综合练习

人员

王承周、董昱含、褚锦轩 到课, 司云心、阮文璋 线上

上周作业检查

https://cppoj.kids123code.com/contest/107

2025-0622 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
1	wangchengzhou	王承周	300	0	300	519	100	100		100
2	siyunxin	司云心	300	0	300	1303	100	100		100
3	dongyuhan	董昱含	300	0	300	1916	100	100		100
4	chujinxuan	褚锦轩	200	0	200	508	100	100		
5	chenyiran	陈奕然	200	0	200	566	100	100		
6	yuxiaolong	于霄龙	100	0	100	86	100			
7	caoyuan	曹塬	100	0	100	179	100			
8	xuruiqian	许睿谦	100	0	100	511	0	100		

作业

https://cppoj.kids123code.com/contest/134 (课上讲了 A ~ D 题, 课后作业是 E 题)

课堂表现

今天的 C 题稍微复杂一些, 同学们课上基本都没有做出来, 课下要好好补一下这道题。

课堂内容

[常州市赛 2024] 盒子

把所有盒子从小到大排序, 然后从前往后双指针扫, 把小盒子往中盒子里放, 直到不能放为止

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

using namespace std;

typedef long long LL;
const int maxn = 1e5 + 5;
LL w[maxn], f[maxn];

int main()
{
  int n; cin >> n;
```

```
  for (int i = 1; i <= n; ++i) cin >> w[i];
  sort(w+1, w+n+1);
  for (int i = 1; i <= n; ++i) f[i] = w[i] / 2;

  for (int l = 1, r = 2; l <= n; ++l) {
    while (r<=n && f[r]<w[l]) ++r;
    if (r > n) {
      cout << n - l + 1 << endl;
      return 0;
    }
    f[r] -= w[l];
  }
  return 0;
}
```

[蓝桥杯 2023 省 Python B] 松散子序列

简单线性dp, 设 f[i] 代表以 i 结尾时能获得的最大价值是多少, f[i] 可以从 f[i-2] 和 f[i-3] 中更大的那个转移过来

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

using namespace std;

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

int main()
{
  cin >> (s+1);
  int n = strlen(s+1);
  for (int i = 1; i <= n; ++i) {
    if (i <= 2) f[i] = s[i]-'a'+1;
    else f[i] = max(f[i-2], f[i-3]) + s[i]-'a'+1;
  }
  cout << max(f[n-1], f[n]) << endl;
  return 0;
}
```

最长括号匹配

线性 dp, 这个题会比较难想一些

设 f[i] 是以 i 结尾时, 往前最长的一段合法的括号串的长度

1. 当 s[i]=='(' 或 s[i]=='[' 时, f[i] 是 0
2. 当 s[i]==')' 且 s[i-f[i-1]-1]=='(' 时, 或者 s[i]==']' 且 s[i-f[i-1]-1]=='[' 时, f[i] 可以从前面转移过来

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

using namespace std;

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

int main()
{
  cin >> (s+1);
  int n = strlen(s+1);

  int pos = 0;
  for (int i = 1; i <= n; ++i) {
    if (s[i]=='(' || s[i]=='[') continue;

    int len = f[i-1];
    int p = i - len - 1;
    if (s[i] == ')') {
      if (s[p] == '(') f[i] = len+2+f[p-1];
    } else {
      if (s[p] == '[') f[i] = len+2+f[p-1];
    }

    if (f[i] > f[pos]) pos = i;
  }

  int l = pos - f[pos] + 1, r = pos;
  for (int i = l; i <= r; ++i) cout << s[i];
  cout << endl;
  return 0;
}
```

[USACO16OPEN] Diamond Collector S

维护 ml 和 mr 两个数组, ml[i] 代表以 i 结尾时, 往左找最远能找到哪个位置; mr[i] 代表以 i 开头时, 往右找最远能找到哪个位置, 这个过程可以用 二分查找 来做

然后枚举一个分界点 i, 此时可以以 i 作为分界线, 在左边找一段长的, 在右边找一段长度

此时, 就是看以 1~i 结尾中, 哪一段最长; 还有以 i+1~n开头, 哪一段最长

这里可以用一个 前缀最大值 和 后缀最大值 数组维护, 就可以 O(1) 求了

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

using namespace std;

const int maxn = 50000 + 5;
```

```
int w[maxn], mr[maxn], ml[maxn];
int pmax[maxn], smax[maxn];

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

  for (int i = 1; i <= n; ++i) {
    int r = upper_bound(w+1, w+n+1, w[i]+k) - w - 1;
    int l = lower_bound(w+1, w+n+1, w[i]-k) - w;
    r = min(r, n), l = max(l, 1);
    mr[i] = r, ml[i] = l;
  }

  for (int i = 1; i <= n; ++i) pmax[i] = max(pmax[i], i - ml[i] + 1);
  for (int i = n; i >= 1; --i) smax[i] = max(smax[i+1], mr[i] - i + 1);

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

[蓝桥杯 2013 国 C] 危险系数

枚举每一个点, 考虑把这个点去掉, 能否使得 1 号点和 n 号点联通

O(n) 枚举, O(n) 进行判断, 时间复杂度为 O(n^2) 级别

判断两个点是否联通, 可以用搜索, 也可以用并查集, 并查集会更简单一些

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

using namespace std;

const int maxn = 2000 + 5;
struct node {
  int u, v;
} w[maxn];

int f[maxn];
int fFind(int x) {
  if (f[x] != x) f[x] = fFind(f[x]);
  return f[x];
}

int n, m;
int st, ed;
bool check(int x) {
```

```
  for (int i = 1; i <= n; ++i) f[i] = i;
  for (int i = 1; i <= m; ++i) {
    int u = w[i].u, v = w[i].v;
    if (u==x || v==x) continue;
    int fu = fFind(u), fv = fFind(v);
    if (fu != fv) f[fu] = fv;
  }
  return fFind(st) != fFind(ed);
}

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

  cin >> st >> ed;
  if (check(0)) { cout << -1 << endl; return 0; }

  int cnt = 0;
  for (int i = 1; i <= n; ++i) {
    if (i==st || i==ed) continue;
    if (check(i)) ++cnt;
  }
  cout << cnt << endl;
  return 0;
}
```