离散化

人员

王承周、董昱含、褚锦轩、司云心、王毅博、曹塬 到课

上周作业检查

https://cppoj.kids123code.com/contest/134

2025-0629 周日13:30 (综合练习)

刷新

#	用户名	姓名	总分	选择分	编程分	时间	A	B	C	D	E
1	ruanwenzhang	阮文璋	500	0	500	591	100	100	100	100	100
2	wangchengzhou	王承周	500	0	500	679	100	100	100	100	100
3	dongyuhan	董昱含	500	0	500	726	100	100	100	100	100
4	chujinxuan	褚锦轩	375	0	375	2143	75	100	100		100
5	siyunxin	司云心	210	0	210	295	100	100	10		

作业

https://cppoj.kids123code.com/contest/179 (课上讲了 A ~ D 题, 课后作业是 E 题)

课堂表现

同学们课上做题做的整体还不错, 离散化 这个思想比较重要, 同学们课下可以再掌握掌握。

课堂内容

[蓝桥杯 2025 省 A/Python B 第二场] 消消乐

希望最终剩下的字符尽量多, 说明要尽量少的消掉原本字符串中的 A 和 B, 使得最终的字符串中不存在 前A后B 的情况

可以参考 ABAB 这个例子, 我们可以消掉第一个 A 和 最后一个 B, 就只剩 BA, 不需要再消除了

那么, 就是希望尽量消掉靠前的 A 和靠后的 B

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

using namespace std;

const int maxn = 1e6 + 5;
char s[maxn];

int main()
{
  cin >> (s+1);
```

```
  int n = strlen(s+1);
  int i = 1, j = n;
  int res = n;
  while (i < j) {
    while (i<j && s[i]=='B') ++i;
    while (i<j && s[j]=='A') --j;
    if (i < j) res -= 2, ++i, --j;
    else break;
  }
  cout << res << endl;
  return 0;
}
```

区间和

把 输入的 n 个位置 和 查询的 m 次的 l 和 r 统一存起来, 映射成 1 ~ 3e5 之间的整数

因为这些整数我们只需要关心他们的相对大小关系即可

然后, 维护前缀和每次查询区间和即可

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

using namespace std;

typedef long long LL;
const int maxn = 3e5 + 5;
int x[maxn], c[maxn];
int l[maxn], r[maxn];
LL w[maxn], p[maxn];

vector<int> ys;
int fFind(int x) {
    return lower_bound(ys.begin(), ys.end(), x) - ys.begin() + 1;
}

int main()
{
    int n, m; cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> x[i] >> c[i], ys.push_back(x[i]);
    for (int i = 1; i <= m; ++i) {
        cin >> l[i] >> r[i]; ys.push_back(l[i]), ys.push_back(r[i]);
    }

    sort(ys.begin(), ys.end());
    ys.erase(unique(ys.begin(), ys.end()), ys.end());

    for (int i = 1; i <= n; ++i) {
        int pos = fFind(x[i]); w[pos] += c[i];
    }
    for (int i = 1; i < maxn; ++i) p[i] = p[i-1] + w[i];
```

```
    for (int i = 1; i <= m; ++i) {
        int ll = fFind(l[i]), rr = fFind(r[i]);
        cout << p[rr] - p[ll-1] << endl;
    }
    return 0;
}
```

最长上升子序列

n^2 做法: 维护一个 f 数组, 其中 f[i] 代表以 i 结尾时, 最长上升子序列是多长

f[i] 可以从 1~i-1 中, 所有比 a[i] 小的地方, 挑一个最大的转移过来

```
// n^2 做法
#include <bits/stdc++.h>

using namespace std;

const int maxn = 5000 + 5;
int w[maxn], f[maxn];

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

    for (int i = 1; i <= n; ++i) {
        for (int j = 0; j < i; ++j) {
            if (w[j] < w[i]) f[i] = max(f[i], f[j]+1);
        }
    }

    int res = 0;
    for (int i = 1; i <= n; ++i) res = max(res, f[i]);
    cout << res << endl;
    return 0;
}
```

nlogn 做法: 用一个 vector 进行存储, 其中 vector 第 i 个位置存储的值为, 当最长上升子序列长度为 i 时, 最小的结尾的值是多少

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

using namespace std;

const int maxn = 1e6 + 5;
```

```
int main()
{
  int n; cin >> n;
  vector<int> vec;
  while (n -- ) {
    int x; cin >> x;
    if (vec.empty() || x>vec.back()) vec.push_back(x);
    else *lower_bound(vec.begin(), vec.end(),x) = x;
  }

  cout << vec.size() << endl;
  return 0;
}
```

红牌

dp, 跟之前同学们做过的 数字三角形 那道题很像, 每次只能往下或者往右走, 求路径上的最小值

定义 f[i][j], 代表从前面一路走到 (i,j) 这个格子, 路径上值全部和 的最小值是多少

这个题多了一个循环的思想, 因此可以在原数组后面再拷贝一份

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

using namespace std;

const int maxn = 4000 + 5;
const int inf = 0x3f3f3f3f;
int w[maxn][maxn], f[maxn][maxn];

int main()
{
  int n, m; cin >> n >> m;
  for (int i = 1; i <= m; ++i) {
    for (int j = 1; j <= n; ++j) cin >> w[i][j];
  }
  for (int i = m+1; i <= 2*m; ++i) {
    for (int j = 1; j <= n; ++j) w[i][j] = w[i-m][j];
  }

  memset(f, 0x3f, sizeof(f));
  for (int i = 0; i < maxn; ++i) f[i][0] = 0;

  for (int i = 1; i <= 2*m; ++i) {
    for (int j = 1; j <= n; ++j) f[i][j] = min(f[i][j-1], f[i-1][j-1]) + w[i][j];
  }

  int res = 2e9 + 5;
  for (int i = 1; i <= 2*m; ++i) res = min(res, f[i][n]);
  cout << res << endl;
```

```
  return 0;
}
```

公园

二分 + 二维前缀和

二分, 判断是否存在有一个 K * K 的区间的中位数超过 mid

可以把整个数组中所有 >=mid 的值定为 1, 把 <mid 的值定为 0

然后就是判断是否有一个 K * K 的矩阵的矩阵和超过 (K * K) / 2 + 1 即可

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

using namespace std;

const int maxn = 800 + 5;
int w[maxn][maxn], f[maxn][maxn];

int get_sum(int x1, int y1, int x2, int y2) {
  return f[x2][y2] - f[x1-1][y2] - f[x2][y1-1] + f[x1-1][y1-1];
}

bool check(int n, int k, int mid) {
  memset(f, 0, sizeof(f));
  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n; ++j) {
      f[i][j] = (w[i][j]<=mid ? 1 : 0);
      f[i][j] += f[i-1][j] + f[i][j-1] - f[i-1][j-1];
      if (i>=k && j>=k && get_sum(i-k+1, j-k+1, i, j)>=(k*k+1)/2) return true;
    }
  }
  return false;
}

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

  int l = 0, r = 1e9;
  while (l <= r) {
    int mid = (l + r) / 2;
    if (check(n, k, mid)) r = mid-1;
    else l = mid+1;
  }
  cout << l << endl;
```

```
    return 0;
}
```