本章主要考构造以及对于数据结构的优化(时空复杂度)

译题 - milk

描述

牛奶包装是一个如此低利润的生意,所以尽可能低的控制初级产品(牛奶)的价格变的十分重要。请帮助 Marry 的牛奶制造公司(Merry Milk Makers')以可能的最廉价的方式取得他们所需的牛奶。Marry 的牛奶制造公司从一些农民那购买牛奶，每个农民卖给牛奶制造公司的价格不一定相同。而且,如一只母牛一天只能生产一定量的牛奶,农民每一天只有一定量的牛奶可以卖。每天,Marry 的牛奶制造公司从每个农民那购买一定量的牛奶,少于或等于农民所能提供的最大值。给出 Marry 牛奶制造公司的每日的牛奶需求,连同每个农民的可提供的牛奶量和每加仑的价格,请计算 Marry 的牛奶制造公司所要付出钱的最小值。

注意:每天农民生产的牛奶的总数对 Marry 的牛奶制造公司来说足够的。

格式

PROGRAM NAME: milk

INPUT FORMAT:

(file milk.in)

第 1 行:二个整数, N 和 M。

第一个数值,N,(0<= N<=2,000,000)是 Marry 的牛奶制造公司的一天需要牛奶的数量。

第二个数值,M,(0<= M<=5,000)是提供牛奶的农民个数。

第 2 到 M+1 行:每行二个整数,P_i 和 A_i。

P_i(0<= P_i<=1,000) 是农民 i 的牛奶的价格。

A_i(0 <= A_i <= 2,000,000)是农民 i 一天能卖给 Marry 的牛奶制造公司的牛奶数量。

OUTPUT FORMAT:

(file milk.out)

单独的一行包含单独的一个整数，表示 Marry 的牛奶制造公司拿到所需的牛奶所要的最小费用

SAMPLE INPUT

```
100 5
5 20
9 40
3 10
8 80
6 30
```

SAMPLE OUTPUT

```
630
```

解题报告 - milk

分析

简单的贪心算法。

将所有的牛奶按价格升序排列（用快排），然后从低到高买入，知道买够 m 为止。

贪心的证明：

假设某次买了价格稍高的牛奶，可以得到最优解。那么把这次买的牛奶换成价格更低的牛奶，其它不变，那么所得的解较优。假设不成立。

利用桶排的思想可以把代码压缩到极限!参见代码三! 因其价格范围为[0..1000]可以用计数排序来做，就可以得到一个傻瓜代码(参见代码四)。

注意 0 的情况!

译题 - barn1

描述

在一个暴风雨的夜晚,农民约翰的牛棚的屋顶、门被吹飞了。 好在许多牛正在度假，所以牛棚没有住满。 剩下的牛一个紧挨着另一个被排成一行来过夜。 有些牛棚里有牛，

有些没有。 所有的牛棚有相同的宽度。 自门遗失以后,农民约翰必须尽快在牛棚之前竖立起新的木板。 他的新木材供应者将会供应他任何他想要的长度,但是供应者只能提供有限数目的木板。 农民约翰想将他购买的木板总长度减到最少。

给出 M(1<= M<=50),可能买到的木板最大的数目;S(1<= S<=200),牛棚的总数;C(1 <= C <=S) 牛棚里牛的数目,和牛所在的牛棚的编号 stall_number(1 <= stall_number <= S),计算拦住所有有牛的牛棚所需木板的最小总长度。 输出所需木板的最小总长度作为的答案。

格式

PROGRAM NAME: barn1

INPUT FORMAT:

(file barn1.in)

第 1 行: M , S 和 C(用空格分开)

第 2 到 C+1 行: 每行包含一个整数，表示牛所占的牛棚的编号。

OUTPUT FORMAT:

(file barn1.out)

单独的一行包含一个整数表示所需木板的最小总长度。

SAMPLE INPUT

```
4 50 18
3
4
6
8
14
15
16
17
21
25
26
27
30
```

31
40
41
42
43

SAMPLE OUTPUT

25

[一种最优的安排是用板拦住牛棚 3-8,14-21,25-31,40-43.]

题解 - barn1

分析

思路一

要使木板总长度最少，就要使未盖木板的长度最大。

我们先用一块木板盖住牛棚，然后，每次从盖住的范围内选一个最大的空隙，以空隙为界将木板分成两块，重复直到分成 m 块或没有空隙。

可以用二叉堆来优化算法。

贪心的证明略。

思路二

相比于思路一更容易理解。 显然，当所有木板均用上时，长度最短（证明....）。 正向思维，初始状态有 c 块木板，每块木板只盖住一个牛棚。

由 c 块倒推至 m 块木板,每次寻找这样的两个牛棚:其间距在所有未连接的木板中最小。当这两个牛棚的木板连接时，总木板数减一，总长度增加。

思路三

还可以用动态规划求解,将有牛的牛棚排序后,设置一个函数 d[i,j]表示第 i 个牛修到第 j 个牛需要使用的木版长度,设 f[i,j]表示用前 i 个木版修到第 j 头牛所用的最短长度.

f[i,j]=f[i-1,k-1]+d[k,j] (i<=k<=j)

译题 - calfflac

描述

据说如果你给无限只母牛和无限台巨型便携式电脑(有非常大的键盘),那么母牛们会制造出世上最棒的回文。你的工作就是去寻找这些牛制造的奇观(最棒的回文)。

在寻找回文时不用理睬那些标点符号、空格(但应该保留下来以便做为答案输出),只用考虑字母'A'-'Z'和'a'-'z'。要你寻找的最长的回文的文章是一个不超过 20,000 个字符的字符串。

我们将保证最长的回文不会超过 2,000 个字符(在除去标点符号、空格之前)。

题目名称：calfflac

输入格式

输入文件不会超过 20,000 字符。这个文件可能一行或多行，但是每行都不超过 80 个字符(不包括最后的换行符)。

样例输入(file calfflac.in)

Confucius say:Madam,I'm Adam.

输出格式

输出的第一行应该包括找到的最长的回文的长度。

下一行或几行应该包括这个回文的原文（没有除去标点符号、空格），把这个回文输出到一行或多行（如果回文中包括换行符）。

如果有多个回文长度都等于最大值，输出最前面出现的那一个。

样例输出(file calfflac.out)

11
Madam,I'm Adam

解题报告 - calfflac

分析

没什么好说的，硬搜，数据刚好不会超时。

枚举中间数，然后左右扩展（忽略非字母）至出界和不等，更新最大值。

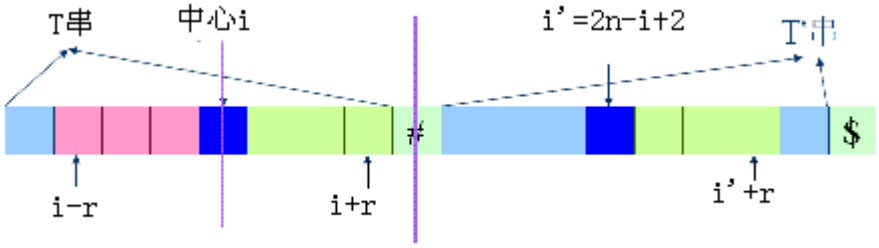
要考虑回文是奇数和偶数两种情况。

输入中的换行符可以维持原样不变。

一个整体 O(n)的算法（参考许智磊论文）

整体用后缀数组实现。

求以 i 为中心向两边扩展的最远值,等价于求 Suffix(i)和 Suffix(i')的最长公共前缀。其中 i'=2n-i+2



解法： 1.初始化答案为 0。S=T+'#'+Reverse(T)+$，得到串 S （O(n)） 2.求出后缀数组 SA、名次数组 Rank （倍增法：O(nlogn) Dc3 算法：O(n)） 3.计算 height 数组并进行标准 RMQ 方法预处理 （O(n)） 4.枚举 i，计算以 i 为对称中心的极长回文串并更新答案 （O(n)）

我的程序明显写得很~~，Dc3+RMQFast，9k 多。 我的程序的整体时间复杂度：O(2n)+O(3*(2n))+O(2n)+O(2n)=O(n)

```
Test 1: TEST OK [0.022 secs, 14568 KB]
Test 2: TEST OK [0.032 secs, 14572 KB]
Test 3: TEST OK [0.032 secs, 14572 KB]
Test 4: TEST OK [0.022 secs, 14568 KB]
Test 5: TEST OK [0.032 secs, 14568 KB]
Test 6: TEST OK [0.022 secs, 14568 KB]
Test 7: TEST OK [0.022 secs, 14568 KB]
Test 8: TEST OK [0.043 secs, 14572 KB]
```

时间效率的确非常理想。

//By Nettle99 从头开始一个一个找也并非很慢

```
Test 1: TEST OK [0.000 secs, 3000 KB]
Test 2: TEST OK [0.000 secs, 3000 KB]
Test 3: TEST OK [0.011 secs, 2996 KB]
Test 4: TEST OK [0.011 secs, 2996 KB]
Test 5: TEST OK [0.000 secs, 2996 KB]
Test 6: TEST OK [0.000 secs, 3000 KB]
Test 7: TEST OK [0.011 secs, 2996 KB]
Test 8: TEST OK [0.097 secs, 2996 KB]
```

总时间及内存都要短.也许也只是巧合...

另一种解法:扩展 kmp+分治复杂度: O(nlogn) 效率:

```
  Test 1: TEST OK [0.000 secs, 676 KB]
  Test 2: TEST OK [0.000 secs, 672 KB]
  Test 3: TEST OK [0.011 secs, 672 KB]
  Test 4: TEST OK [0.000 secs, 672 KB]
  Test 5: TEST OK [0.000 secs, 676 KB]
  Test 6: TEST OK [0.011 secs, 676 KB]
  Test 7: TEST OK [0.011 secs, 672 KB]
  Test 8: TEST OK [0.065 secs, 672 KB]
```

详见何林的论文<<求最长回文子串与最长重复子串>>及代码 3

译题 - crypt1

Prime Cryptarithm 牛式

描述

下面是一个乘法竖式，如果用我们给定的那 n 个数字来取代*，可以使式子成立的话，我们就叫这个式子牛式。

```
      * * *
  x     * *
   -------
     * * *
    * * *
    -------
```

```
    * * * *
```

数字只能取代*，当然第一位不能为 0。

写一个程序找出所有的牛式。

格式

PROGRAM NAME: crypt1

INPUT FORMAT:

(file crypt1.in)

Line 1:数字的个数 n。

Line 2:N 个用空格分开的数字（每个数字都∈{1,2,3,4,5,6,7,8,9}）。

OUTPUT FORMAT:

(file crypt1.out)

共一行，一个数字。表示牛式的总数。

下面是样例的那个牛式。

```
      2 2 2
    x   2 2
     ------
      4 4 4
    4 4 4
  ---------
    4 8 8 4
```

题解 - crypt1

分析一:

```
   S1  S4  S5
×     S2  S3
```

约束条件只有 3 个：第 3、4 行是 3 位数，第 5 行是 4 位数。我们按 S1 到 S5 的顺序搜索。

如果 S1×S2>10 或者 S1×S3>10，则 3、4 行肯定不是 3 位数，剪枝。

即 S1*S2+S4*S2/10>=10 || S1*S3+S4*S3/10>=10 剪枝

分析二：

使用上述方法大多数人都容易想到,但还有一个更简便的方法:

```
program crypt1 (input,output);
var nu:array['1'..'9'] of boolean;
n:1..10;
k,i,j,tot:longint;
function zai(x:longint):boolean;
var s:string;
i:longint;
begin
      str(x,s);
      for i:=1 to length(s) do
      if not(nu[s[i]]) then exit(false);
      exit(true);
end;
begin
      assign(input,'crypt1.in');
assign(output,'crypt1.out');
reset(input);
rewrite(output);
readln(n);
for i:=1 to n do begin read(k);nu[char(k+ord('0'))]:=true;end;
      tot:=0;
    for i:=100 to 999 do{三位数乘二位数}
      for j:=10 to 99 do begin
      if(i*(j div 10)<=999)and(i*(j mod 10)<=999)and(i*j<=9999)
            and(i*(j div 10)>=100)and(i*(j mod 10)>=100)and(i*j>=1000)
      and((zai(i))and(zai(j))and(zai(i*(j div 10)))and(zai(i*(j mod 10)))and(zai(i*j)))
      then inc(tot);
      end;
      writeln(tot);
      close(input);
      close(output)
end.
```

使用字符串辅助,直接枚举这样编程复杂度会大大降低,避免编程错误,也使得判断时大为方便.