CITRC 期末競賽題解

- 基礎組： A, C, D, E, G, H, I, J
- 進階組： B, F

題目難度

$D << E < C = G < A < H < J < I < F << B$

A. 我愛CITRC

I love CITRC

- 考點： 基本輸入輸出、跳脫字元
- 難度： 3 / 10
- 首殺：
- 提交次數：

- XXXXX當然是CITRC
- `\` 要加跳脫字元
- 最後記得換行

```
cout << "Yes, I love CITRC so much.(\\\\\\>.<\\\\\\)\n"
```

第二行要輸出一個數字，可以是浮點數
既然麥克阿瑟都說了 "Don't be greedy..."
輸出 100 太貪心了肯定沒分
而你輸出多少分就可以拿到多少分
100 分的方法是輸出 $(99.0, 100.0)$ 區間的小數

Full Solution (100/100)

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    cout << "Yes, I love CITRC so much.(\\\\\\>.<\\\\\\)\n";
    cout << "99.7\n";
    return 0;
}
```

B. 小孤獨，不要化成灰啊！

- 考點： 樹 + dp
- 難度： 9 / 10
- 首殺：
- 提交次數：

Subtask 1. 結構是一條鏈

直接用一維陣列做就行，實作方法有很多
因為是鏈所以把涼的位置當作誰都不影響結果
實作上先把Y設成K或B就好

Subtask 2. $n \leq 2 \times 10^5$

若且唯若此結構由 n 個點組成，有 $n-1$ 條邊相連
兩點間必有且僅有一條路徑，則此結構是一棵樹

這題首先你要先看出是一棵樹
可以觀察出喜多(K)和小孤獨(B)是會互相影響的
而涼(Y)是獨立不受影響的，所以可以直接視為K或B
但是視為K或B會因為不同的子節點情況受到影響
因此在dfs的時候用動態規劃去找到每個點的情況

定義 $K[i], B[i], Y[i]$ 為以 i 為根節點所需最少的放置數
K, B, Y 是討論此節點是不同人的答案
而已知 K, B 會互相影響所以不能視為其他可能
而 Y 可以視情況當作 B 或 K ，因此就可以得到轉移式

定義集合 S_i 為 i 的所有子節點
$K[i] = \sum_{j \in S_i} min(K[j],\ B[j]+1,\ Y[j])$
$B[i] = \sum_{j \in S_i} min(K[j]+1,\ B[j],\ Y[j])$
$Y[i] = \sum_{j \in S_i} min(K[j]+1,\ B[j]+1,\ Y[j])$

Full Solution (100/100)

```
const int INF = 1e18;
void dfs(int cur, int from) {
    for(auto &nxt : g[cur]) {
        if(nxt == from) continue;
        dfs(nxt,cur);
        if(role[cur] == 'Y') {
            Kita[cur] += min({Kita[nxt], Bocchi[nxt]+1, Yamada[nxt]});
            Bocchi[cur] += min({Kita[nxt]+1, Bocchi[nxt], Yamada[nxt]});
            Yamada[cur] += min({Kita[nxt]+1, Bocchi[nxt]+1, Yamada[nxt]});
        }else if(role[cur] == 'B') {
```

```
            Kita[cur] = Yamada[cur] = INF;
            Bocchi[cur] += min({Kita[nxt]+1, Bocchi[nxt], Yamada[nxt]});
        }else {
            Bocchi[cur] = Yamada[cur] = INF;
            Kita[cur] += min({Kita[nxt], Bocchi[nxt]+1, Yamada[nxt]});
        }
    }
    if(role[cur] == 'K') Bocchi[cur] = Yamada[cur] = INF;
    if(role[cur] == 'B') Kita[cur] = Yamada[cur] = INF;
}
```

C. 嚴禁電神裝弱

HARC

- 考點： if statement
- 難度： 2 / 10
- 首殺：
- 提交次數：

就找出數字最大的就是最電的那個
然後就把數字最大的那個名字輸出就好
這題主要只是要考在比較三個數字的時候
不能寫 $a > b > c$ 這種寫法

Full Solution (100/100)

```
string nameA, nameB, nameC;
int a,b,c;
cin >> nameA >> a;
cin >> nameB >> b;
cin >> nameC >> c;
if(a > b && a > c) cout << nameA << '\n';
if(b > a && b > c) cout << nameB << '\n';
if(c > a && c > b) cout << nameC << '\n';
```

不覺得寫很多 if 很麻煩嗎

```
pair<int,string> a,b,c;
cin >> a.second >> a.first;
cin >> b.second >> b.first;
cin >> c.second >> c.first;
cout << max({a,b,c}).second << '\n';
```

D. 雖然我不是數學家

Mathematician

- 基本分考點： 整數四則運算
- 基本分難度： 0 / 10
- 首殺：
- 提交次數：

- Bonus考點： string, 陣列
- Bonus難度： 7 / 10
- 首殺：

這題根本就沒有難度
之前強調過很多次數字太大要開long long了
甚至你把之前題解的程式碼照著貼上就有 100 分了

Subtask 1. $0 \leq A, B \leq 10^6$

`int` 範圍是 $-2^{32} \leq int \leq 2^{32} - 1$
2^{32} 大概是 2×10^9

Subtask 2. $0 \leq A, B \leq 10^{15}$

把 `int` 改成 `long long` 就好了
範圍： $-2^{64} \leq longlong \leq 2^{64} - 1$
大概比 10^{18} 次方再大一點

Accepted Solution (100/100)

```cpp
long long a,b;
cin >> a >> b;
cout << a+b << '\n';
```

Bonus : $10^{24} \leq A, B \leq 10^{10^6}$

數字連 `long long` 都超過了，沒辦法紀錄
使用陣列去存每一個數字，同位數再相加進位就好

Bonus Solution(160/160)

```cpp
string a,b;
cin >> a >> b;
```

```
int A[1000005],B[1000005],C[1000005];
for(int i=a.size()-1;i>=0;i--) A[a.size()-i-1] = a[i]-'0';
for(int i=b.size()-1;i>=0;i--) B[b.size()-i-1] = b[i]-'0';
for(int i=0;i<1000005;i++) C[i] = A[i] + B[i];
for(int i=0;i<1000005-1;i++) {
    if(C[i] >= 10) {
        C[i+1] += C[i] / 10;
        C[i] %= 10;
    }
}
int start = 0;
for(int i=1000004;i>=0;i--) {
    if(C[i] != 0) {
        start = i;
        break;
    }
}
for(int i=start;i>=0;i--) cout << C[i];
```

E. 進擊的苔雞殿

Attack on Chicken

- 考點： 基本輸入輸出
- 難度： 1 / 10
- 首殺：
- 提交次數：

就跟 hello, world 沒兩樣
題目說要幹嘛就幹嘛，注意一下空格就好了

Full Solution (100/100)

```
string a,b;
cin >> a >> b;
cout << a << ":Devote your " << b << "!\n";
```

F. 關於我熬夜爆肝練題

成為IOI國手的那檔事

Stay Up

- 基本分考點： dp
- 基本分難度： 7.5 / 10

- 首殺：
- 提交次數：

- Bonus考點： 矩陣快速冪
- Bonus難度： 7.6 / 10
- 首殺：

很有趣的是這是2024TRML思考賽的題目
只是被我魔改了題目敘述

Subtask 1: $1 \leq n \leq 20$

數字這麼小就窮舉就好了啊
你甚至叫電腦幫你窮舉還不用自己窮舉

Subtask1 Solution(55/100)

```cpp
const int mod = 1e9+7;
int n, ans = 0;
cin >> n;
for(int status = 0; status < (1<<n); status++) {
    bool check = 1;
    for(int j=0;j<n-1;j++) {
        if((status>>j)&1 && (status>>(j+1))&1) {
            check = 0; break;
        }
    }
    if(check) ans += 1, ans %= mod;
}
cout << ans << '\n';
```

Subtask 2： $1 \leq n \leq 2 \times 10^5$

這種題目絕對是 dp，所以就開始想 dp

可以知道每天有兩種狀態，要或不要
所以轉移式也很簡單
定義 $ch[i]$ 為前 i 天中第 i 天要取的方法數
反之， $nch[i]$ 為前 i 天中第 i 天不取方法數

顯而易見，考慮前後天關係而已
$ch[i] = nch[i-1]$
$nch[i] = ch[i] + nch[i-1]$
實作的時候記得邊加邊 mod 以免溢位

Accepted Solution (100/100)

```
const int mod = 1e9+7;
int n;
cin >> n;
ch[1] = notch[1] = 1;
for(int i=2;i<=n;i++) {
    ch[i] = notch[i-1] % mod;
    notch[i] = (ch[i-1] + notch[i-1]) % mod;
}
cout << (ch[n] + notch[n]) % mod << '\n';
```

但是其實這題還有另一個思考方式
定義 $dp[i]$ 為前 i 天的方法數
方法數有第 i 天取和不取兩種
若第 i 天要取，則第 $i-1$ 必不取，為 $dp[i-2]$
若第 i 天不取，則第 $i-1$ 可以取，為 $dp[i-1]$
因此 $dp[i] = dp[i-1] + dp[i-2]$
這不就是費氏數列嗎ouob

Accepted Solution (100/100)

```
const int mod = 1e9+7;
int n;
cin >> n;
dp[1] = 2, dp[2] = 3;
for(int i=3;i<=n;i++) {
    dp[i] = (dp[i-1] + dp[i-2]) % mod;
}
cout << dp[n] % mod << '\n';
```

Bonus ： $10^9 \leq n \leq 10^{18}$

數字很大沒辦法用迴圈跑怎麼辦？

既然是費氏數列那就數學解？

$ans = \frac{1}{\sqrt{5}}[(\frac{1+\sqrt{5}}{2})^{n+2} - (\frac{1-\sqrt{5}}{2})^{n+2}]$

很明顯不可能
都說了是費氏數列
那就用矩陣快速冪ouob
$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n+2}$

Full Solution (160/160)

```
const int mod = 1e9+7;
struct matrix {int a = 0,b = 0,c = 0,d = 0;};
matrix cross(const matrix x,const matrix y){
    matrix res;
    res.a = (x.a*y.a+x.b*y.c)%mod;
    res.b = (x.b*y.a+x.d*y.b)%mod;
    res.c = (x.a*y.c+x.c*y.d)%mod;
    res.d = (x.b*y.c+x.d*y.d)%mod;
    return res;
}
matrix fast_power(matrix a,int b) {
    if(b == 1) return a;
    if(b & 1) return cross(a,fast_power(a,b-1));
    matrix half = fast_power(a,b>>1);
    return cross(half,half);
}
int main() {
    long long n;
    cin >> n;
    n += 2;
    matrix T = {1,1,1,0};
    matrix ans = fast_power(T,n);
    cout << ans.b%mod << '\n';
}
```

G. 神之一手

Checkmate

- 考點： if statement
- 難度： 2 / 10
- 首殺：
- 提交次數：

題目廢話偏多，只要看是否同行或同列就好

Full Solution (100/100)

```
char a,b,c,d;
cin >> a >> b >> c >> d;
if(a == c && b == d) cout << "What\n";
else if(a == c || b == d) cout << "Checkmate\n";
else cout << "Nothing Happened\n";
```

H. 傳說九缺一

10-1=9

- 考點： 迴圈
- 難度： 4 / 10
- 首殺：
- 提交次數：

使用for迴圈或while迴圈跑規定的次數，就這麼簡單
比較特別的是這題是 $Sh1ng$ 出的

Full Solution (100/100)

```
int n,times;
string who;
cin >> n;
for(int i=0;i<n;i++) {
    cin >> who >> times;
    for(int t=0;t<times;t++) {
        cout << "@" << who << '\n';
    }
}
```

I. 聊天室亂源

Chaos

- 基本分考點： for迴圈
- 基本分難度： 6.5 / 10
- 首殺：
- 提交次數：

- Bonus 考點： KMP演算法
- Bonus 難度： `12` / 10
- 首殺：

簡化問題就是給予字串 S_1, S_2
詢問 S_2 是否為 S_1 的子字串

Subtask 1 : $|S_2| = 1$

只有一個字母就檢查整個字串裡面有沒有這個字就好

Subtask1 Solution(50/100)

```
string a;
char b;
cin >> a >> b;
for(int i=0;i<a.size();i++) {
    if(a[i] == b) {
        cout << "fake\n";
        return 0;
    }
}
cout << "real\n";
```

Subtask 2: $|S_2| \leq |S_1| \leq 1000$

1000非常小，直接檢查所有的可能就好了
簡單來說就是試試看每個字為開頭的可能
時間複雜度 $O(|S_1| \times |S_2|)$

Accepted Solution (100/100)

```
string a,b;
cin >> a >> b;
// i 為可能的比對, j 為可能相同的開頭位置
for(int i=0;i<a.size()-b.size()+1;i++) {
    string check = ""; // 記得設為空字串
    for(int j=0;j<b.size();j++) {
        check += a[i+j];
    }
    if(b == check) {
        cout << "fake\n";
        return 0;
    }
}
cout << "real\n";
```

Bonus : $|S_2| \leq |S_1| \leq 2 \times 10^5$

考慮worst case,當 $|S_1|, |S_2|$ 都很大的時候
一秒是不夠跑完的，因此要優化
有個叫做 KMP 的演算法
可以在 $O(|S_1| + |S_2|)$ 的複雜度內解出
有興趣可以自己 Google

Full Solution (200/200)

```cpp
bool KMP_Algorithm(string &haystack, string &needle) {
    int lps[needle.size()];
    lps[0] = 0;
    int prev = 0, ptr = 1;
    while(ptr < needle.size()) {
        if(needle[ptr] == needle[prev])
            lps[ptr] = ++prev, ptr += 1;
        else if(prev == 0)
            lps[ptr] = 0, ptr += 1;
        else
            prev = lps[prev-1];
    }
    int i = 0, j = 0;
    while(i < haystack.size()) {
        if(haystack[i] == needle[j])
            i += 1, j += 1;
        else
            if(j == 0) i += 1;
            else j = lps[j-1];
        if(j == needle.size()) return 1;
    }
    return 0;
}
```

J. 神啊拜託，讓暑假重回第一天吧

Summer

- 考點： 一維陣列
- 難度： 5 / 10
- 首殺：
- 提交次數：

看似沒難度，實則沒難度
把數列存起來之後反過來輸出就好

Full Solution (100/100)

```cpp
int n;
cin >> n;
int arr[n];
for(int i=0;i<n;i++) cin>>arr[i];
for(int i=n-1;i>=0;i--) cout<<arr[i]<<' ';
```

還有黑魔法 reverse

```cpp
int n;
cin >> n;
vector<int> f(n);
for(int &i : f) cin >> i;
reverse(f.begin(),f.end());
for(int &i : f) cout << i << ' ';
```

K. 不時以培訓為由翹課的鄰座 $Zhenzhe$ 同學

Skip Class

- 考點： map
- 難度： 7 / 10
- 首殺：
- 提交次數：

用 map 紀錄什麼課共出現幾次
最後再檢查哪個最多和總和是多少

Full Solution (100/100)

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n,q,x;
    string cls;
    cin >> n >> q;
    map<string,int> mp;
    for(int i=0;i<n;i++) {
        cin >> cls >> x;
        mp[cls] += x;
    }
    int MAX = 0, total = 0;
    string m;
    for(auto [cs, tms] : mp) {
        if(tms > MAX) {
            MAX = tms;
            m = cs;
        }
        total += tms;
    }
    for(int i=0;i<q;i++) {
        cin >> cls;
        cout << mp[cls] << '\n';
    }
    cout << "total:" << total << '\n';
```

```
    cout << "most:" << m << '\n';
}
```

L. 大盤子法師 $Sh1ng$

Plates Master

- 考點： string
- 難度： 5 / 10
- 首殺：
- 提交次數：

Subtask 1. $n = 0$

就說輸出 ***Happy Birthday!*** 就有 1 分

Subtask 2. $n \leq 10, P_i \leq 10^{18}$

既然要判斷是不是 11 的倍數，那就對 11 取餘數就好

Subtask 3. $n \leq 1000, P_i \leq 10^{1000}$

11的倍數判斷方法就是奇數位數跟偶數位數的差是否為11的倍數去判斷

// *Full Solution (100/100)*

```
if(n == 0) {cout<<"Happy Birthday!\n";return 0;}
for(int i=0;i<n;i++) {
    string s;
    cin >> s;
    int diff = 0;
    for(int j=0;j<s.size();j++) {
        if(j & 1) diff += s[j]-'0';
        else diff -= s[j]-'0';
    }
    diff = abs(diff);
    cout << (diff%11==0? "How perfect this plate is!\n" : "Broken\n");
}
```

M. 競程有奇樹，剖分發華茲

O. 社點

Dessert

- 考點： `two-pointer` `prefix-sum` `set` `binary_search`
- 難度： 5 / 10
- 首殺：
- 提交次數：

---

subtask1

唬爛用，$O(n^7)$ 都會過

我不知道你怎麼寫出 $O(n^7)$ 解的

subtask2

暴力解，$O(n^3)$ 都會過

```
vector<int>f(n);

int last = 0;
for(int &i : f) cin>>i, i = last += i;

#define sig(i,j) accumulate(f.begin() + i, f.begin() + r + 1, 0) // 其實就是三層迴圈

int ans = 0;

for(int i = 0;i<n;i++) {
    for(int j = i;j<n;j++) {
        if(sig(i,j) == j) ans++;
    }
}

return ans;
```

subtask3

聰明點的暴力解，$O(n^2)$ 都會過

`聰明點的暴力解` = `暴力解` + ***prefix sum***

```
vector<int>f(n);

int last = 0;
for(int &i : f) cin>>i, i = last += i;

#define sig(i,j) (f.at(j) - (i == 0 ? 0 : f.at(i - 1)))

int ans = 0;
```

```
for(int i = 0;i<n;i++) {
    for(int j = i;j<n;j++) {
        if(sig(i,j) == j) ans++;
    }
}

return ans;
```

subtask4

砸 `multiset` 或 `map` 就可以過了
時間複雜度 $O(n\ log_2^n)$

```
vector<int>f(n);

int last = 0;
for(int &i : f) cin>>i, i = last += i;

int ans = 0;

multiset<int>s({0});
for(int &i : f) {
    ans += s.count(i - k);
    s.insert(i);
}

return ans;
```

subtask5

用 ***two pointer*** ，維護 **左指針** 和 **右指針** ，每次將 **左指針** 往右移動一格，接著移動 **右指針** 來讓這個區間中的和 **不大於** k ，最後只要和是 k 的話就加一。

由於每個元素分別會被 **左指針** 和 **右指針** 穿過一次，因此時間複雜度為 $O(n)$

```
vector<int>f(n);

int last = 0;
for(int &i : f) cin>>i;

int sig = 0;
int ans = 0;
int l,r;l = r = -1;

while(l != n - 1) {
    if(l != -1) sig -= f.at(l++);
    else l++;
```

```
        while(r != n - 1 && sig + f.at(r + 1) <= k) sig += f.at(++r);
        if(sig == k) ans++;
    }
```

證明

為什麼能這樣解呢，這題跟 ***CJ*** 第一題很不一樣的地方在於，這題保證每個元素都是正的，因此我們可以推得當左界固定的時候，使和為 k 的右界只會有至多一個，證明應該滿容易的，就留給各位證了ouob

若不保證每個元素都是正的，則最佳解為 ***subtask4***