综合混练

人员

褚锦轩、许睿谦、王毅博、阮文璋、王承周、司云心、曹塬 到课

上周作业检查

https://www.luogu.com.cn/contest/241021

2025-0413六队上课(综合混练)

报名 编辑比赛

题目数 5 | 报名人数 24

比赛说明 题目列表 排行榜

名次	参赛者	总分	A	B	C	D	E
#1	李锦澍	415 (2.65s)	100 (155ms)	100 (36ms)	100 (65ms)	100 (2.38s)	15 (19ms)
#2	徐思远	415 (2.27h)	100 (149ms)	100 (36ms)	100 (66ms)	100 (818ms)	15 (2.27h)
#3	杨俊彦	415 (5.02h)	100 (158ms)	100 (36ms)	100 (64ms)	100 (2.22h)	15 (2.80h)
#4	阮文璋	415 (23.26h)	100 (150ms)	100 (39ms)	100 (65ms)	100 (13.81h)	15 (9.45h)
#5	周治润	415 (11.04d)	100 (148ms)	100 (37ms)	100 (74ms)	100 (5.53d)	15 (5.51d)
#6	袁晨峻	415 (13.27d)	100 (1.99h)	100 (36ms)	100 (57ms)	100 (6.60d)	15 (6.59d)
#7	王陆文龙	400 (6.99d)	100 (150ms)	100 (38ms)	100 (65ms)	100 (6.99d)	
#8	王毅博	400 (11.97d)	100 (74ms)	100 (38ms)	100 (5.47d)	100 (6.50d)	
#9	龙沛轩	340 (19.49d)	100 (6.50d)	100 (36ms)	100 (6.49d)	40 (6.50d)	
#10	王承周	315 (4.43d)	100 (156ms)	100 (40ms)	100 (66ms)		15 (4.43d)
#11	SSJ司云心	315 (11.91d)	100 (160ms)	100 (43ms)	100 (5.47d)		15 (6.44d)
#12	刘奕辰	315 (22.39d)	100 (4.49d)	100 (4.49d)	100 (6.45d)	0	15 (6.96d)
#13	许睿谦	300 (172ms)	100 (73ms)	100 (37ms)	100 (62ms)		
#14	韩鸣蔚	300 (244ms)	100 (142ms)	100 (36ms)	100 (66ms)		
#15	陈欣妙	300 (259ms)	100 (153ms)	100 (38ms)	100 (68ms)		
#16	李雨谦	300 (259ms)	100 (152ms)	100 (41ms)	100 (66ms)		
#17	杨咏丞	300 (1.91d)	100 (151ms)	100 (1.41d)	100 (11.91h)		
#18	褚锦轩	300 (6.47d)	100 (154ms)	100 (43ms)	100 (6.47d)		
#19	董昱含	300 (7.35d)	100 (2.45d)	100 (2.43d)	100 (2.48d)		
#20	曹原	300 (7.49d)	100 (148ms)	100 (1.45d)	100 (6.05d)		
#21	潘俊伊	265 (7.00d)	40 (0ms)	100 (38ms)	100 (65ms)	10 (0ms)	15 (7.00d)
#22	王博涵	144 (147ms)	100 (147ms)	44 (0ms)			
#23	白芸珲	130 (6.24d)	30 (0ms)	100 (6.24d)			

作业

https://www.luogu.com.cn/contest/242814 (课上讲了 A ~ C 题, 课后作业是 D 题)

课堂表现

今天课上主要带同学们复习了一些 dfs 和 bfs 的内容, 同学们对搜索不熟的, 课下一定要把这几道题目多写几遍。

课堂内容

B3799 [NICA #1] 序列

每次 1 操作, 要把所有 n 个数全部 +x 或者全部 -x, 太慢了 -> 所以可以维护一个 delta, 把每次的改变量维护在 delta 信息上

即: 每次让 delta += x 即可 (这样, 每次的 1 操作就可以 O(1) 完成了)

对于 2 操作, 要找最大的子序列和, 其实就是把数组里面所有的 >=0 的数进行求和即可

已知, 所有数在原本数组的基础上都加了个 delta, 那么其实就是找到原数组中 >=-delta 的数, 这些数就是目前 >=0 的数

这里可以先把原数组进行 sort 从小到大排序, 然后就可以二分查找 logn 的找到第一个 >=-delta 的数了

最终的子序列和就比较好求了: 就是 上面说的数 到 第n个数的 区间和, 再加上 delta*cnt 即可 (cnt代表这一段有多少数)

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

using namespace std;

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

LL get_sum(int l, int r) { return (l>r ? 0 : p[r]-p[l-1]); }

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

  LL delta = 0;
  while (m -- ) {
    int op; cin >> op;
    if (op == 1) {
      int k; cin >> k; delta += k;
    } else {
      // 在 w 数组中找到所有 >=-delta 的数
      int pos = lower_bound(w+1, w+n+1, -delta) - w;
      int cnt = n-pos+1; LL sum = get_sum(pos, n);
      cout << sum + (LL)delta*cnt << endl;
    }
  }
  return 0;
}
```

P10294 [CCC 2024 J5] Harvest Waterloo

直接开 n*m 的数组开不下, 可以用 vector 数组

剩下的就是 dfs 搜索即可

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

using namespace std;

const int maxn = 1e5 + 5;
vector<char> vec[maxn];
char s[maxn];
int dx[] = {-1, 1, 0, 0}, dy[] = {0, 0, -1, 1};
struct node {
  int x, y;
};
vector<bool> f[maxn];

int get(int x, int y) {
  char c = vec[x][y];
  if (c == 'S') return 1;
  if (c == 'M') return 5;
  if (c == 'L') return 10;
  return 0;
}

int main()
{
    int n, m; cin >> n >> m;
    for (int i = 0; i <= n+1; ++i) {
    for (int j = 0; j <= m+1; ++j) vec[i].push_back('0'), f[i].push_back(false);
    }

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

    int sx, sy; cin >> sx >> sy; ++sx, ++sy;
    int res = 0;
    queue<node> q; q.push({sx, sy}); res += get(sx, sy); f[sx][sy] = true;
    while (!q.empty()) {
    node u = q.front(); q.pop();
    int x = u.x, y = u.y;
    for (int i = 0; i < 4; ++i) {
      int nx = x+dx[i], ny = y+dy[i];
      if (nx>=1 && nx<=n && ny>=1 && ny<=m && vec[nx][ny]!='*' && f[nx]
[ny]==false) {
        q.push({nx, ny}); res += get(nx, ny); f[nx][ny] = true;
      }
    }
    }
    cout << res << endl;
```

```
  return 0;
}
```

B4286 [蓝桥杯青少年组省赛 2022] 农作物

dfs 找连通块数量, 搜过的点打上标记即可。

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

using namespace std;

const int maxn = 500 + 5;
char s[maxn][maxn];
bool st[maxn][maxn];
int n, m;
int dx[] = {-1, 1, 0, 0}, dy[] = {0, 0, -1, 1};

void dfs(int x, int 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<=m && s[nx][ny]=='R' && !st[nx][ny]) dfs(nx,
ny);
  }
}

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

  int res = 0;
  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= m; ++j) {
      if (s[i][j]=='R' && !st[i][j]) dfs(i,j), ++res;
    }
  }
  cout << res << endl;
  return 0;
}
```

P1588 [USACO07OPEN] Catch That Cow S

可以把原问题转化成一个 bfs 问题, 认为 x 和 x-1, x+1, 2*x 连边

只需要考虑 1 ~ 2e5 之间的数, 跑 起点 到 终点 之间的 bfs

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

```
using namespace std;

const int maxn = 2e5 + 5;
int f[maxn];

void solve() {
  memset(f, -1, sizeof(f));
  int st, ed; cin >> st >> ed;
  queue<int>q; q.push(st); f[st] = 0;
  while (!q.empty()) {
    int u = q.front(); q.pop();
    if (u-1>=0 && f[u-1]==-1) q.push(u-1), f[u-1] = f[u]+1;
    if (u+1<maxn && f[u+1]==-1) q.push(u+1), f[u+1] = f[u]+1;
    if (u*2<maxn && f[u*2]==-1) q.push(u*2), f[u*2] = f[u]+1;
  }
  cout << f[ed] << endl;
}

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

U537157 souvenir

从每个点开始, 做一遍 bfs, bfs 的过程中, 既维护最短路, 又维护最大值

如果是第一次从 u 到 i, 那么到 i 的最大值 f[i] = f[u] + a[i]

如果不是第一次从 u 到 i, 只要目前的长度 == 最短路的长度, 那么 f[i] 就要跟 f[u]+a[i] 取最大值

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

using namespace std;

typedef long long LL;
const int maxn = 300 + 5;
int w[maxn];
int dis[maxn][maxn]; LL f[maxn][maxn];
char s[maxn][maxn];
int n;

void bfs(int x) {
  queue<int> q; q.push(x); dis[x][x] = 0; f[x][x] = w[x];
  while (!q.empty()) {
    int u = q.front(); q.pop();
    for (int i = 1; i <= n; ++i) {
      if (s[u][i]=='Y' && dis[x][i]==-1) {
        q.push(i); dis[x][i] = dis[x][u]+1; f[x][i] = f[x][u]+w[i];
```

```
      } else if (s[u][i]=='Y' && dis[x][u]+1==dis[x][i]) {
        f[x][i] = max(f[x][i], f[x][u]+w[i]);
      }
    }
  }
}

int main()
{
  memset(dis, -1, sizeof(dis)); memset(f, 0, sizeof(f));

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

  for (int i = 1; i <= n; ++i) bfs(i);

  int m; cin >> m;
  while (m -- ) {
    int u, v; cin >> u >> v;
    if (dis[u][v] == -1) cout << "Impossible" << endl;
    else cout << dis[u][v] << " " << f[u][v] << endl;
  }
  return 0;
}
```

bfs 中维护特定信息

在最短路的前提下，路程上的总价值最大

```
int f[maxn];
f[i]: 起点 -> i 最大价值

dis[maxn];
queue<int> q; q.push(st); dis[st]=0; f[st]=a[st];
while (!q.empty()) {
    int u = q.front(); q.pop();
    for (int i : u) {
        if (dis[i] == -1) {  // 第一次到 i 这个点
            q.push(i); dis[i] = dis[u]+1; f[i] = f[u]+a[i];
        } else if (dis[u]+1 == dis[i]) {
            f[i] = max(f[i], f[u]+a[i]);
        }
    }
}
```

求最短路的条数

```
int f[maxn];
f[i]: 起点 -> i 有多少条最短路
```

```
dis[maxn];
queue<int> q; q.push(st); dis[st]=0; f[st]=1;
while (!q.empty()) {
    int u = q.front(); q.pop();
    for (int i : u) {
        if (dis[i] == -1) {  // 第一次到 i 这个点   (u->i)
            q.push(i); dis[i] = dis[u]+1; f[i] = f[u];
        } else if (dis[u]+1 == dis[i]) {
            f[i] += f[u];
        }
    }
}
```