综合混练

人员

褚锦轩、白芸珲、王毅博、司云心、周治润 到课, 王承周 线上

上周作业检查

https://www.luogu.com.cn/contest/242814

2025-0420六队上课(综合混练)

报名 编辑比赛

题目数 4 | 报名人数 22

比赛说明 题目列表 排行榜

名次	参赛者	总分	A	B	C	D
#1	袁晨峻	400 (3.81s)	100 (84ms)	100 (41ms)	100 (3.55s)	100 (131ms)
#2	阮文璋	400 (5.87s)	100 (98ms)	100 (88ms)	100 (5.57s)	100 (116ms)
#3	杨俊彦	400 (6.53h)	100 (71ms)	100 (120ms)	100 (2.57h)	100 (3.96h)
#4	刘奕辰	300 (231ms)	100 (96ms)	100 (57ms)		100 (78ms)
#5	陈欣妙	300 (275ms)	100 (107ms)	100 (83ms)	0 (0ms)	100 (85ms)
#6	杨咏丞	300 (5.63s)	100 (96ms)	100 (86ms)	100 (5.45s)	0 (0ms)
#7	王博涵	300 (6.15s)	100 (91ms)	100 (79ms)	100 (5.98s)	
#8	徐思远	300 (4.45h)	100 (97ms)	100 (47ms)	0 (0ms)	100 (4.45h)
#9	李雨谦	212 (166ms)	100 (96ms)	100 (70ms)	12 (0ms)	
#10	曹塬	200 (139ms)	100 (97ms)	100 (42ms)		
#11	周治润	200 (144ms)	100 (86ms)	100 (58ms)		
#12	董昱含	200 (155ms)	100 (95ms)	100 (60ms)		
#13	褚锦轩	200 (159ms)	100 (72ms)	100 (87ms)		
#14	潘俊伊	200 (162ms)	100 (90ms)	100 (72ms)		
#15	龙沛轩	200 (165ms)	100 (96ms)	100 (69ms)		
#16	SSJ司云心	200 (165ms)	100 (85ms)	100 (80ms)	0 (0ms)	
#17	白芸珲	200 (200ms)	100 (78ms)	100 (122ms)	0 (0ms)	
#18	王承周	200 (333ms)	100 (103ms)	100 (230ms)	0 (0ms)	
#19	王毅博	200 (4.42s)	100 (70ms)	0 (0ms)	100 (4.35s)	
#20	李锦澍	190 (92ms)	100 (92ms)	90 (0ms)		
#21	许睿谦	100 (64ms)	100 (64ms)	0 (0ms)		
#22	王陆文龙	100 (104ms)	100 (104ms)	0 (0ms)		

作业

https://www.luogu.com.cn/contest/244893 (课上讲了 A ~ C 题, 课后作业是 D 题)

课堂表现

今天的 C 题同学们课上整体做的不是太好, 涉及到记忆化搜索的内容, 同学们遗忘比较多, 课下要好好复习一下 C 题。

课堂内容

P4086 [USACO17DEC] My Cow Ate My Homework S

题目本意是让求: 2~n, 3~n, 4~n, ..., n-1~n 区间中, 在每个区间都去掉一个最低分的情况下, 哪种情况下的区间平均值最大

因此, 可以 O(n) 维护一个 suf[i] 的后缀和数组 和一个 suf_min[i] 的后缀最小值数组

- suf[i] 代表: 区间 i~n 的区间和
- suf_min[i] 代表: 区间 i~n 的最小值

那么区间 i~n 在去掉一个最低分时区间的平均值是: (suf[i] - suf_min[i]) / (n-i) -> 可以 O(1) 求

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

using namespace std;

const int maxn = 1e5 + 5;
int w[maxn];
int suf_sum[maxn], suf_min[maxn];

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> w[i];
  suf_min[n+1] = 10000 + 5;
  for (int i = n; i >= 1; --i) {
    suf_sum[i] = suf_sum[i+1] + w[i];
    suf_min[i] = min(suf_min[i+1], w[i]);
  }

  double maxx_avg = -1.0;
  for (int k = 1; k <= n-2; ++k) {
    double t = 1.0*(suf_sum[k+1]-suf_min[k+1]) / (n-k-1);
    maxx_avg = max(maxx_avg, t);
  }

  for (int k = 1; k <= n-2; ++k) {
    double t = 1.0*(suf_sum[k+1]-suf_min[k+1]) / (n-k-1);
    if (t == maxx_avg) cout << k << endl;
  }
  return 0;
}
```

P1141 01迷宫

求每个联通块的大小即可

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

```cpp
using namespace std;

const int maxn = 1000 + 5;
char s[maxn][maxn];
bool st[maxn][maxn];
int f[maxn][maxn];
int n;

struct node {
  int x, y;
};
vector<node> vec;
int dx[] = {-1, 1, 0, 0}, dy[] = {0, 0, -1, 1};

void dfs(int x, int y) {
  vec.push_back({x, y}), st[x][y] = true;
  for (int i = 0; i < 4; ++i) {
    int nx = x+dx[i], ny = y+dy[i];
    if (nx>=1 && nx<=n && ny>=1 && ny<=n && s[nx][ny]!=s[x][y] && !st[nx][ny])
dfs(nx, ny);
  }
}

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

  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n; ++j) {
      if (!st[i][j]) {
        vec.clear();
        dfs(i, j);
        for (node it : vec) f[it.x][it.y] = (int)vec.size();
      }
    }
  }

  while (m -- ) {
    int x, y; cin >> x >> y;
    cout << f[x][y] << endl;
  }
  return 0;
}
```

U552391 三角形

对数组进行排序, 然后 O(n^2) 枚举前两条三角形的边 w[i] 和 w[j]

那么第三条边一定要求 大于等于w[j] 并且 小于w[i]+w[j]

此时可以用二分查找, 查找数组中第一个 大于w[i]+w[j] 的数在哪里

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

using namespace std;

typedef long long LL;
const int maxn = 2e3 + 5;
int w[maxn];

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

  LL res = 0;
  for (int i = 1; i <= n; ++i) {
    for (int j = i+1; j <= n; ++j) {
      int limit = w[i] + w[j];
      int k = lower_bound(w+j+1, w+n+1, limit) - w - 1;
      res += k - j;
    }
  }
  cout << res << endl;
  return 0;
}
```

P7995 [USACO21DEC] Walking Home B

维护一个 dfs, 参数为 x,y,id,k 4 个参数, 代表目前在点 (x,y), 沿着 id 方向走, 最多转 k 次时, 到终点有多少种不同的方案

直接搜会超时, 所以把每次搜到的结果可以用 记忆化 保存下来, 这样 时间复杂度 和 空间复杂度 就都是 50 * 50 * 2 * 4

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

using namespace std;

const int maxn = 50 + 5;
char s[maxn][maxn];
int f[maxn][maxn][2][5];
int n;
int dx[] = {0,1}, dy[] = {1,0};

int dfs(int x, int y, int id, int k) {  // (x,y), 沿着 id 方向, 最多转 k 次
  if (k < 0) return 0;
  if (s[x][y] == 'H') return 0;
  if (x==n && y==n) return 1;
  if (f[x][y][id][k] != -1) return f[x][y][id][k];
```

```
  int nx1 = x+dx[id], ny1 = y+dy[id];
  int nx2 = x+dx[id^1], ny2 = y+dy[id^1];

  int res = 0;
  if (nx1>=1 && nx1<=n && ny1>=1 && ny1<=n) res += dfs(nx1, ny1, id, k);
  if (nx2>=1 && nx2<=n && ny2>=1 && ny2<=n && k>0 && (x!=1||y!=1)) res += dfs(nx2,
ny2, id^1, k-1);
  f[x][y][id][k] = res;
  return f[x][y][id][k];
}

void solve() {
  memset(f, -1, sizeof(f));

  int k; cin >> n >> k;
  for (int i = 1; i <= n; ++i) cin >> (s[i]+1);

//  cout << "---------------------- ";
  cout << dfs(1,1,0,k) + dfs(1,1,1,k) << endl;
}

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

P3131 [USACO16JAN] Subsequences Summing to Sevens S

跟之前的 k 倍区间是很像的一个题

首先维护前缀和, 然后把所有前缀和 %7

要找最长的 区间和是7的倍数 的区间, 就是找:

前缀和为 0 的最靠左、右的位置

前缀和为 1 的最靠左、右的位置

...

前缀和为 6 的最靠左、右的位置

找出一个最长的即可

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

using namespace std;

typedef long long LL;
const int maxn = 5e4 + 5;
```

```cpp
int w[maxn];
LL pre[maxn];
int p[10], s[10];

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> w[i];
  for (int i = 1; i <= n; ++i) {
    pre[i] = pre[i-1] + w[i]; pre[i] %= 7;
  }

  for (int i = 0; i <= 6; ++i) p[i] = s[i] = -1;
  for (int i = 0; i <= n; ++i) {
    int t = pre[i];
    if (p[t] == -1) p[t] = i;
  }
  for (int i = n; i >= 0; --i) {
    int t = pre[i];
    if (s[t] == -1) s[t] = i;
  }

  int res = 0;
  for (int i = 0; i <= 6; ++i) {
    int l = p[i], r = s[i];
    if (l==-1 || r==-1) continue;
    res = max(res, r-l);
  }
  cout << res << endl;
  return 0;
}
```