (k) {
    if (k&1) res = (LL)res*a % mod;
    a = (LL)a*a % mod;
    k >>= 1;
  }
  return res;
}

int inv(int x) { return qmod(x, mod-2); }

int C(int n, int m) {
  if (n < m) return 0;
  return (LL)fac[n]*inv_fac[m]%mod*inv_fac[n-m]%mod;
}

struct node {
  int val, cnt;
};

int main()
{
  fac[0] = 1;
  for (int i = 1; i < maxn; ++i) fac[i] = (LL)fac[i-1]*i % mod;
  inv_fac[maxn-1] = inv(fac[maxn-1]);
  for (int i = maxn-2; i >= 0; --i) inv_fac[i] = (LL)inv_fac[i+1]*(i+1) % mod;

  int n, m; cin >> n >> m;
  for (int i = 1; i <= n; ++i) cin >> w[i], c[w[i]]++;

  vector<node> vec;
  for (int i = 1; i < maxn; ++i) {
    if (c[i]) vec.push_back({i, c[i]});
  }

  for (int k = 1; k <= m; ++k) {
    int res = 1;
    for (auto it : vec) {
```

```
      int val = it.val, cnt = it.cnt;
      res = (LL)res * qmod(C(k,val), cnt) % mod;
    }
    cout << res << "\n";
  }
  return 0;
}
```

CF2026D Sums of Segments

二分 + 前缀和, 先预处理出每一段的和以及每一段的长度, 后面用 二分查找 定位到第 x 个数在哪一段, 然后可以 O(1) 求前面的和

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

using namespace std;

typedef long long LL;
const int maxn = 3e5 + 5;
int w[maxn], len[maxn];
LL pre_w[maxn], pre_len[maxn];
LL cw[maxn], pre_cw[maxn];
LL seg[maxn], pre_seg[maxn];
int n;

LL get_sum(int l, int r, LL a[]) { return (l<=r ? a[r]-a[l-1] : 0); }

LL calc(LL x) {
  if (!x) return 0;

  int k = lower_bound(pre_len+1, pre_len+n+1, x) - pre_len;
  LL res = pre_seg[k-1];

  int nums = x - pre_len[k-1];
  res += get_sum(k, k+nums-1, pre_cw);
  res -= get_sum(k, k+nums-1, pre_w)*(n-k+1-nums);

  return res;
}

int main()
{
  cin >> n;
  for (int i = 1; i <= n; ++i) {
    cin >> w[i], pre_w[i] = pre_w[i-1] + w[i];
    cw[i] = (LL)w[i]*(n-i+1), pre_cw[i] = pre_cw[i-1] + cw[i];
    len[i] = n-i+1, pre_len[i] = pre_len[i-1] + len[i];
  }
  for (int i = 1; i <= n; ++i) {
    seg[i] = get_sum(i, n, pre_cw), pre_seg[i] = pre_seg[i-1] + seg[i];
  }
```

```
  int m; cin >> m;
  while (m -- ) {
    LL l, r; cin >> l >> r;
    cout << calc(r) - calc(l-1) << "\n";
  }
  return 0;
}
```