24级程序设计竞赛协会新生选拔赛题解

A题

原题链接：B3792 [NICA #1] 年龄问题 - 洛谷 | 计算机科学教育新生态

因为小鱼的爸爸比小鱼年长 y 岁，因此小鱼的年龄是 $x-y$ 岁。
因为小鱼的爸爸比奶奶年轻 z 岁，因此小鱼的奶奶的年龄是 $x+z$ 岁。

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

using namespace std;

int main () {
    int x, y, z;

    cin >> x >> y >> z;

    cout << x - y << " " << x + z;
}
```

B题

原题链接：B3963 [语言月赛 202404] 吃苹果 - 洛谷 | 计算机科学教育新生态

题目意思可以简化为，给你 n 个数，让你找出其中最大的数和最小的数，输出他们的和。
注意：需要开 *long long*，不然就寄了。

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

using namespace std;

const int N = 1e5 + 10, MAX = 2e9 + 10;

int n;
long long a[N];
long long maxx = -MAX, minn = MAX;

int main () {
    cin >> n;

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

        if(a[i] > maxx)
```

```
            maxx = a[i];

        if(a[i] < minn)
            minn = a[i];
    }

    cout << maxx + minn;
}
```

C题

原题链接：[B2052 简单计算器 - 洛谷 | 计算机科学教育新生态]

感觉比 B 题简单，你会 if 就能写。

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

using namespace std;

int main () {

    int a, b;
    char c;

    cin >> a >> b >> c;

    if (c == '+')
        cout << a + b;
    else if (c == '-')
        cout << a - b;
    else if (c == '*')
        cout << a * b;
    else if (c == '/') {
        if (b == 0)
            cout << "Divided by zero!";
        else
            cout << a / b;
    }
    else
        cout << "Invalid operator!";

    return 0;
}
```

D题

原题链接：[P1597 语句解析 - 洛谷 | 计算机科学教育新生态]

输入模式是：`[魔法师名字]:=[另一个魔法师名字或一个一位的神秘数字];`，既然模式都固定了，我们则可以考虑用 $scanf$ 来完成输入。
魔法咒语只会有两种：

- `[魔法师名字]:=[另一个魔法师名字];`
- `[魔法师名字]:=[一个神秘数字];`

则我们可以用数组存不同魔法师对应的魔力，还记得强制类型转换吗？
`magic['a'], magic['b'], magic['c']`

- 当为另一个魔法师名字的时候，直接赋值魔力 `magic[a] = magic[b]`。
- 当为一个神秘数字的时候，把当前魔法师的魔力改为这个神秘数字 `magic[a] = b - '0'`，注意数字是字符，要 `- '0'` 后才行。

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

using namespace std;

int magic['c' + 1];
char a, b;

int main () {
    while(scanf("%c:=%c;", &a, &b)) {
        if(b >= '0' && b <= '9')
            magic[a] = b - '0';
        else
            magic[a] = magic[b];
    }

    cout << magic['a'] << " " << magic['b'] << " " << magic['c'];

    return 0;
}
```

E题

原题链接：[Problem - 1342A - Codeforces]

可以分为两部分

- x, y 中，可以同时被消除的部分，这个消除可以用
  - 单次消除 a → 最后花费 `min(x, y) * (2 * a)`
  - 双次消除 b → 最后花费 `min(x, y) * b`
- x, y 中，被同时消除后，还剩下的部分，直接单次消除
  - `(max(x, y) - min(x, y)) * a`

答案就是他们加起来。

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

using namespace std;

long long t, a, b, x, y;

int main () {
    cin >> t;

    while (t--) {
        cin >> x >> y >> a >> b;

        cout << min(x, y) * min(2 * a, b) + (max(x, y) - min(x, y)) * a <<
endl;
    }

    return 0;
}
```

F题

原题链接：[P8837 [传智杯 #3 决赛] 商店 - 洛谷 | 计算机科学教育新生态]

首先我们知道对于防御值小于所有冲击波当中最小威力的防御魔法是没用的，那么显然对于一个可以被抵挡的冲击波我们匹配一个恰好防御值大于等于其威力的防御魔法是最优的，那么我们先对冲击波和防御魔法按照值从小到大排序(由于这里数据量在 $1e5$ 左右，是不能采用时间复杂度为 N^2 的排序算法的—推荐 $c++$ 内置的 $sort$ 函数)，那么我们可以用 cnt 表示当前剩下的冲击波中威力最小的下标初始为 0，然后从 0 到 n 枚举防御魔法，如果当前防御魔法可以挡下那么就使得 `cnt++`，来抵挡下一个冲击波，最后cnt的值就是抵挡的冲击波数量，当然需要考虑越界情况所以最好判断当 `cnt=m` 时已经抵挡完了就应该跳出循环

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

using namespace std;

const int N = 1e5 + 10;

int a[N], b[N];

int main() {
  ios::sync_with_stdio(0); // 这三行是关闭cin，cout的同步输入流使得输入输出更块，
一般在1e5等大数据读入输出时可以加上加快读入输出时间，所以不要也行不影响
  cin.tie(0);
  cout.tie(0);
```

```
  int n, m;
  cin >> n >> m;
  for (int i = 0; i < n; i++)
    cin >> a[i];
  for (int i = 0; i < m; i++)
    cin >> b[i];

  sort(a, a + n);
  sort(b, b + m);

  int cnt = 0;
  for (int i = 0; i < n && cnt < m; i++) {
    if (a[i] >= b[cnt])
      cnt++;
  }

  cout << cnt;
}
```

G题

原题链接：[P3612 USACO17JAN] Secret Cow Code S - 洛谷 | 计算机科学教育新生态

解析：

当字符串 s 被“玩弄”一次后，s 的长度就会进行一次倍增，我们首先要知道 s 要倍增多少次，s 的长度才会大于等于下标 n。

当字符串 s 倍增若干次后，倍增后的字符串长度 l 第一次大于等于 n，然后我们接下来对下标 n 进行处理。

我们可以发现若 $n > \frac{l}{2}$，下标为 n 的字符与下标为 $n - \frac{l}{2} - 1$ 的字符是相同的，若 $n - \frac{l}{2} - 1 = 0$，那么下标为 n 的字符则与下标为 $\frac{l}{2}$ 的字符相同，可以循环进行以下操作

- 若 $n > \frac{l}{2}$，将 n 更新为 $n - \frac{l}{2} - 1$，若 $n - \frac{l}{2} - 1 = 0$，则将 n 更新为 $\frac{l}{2}$，然后将 l 除 2。
- 若 $n \leq \frac{l}{2}$，则直接将 l 除 2 即可。

直到当 l 等于初始字符串的长度时，结束循环。

朴素代码

```
#include<bits/stdc++.h>
using namespace std;

#define ll long long
```

```cpp
int main(){
    string s;
    ll n;

    cin >> s >> n;
    ll l = s.size();
    s = ' ' + s;

    while(l < n){
        l *= 2;
    }
    while(l > s.size()){
        if(n > l / 2){
            n -= l / 2 + 1;
            if(n == 0)
                n = l / 2;
        }
        l = l / 2;
    }

    cout << s[n] << '\n';
    return 0;
}
```

极致丝滑代码

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
string str;
ll n;

void dfs(ll len) {
    if (n >= len * 2) dfs(len * 2);
    if (n >= len) n = (n + len - 1) % len;
}

int main() {
    cin.tie(0)->sync_with_stdio(false);
    cin >> str >> n;
    n--;
    dfs(str.size());
    cout << str[n];
}
```

H题

原题链接：[P1209 USACO1.3] 修理牛棚 Barn Repair - 洛谷 | 计算机科学教育新生态

解析：

我们只需要考虑编号最小的有人的房子和编号最大的有人的房子，以及这两房子中间的房子。

我们可以反向思考一下，假设有一块屏障从编号最小的有人的房子一直布设到编号最大的有人的房子，那么

问题就变成了最多抹除 $m-1$ 处连续无人空房子的屏障，使得屏障总长度最小。我们只需要优先选择抹除长度较大的连续空房子的屏障就行了，直到抹除完 $m-1$ 次，或者没有空房子可以抹除。

```
#include<bits/stdc++.h>
using namespace std;

bool cmp(int a, int b){
    return a > b;
}

signed main(){
    int m, s, c;
    cin >> m >> s >> c;

    vector<int> a(c);
    vector<int> l;
    for (int i = 0; i < c; i++){
        cin >> a[i];
    }

    sort(a.begin(), a.end());

    int res = a[c - 1] - a[0] + 1;

    for (int i = 1; i < c; i++){
        if(a[i] - a[i - 1] > 1){
            l.push_back(a[i] - a[i - 1] - 1);
        }
    }
    sort(l.begin(), l.end(), cmp);

    for (int i = 0; i < m - 1 && i < l.size(); i++){
        res -= l[i];
    }

    cout << res << '\n';

    return 0;
}
```

I题

原题链接：https://www.luogu.com.cn/problem/P2737

解析：

$Louis$ 独家制作：

给定 N 种钞票，求不能用这些钞票凑出的最小面额。可以先将钞票面额排个序 ค(°ω°ค)

```
for (int i = 1; i <= n; i++) cin >> a[i];
sort(a.begin() + 1, a.end());
```

设 a 为 N 种钞票里面最小的面额，（根据小学数学）任意一个数 X 可表示为 $X = qa + r\ (0 \leq r < a)$。

我们都知道 r 是 X 除 a 的余数，如果我们只用面额为 a 的数去凑钱的话，凑出来的只能是 a 的整数倍！

所以对于每一个余数 r，我们都要用其他面额的数来凑。

对于一个数 b，请问它能凑出多少种类的余数呢?

答案就是：$b\%a, 2b\%a, \cdots, (a-1)b\%a$，最多 $a-1$ 种余数!

我们可以设 `dp[i]` 表示用所有面额能够凑出模 a 余数为 i 的最小面额。

如果有一个余数凑不出来的话！那么我们最小不能凑出的面额就是 ∞ 了！（比如我们只有一种面额 2，那么我们不能凑出的面额就是所有模 2 余 1 的数，即所有奇数！）

如果所有余数都能凑出来的话！那么我们可以找到最大的 `dp[i]`，最小不能凑出的面额就是 `dp[i] - a` 了(๑´ㅅ`๑)!

如何用求出 `dp[i]` 呢?

可以先每次只用一种钞票来凑：

```
for (int i = 1; i <= n; i++)
   for (int j = 1; j < a[1]; j++)
       d[(a[i]*j)%a[1]] = min(d[a[i]*j%a[1]], a[i]*j);
```

然后再用多种钞票来凑：

```
for (int i = 1; i <= n; i ++)
   for (int j = 0; j < a[1]; j++)
       for (int k = 0; k <= a[1]; k++)
           d[j] = min(d[j], d[(j - (k*a[i]%a[1]) + a[1]) % a[1]] + k*a[i]);
```

最后找到最大的 `dp[i]` 就可以啦！（别忘了把 `dp[i]` 初始化为超级大的值哦！）

```
int maxn = 0;
for (int i = 0; i < a[1]; i++) maxn = max(maxn, d[i] - a[1]);
if (maxn == INF) cout << 0 << endl;
else cout << maxn < endl;
```

不会的可以在群里艾特 $Louis$。

J 题

原题链接：[D. Buy Low Sell High]

前置知识点:
C++中小根堆的使用:优先队列,若未掌握可先理解解题思路,在学习了相关知识点后再进行错题补充.
本题思维:反悔贪心

一个来自洛谷较好且简洁的题解:

> 一般想到的的贪心策略是，低价买入高价卖出，能赚钱就赚钱，差值计入 *ans* 里。
>
> 但如果遇到了一个更高价的物品，那这种方法就不是最优解，于是可以实现反悔，将高价买入再以更高价卖出，这样 *ans* 就是最大的了。

若仍然无法理解则可以看看下面的思路,或许对你有些帮助:

首先可以将每天的操作分为买入和卖出两个操作,为了实现先买后卖,我们不可能打乱数据的顺序,**只能从前往后按照事情发展顺序依次考虑**!!!

值得注意的是,最终的结果一定是买入日和卖出日两两一组且一一匹配.

使用集合的方法进行描述,我们不妨用一个集合,以小根堆的形式(即我当前每次取出的都是目前能进行买入的最低价格,此处需要学习**优先队列/堆**的知识)维护当前的可以进行买入操作的买入价格集合.(注意:集合的元素是当前可进行买入的所有价格,而非哪天,我们不用哪天进行集合描述)

对于一个按事情发展顺序的元素有哪些状态和相应要执行的操作的呢?

1. 可行买入价格集合为空,则操作为直接入集合/小根堆
2. 集合非空且当前(天)的价格低于或等于可行买入价格集合中的最小价格,即没有人能够和它进行匹配,那么操作为直接入集合.
3. 集合非空且当前(天)的价格 *nowvalue* 高于可行买入价格集合中的最小价格 *minvalue* ,使用一种贪心的思想,应让**当前的 *minvalue* 离开集合**,成为一次买入操作的买入价格,而当前的 *nowvalue* 成为与之匹配的卖出操作.并将这个差值作为答案的一个贡献值.**重点:同时应该将 *nowvalue* 加入可行买入价格集合两次.原因:若之后有比 *nowvalue* 更应该与 *minvalue* 配对的 *bettervalue* ,那么实际上,这个 *bettervalue* 必然大于 *nowvalue* 且当**

前 *bettervalue* 的最佳替换者为 *nowvalue*
其中,用 *bettervalue* 替换上面 *nowvalue* 的操作等同于1.先给答案提供 *bettervalue* - *nowvalue* 的贡献 2.再将 *nowvalue* 加入集合中(因为此时 *nowvalue* 又被释放成为了一个可行的买入价格),该操作意义等同于将 *nowvalue* 加入可行买入价格集合两次

对于3的重点部分还能如何思考?使用 -(*nowvalue* - *minvalue*)的贡献取消了 *minvalue* 和 *nowvalue* 的匹配操作,并获得了 *bettervalue* - *minvalue* 的贡献,并使得 *nowvalue* 重新进入可行购买价格集合/小根堆.当然,若之后无满足条件即上文提到的**这个 *bettervalue* 大于 *nowvalue* 且当前 *bettervalue* 的最佳替换者为 *nowvalue*** 则执行操作**将 *nowvalue* 加入可行买入价格集合两次**实际上对结果无影响

```cpp
#include <bits/stdc++.h>
using namespace std;

const int maxN = 300001;

int n, pi;
long long ans = 0;

priority_queue< int, vector<int>, greater<int> > Q; //小根堆

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> pi;
        if (!Q.empty() && (Q.top() < pi)) {
            ans += (pi - Q.top());
            Q.pop();
            Q.push(pi);
        }
        Q.push(pi);
    }
    cout << ans << endl;
    return 0;
}
```