NOIP2007 提高组解题报告

前言

本次 noip 大概是几年来最简单的一次，更多考察的是基础算法以及编写代码的能力。可以说前两题细心 + 后两题乱搞就能达到 SD 的一等线。

因此使得某些菜的分数和大牛的区分不明显。这篇题解可能更多是对参加此次 noip 的感想和体会。感觉有必要记录一下，如果能对别人有所帮助那更好。

受本人水平所限，题目中算法并非是最优的，但确保正确以及能在规定时间出解。coding 照着考试时候的重新写的，基本没什么改动，除了第四题加了一点常数优化得以卡着时限 ac。保证在我的 PentiumM587+496MB 的机器上能跑出 400 分来。

让大牛们见笑了。

为控制长度，题目描述不在此赘述，网上一搜就有。

第一题 count

题目分析

拿到这道题目，相信很多人都和我一样比较意外。排序+线性的处理即可。放到 PJ 中也许能更好点。

数据规模 n 在 200000 因此 o(nlogn) 排序可过。数据范围在 1,500,000,000<maxlongint(2,147,483,647)，所谓使用高精度存储完全没有必要。

应该大部分人和我一样直接写个 QSORT 了事。但在初评成绩公布之前，听到有人说有 40% 的特殊数据专门针对 qsort。尽管最后发现是虚惊一场。

qsort 在一定情况下的最差表现为 o(n^2)。

还好 noip 的数据是很温柔的。不然因为这里而失误会冤枉死。这也是经验不足的表现。

所以可以考虑严格 o(nlogn)排序方法或者使用平衡树。参考程序仍然使用了 qsort。

这道题目适合用来初学者练习排序。

参考程序

```
program count;
var
a: array[0..200000] of longint;
n, i, tot: longint;
procedure qsort(l, r: longint);
var
   ii, jj, kk, mid: longint;
begin
   ii := l;
jj := r;
mid := a[(ii+jj) shr 1];
repeat
  while a[ii] < mid do inc(ii);
  while a[jj] > mid do dec(jj);
  if ii <= jj then begin
    kk := a[ii];
  a[ii] := a[jj];
  a[jj] := kk;
  inc(ii);
```

```
  dec(jj);
  end;
until ii > jj;
if ii < r then qsort(ii, r);
if jj > l then qsort(l, jj);
end;
begin
assign(input, 'count.in');
assign(output, 'count.out');
reset(input);
rewrite(output);
readln(n);
for i := 1 to n do read(a[i]);
qsort(1, n);
tot := 1;
write(a[1]);
for i := 2 to n do begin
   if a[i] = a[i-1] then begin
  inc(tot);
end else begin
  writeln(' ', tot);
  write(a[i]);
  tot := 1;
end;
end;
writeln(' ', tot);
close(input);
close(output);
end.
```

第二题 expand

题目分析

看到这道题目，依然没有什么技术含量。所需要的就是细心。

特别注意的几个地方：

输入长度虽然在 100 之内，但是答案最大为 6000 多，因此答案不能用 string 存储，可以使用 ansistring 或者不存储边处理边输出。

只有两个点需要倒序输出。只有一个点存在类似 d-d 的情况。所以数据还是非常弱的。

注意如果一边为数字一边为字母是不可以的。

特别注意连续'-'或者'-'在首尾的情况。

参考程序写的比较笨拙。但对于这种 ws 题目，细心和准确远比速度更重要。

参考程序

```
program expand;
var
```

```
p1, p2, p3, ii, jj, i, ch1, ch2: longint;
s: string;
procedure print(st, en: longint);
begin
    if p3 = 1 then
      for ii := st to en do begin
        for jj := 1 to p2 do write(chr(ii));
      end
    else
      for ii := en downto st do begin
        for jj := 1 to p2 do write(chr(ii));
      end;
end;
procedure print0(st, en: longint);
begin
    for ii := st to en do begin
      for jj := 1 to p2 do write('*');
    end;
end;
procedure judge;
begin
ch1 := ord(s[i-1]);
ch2 := ord(s[i+1]);
if ch1 < ch2 then begin
   if (ch1 >= 48) and (ch1 <= 57) and (ch2 >= 48) and (ch2 <= 57) then begin
     if p1 = 3 then print0(ch1+1, ch2-1) else print(ch1+1, ch2-1);
   exit;
   end;
   if (ch1 >= 97) and (ch1 <= 122) and (ch2 >= 97) and (ch2 <= 122) then begin
     case p1 of
   1: print(ch1+1, ch2-1);
   2: print(ch1-31, ch2-33);
   3: print0(ch1+1, ch2-1);
   end;
   exit;
   end;
end;
write('-');
end;
begin
assign(input, 'expand.in');
assign(output, 'expand.out');
reset(input);
```

```
rewrite(output);
readln(p1, p2, p3);
readln(s);
write(s[1]);
for i := 2 to length(s)-1 do begin
    if s[i] = '-' then begin
   judge;
end else begin
   write(s[i]);
end;
end;
writeln(s[length(s)]);
close(input);
close(output);
end.
```

第三题 game

题目分析

这道题目，一眼就能看出是个 dp，稍微一想，每一行都是独立的，可以分别处理再累加答案。

{

摘自 yy 大牛的题解：

f[i,j]=max{f[i+1,j]+w*a[i] , f[i,j-1]+w*a[j]}

按照 j-i 的大小进行 DP 即可，每一层循环之后，另 w:=w+w。其中 j-i 的最大值是 m-1，最小值是-1。

最后，max{f[i,i-1]，i=0..m}就是所求。

}

上面的 f[i，j]表示从左面取到 i，从右面取到 j 的最优值。

考试中我所写的方程是

f[i, j] = max{f[i-1, j-1] + a[j]*2^i, f[i-1, j] + a[m-i+j+1]*2^i}

其中 f[i, j]表示一共取出 i 个数，其中从左侧取 j 个，那么右侧就是取 i-j 个，所能取到的最优值。

初始值 f[1, 1] = a[1]*2; f[1, 0] = a[m]*2；

最后 max{f[m, i], i=0..m}即为所求。

其实本质上是一样的。

dp 说完了，再讲高精。由于数据规模比较大，直接高精好像会 tle。需要压位，就是扩大进制数。

说也走运，当时一开始考虑使用万进制是因为题目中 a[i]范围到 1000，不想麻烦的处理 a[i]，最后甚至想改回十进制。

这样时间复杂度是 o(knm^2),k 是高精度的时间常数。

参考程序中使用了较多的 procedure&function，ws 的 4kb 的 code，和考试时候写的基本一致。尽管使得程序冗长，但是感觉分段处理更易于理解和检查。考试时候最后检查，发现了高精中的一个小错误。挽回了 60 分。

现在看来还是有累赘。比如计算 2^n 后可以记录，不必重复计算。
最后想说，这道题目想法不难，关键是千万不要写错。要是因为一些小失误而挂掉真是能欲哭无泪。不过题目中给出了 3 个样例，因此还是比较好调试的。
如果使用 double/int64/extended 可以过 6 个点。搜索也能有部分分。

参考程序

```
program game;
var
f: array[0..80, 0..80, 0..20] of longint;
ans, max, temp, now: array[0..20] of longint;
a: array[0..80] of longint;
lennow, lenmax, lenans, ii, lent, n, m, dep, i, j: longint;
procedure print;
begin
    write(ans[lenans]);
for ii := lenans-1 downto 1 do begin
  if ans[ii] < 1000 then write(0);
  if ans[ii] < 100 then write(0);
  if ans[ii] < 10 then write(0);
  write(ans[ii]);
end;
end;
procedure timenow;
begin
    for ii := 1 to lennow do begin
  inc(now[ii], now[ii]);
end;
for ii := 1 to lennow do
  if now[ii] >= 10000 then begin
    inc(now[ii+1]);
  dec(now[ii], 10000);
  end;
if now[lennow+1] > 0 then inc(lennow);
end;
procedure time(x: longint);
begin
    fillchar(temp, sizeof(temp), 0);
lent := lennow;
for ii := 1 to lent do begin
  temp[ii] := a[x] * now[ii];
end;
for ii := 1 to lent-1 do begin
  inc(temp[ii+1], temp[ii] div 10000);
  temp[ii] := temp[ii] mod 10000;
```

```
end;
while temp[lent] > 10000 do begin
  temp[lent+1] := temp[lent] div 10000;
  temp[lent] := temp[lent] mod 10000;
  inc(lent);
end;
end;
procedure add(x, y: longint);
begin
   if lent < f[x, y, 0] then lent := f[x, y, 0];
for ii := 1 to lent do begin
  inc(temp[ii], f[x, y, ii]);
  if temp[ii] >= 10000 then begin
    inc(temp[ii+1]);
  dec(temp[ii], 10000);
  end;
end;
if temp[lent+1] > 0 then inc(lent);
end;
procedure tihuan(x, y: longint);
begin
   f[x, y, 0] := lent;
for ii := 1 to f[x, y, 0] do f[x, y, ii] := temp[ii];
end;
function panduan(x, y: longint): boolean;
begin
   if f[x, y, 0] > lent then exit(false);
if f[x, y, 0] < lent then exit(true);
for ii := lent downto 1 do begin
  if f[x, y, ii] > temp[ii] then exit(false);
  if f[x, y, ii] < temp[ii] then exit(true);
end;
exit(false);
end;
procedure huanmax(x: longint);
begin
   lenmax := f[m, x, 0];
for ii := 1 to lenmax do max[ii] := f[m, x, ii];
end;
function panmax(x: longint): boolean;
begin
   if lenmax > f[m, x, 0] then exit(false);
if lenmax < f[m, x, 0] then exit(true);
```

```
for ii := lenmax downto 1 do begin
  if max[ii] > f[m, x, ii] then exit(false);
  if max[ii] < f[m, x, ii] then exit(true);
end;
exit(false);
end;
procedure addans;
begin
   if lenans < lenmax then lenans := lenmax;
for ii := 1 to lenans do begin
  inc(ans[ii], max[ii]);
  if ans[ii] >= 10000 then begin
    inc(ans[ii+1]);
  dec(ans[ii], 10000);
  end;
end;
if ans[lenans+1] > 0 then inc(lenans);
end;
begin
assign(input, 'game.in');
assign(output, 'game.out');
reset(input);
rewrite(output);
readln(n, m);
lenans := 0;
fillchar(ans, sizeof(ans), 0);
for dep := 1 to n do begin
   for i := 1 to m do read(a[i]);
fillchar(f, sizeof(f), 0);
fillchar(max, sizeof(max), 0);
f[1, 1, 0] := 1;
f[1, 1, 1] := a[1] * 2;
f[1, 1, 0] := 1;
f[1, 0, 1] := a[m] * 2;
fillchar(now, sizeof(now), 0);
now[1] := 2;
lennow := 1;
for i := 2 to m do begin
  timenow;
  time(i);
  add(i-1, i-1);
  tihuan(i, i);
```

```
  time(m-i+1);
  add(i-1, 0);
  tihuan(i, 0);
  for j := 1 to i-1 do begin
    time(j);
  add(i-1, j-1);
    tihuan(i, j);
  time(m-i+j+1);
  add(i-1, j);
  if panduan(i, j) then tihuan(i, j);
  end;
end;
huanmax(0);
for i := 1 to m do begin
  if panmax(i) then huanmax(i);
end;
addans;
end;
print;
close(input);
close(output);
end.
```

第四题 core

题目分析

尽管这道题目是最有价值写题解的，但是我这篇应该会让人失望。

目前最快的算法是 o(n)的。还有很多诸如 o(nlogn)、o(n^2)的等等。然后 sdyy 告诉我他的 o(n^3)的。然后我这个是跟着题目描述上当的 o(kn^3)的，k 是直径的条数。

所以写出考试时候的程序发现自测第九个点要 1.5s，优化后总算在 0.9xs 出解。

看到题目，发现终于在 noip 见到图了...激动一个。仔细读完题{也许这就进了陷阱}，看到 n=300，立刻想 o(n^3)的算法应该可以。

但是不幸的在现场没有发现一个不能算常数的常数...至少从 oibh 上下了数据之后，我就知道自己的 10 分时怎么丢掉的了。

先说一下我的相当低效的算法吧。基本上是按照题意的笨做法。

首先 floyd 全局最短路{这就比较低效}，然后 n^2 枚举出最长路径起始点并且记录{更低效了}。对于每一条最长路径，先 n^2 求出经过的每个点。然后从一个方向枚举路径上每个点，计算出从这个点出发的最长的不超过 s 的路径并且计算出各个点到该路径的最短距离。统计最短距离的最大值，如果已经不比当前答案更优了就可以枚举下一条路径了。感觉讲的还是不怎么清楚，有兴趣的就看一下我的 ws 程序吧。

由于最长路径条数 k 最坏情况下是是 o(n^2)，因此第九个点甚至是可以看作是 o(n^5)。没办法算法差只能对常数优化苛刻点了。

真正考试的时候想不出好算法，能这样也还好。

比较奇怪的问题，使用 floyd 时记录中间的节点竟然更慢了？

spoj 上也出现了这道题的加强形式，好像 n= 10000。显然就需要更好的算法了。那样就请看别的题解吧。我水平实在有限。

参考程序

```
program core;
const
maxn = 65536;
var
n, s, i, j, aa, bb, k, temp, st, en, ans, tou, max, len, dep, now: longint;
w, f, mid: array[0..300, 0..300] of longint;
p: array[0..50000, 1..2] of integer;
a: array[0..300] of integer;
d: array[0..300] of longint;
begin
assign(input, 'core.in');
assign(output, 'core.out');
reset(input);
rewrite(output);
readln(n, s);
for i := 1 to n do begin
    w[i, i] := 0;
f[i, i] := 0;
    for j := i+1 to n do begin
  f[i, j] := maxn;
  f[j, i] := maxn;
  w[i, j] := maxn;
  w[j, i] := maxn;
end;
end;
for i := 2 to n do begin
    read(aa, bb);
readln(w[aa, bb]);
w[bb, aa] := w[aa, bb];
f[aa, bb] := w[aa, bb];
f[bb, aa] := w[aa, bb];
end;
for k := 1 to n do begin
    for i := 1 to n do begin
  for j := 1 to n do begin
    if f[i, j] > f[i, k] + f[k, j] then f[i, j] := f[i, k] + f[k, j];
  end;
end;
end;
```

```
fillchar(p, sizeof(p), 0);
tou := 0;
max := 0;
for i := 1 to n-1 do begin
    for j := i+1 to n do begin
   if f[i, j] > max then begin
     p[1, 1] := i;
   p[1, 2] := j;
   tou := 1;
   max := f[i, j];
   end else begin
     if f[i, j] = max then begin
    inc(tou);
    p[tou, 1] := i;
    p[tou, 2] := j;
   end;
   end;
end;
end;
ans := maxn;
for dep := 1 to tou do begin
    len := 0;
st := p[dep, 1];
en := p[dep, 2];
now := en;
while f[now, st] <> w[now, st] do begin
   for k := 1 to n do begin
     if (f[st, now] = f[st, k] + w[k, now]) and (k <> now) then begin
    inc(len);
    a[len] := now;
    now := k;
    break;
   end;
   end;
end;
inc(len);
a[len] := now;
inc(len);
a[len] := st;
for i := 1 to len do begin
   temp := 0;
   st := a[i];
   for k := 1 to n do begin
```

```
    d[k] := f[st, k];
  end;
  for j := i+1 to len do begin
    en := a[j];
    if f[st, en] > s then break;
  for k := 1 to n do begin
   if d[k] > f[k, en] then d[k] := f[k, en];
  end;
  end;
  for k := n downto 1 do begin
    if d[k] > temp then begin
   temp := d[k];
   if temp >= ans then break;
  end;
  end;
  if temp < ans then ans := temp;
end;
end;
writeln(ans);
close(input);
close(output);
end.
```

后记

这次 noip 还有一些特点，比如难度随题号递增了，不像 06 年那样让人一头雾水。测试数据给的也很充分，给我这种懒人以方便，但是无形中降低了难度。因为如何自己设计测试数据也是十分有学问的事情。并且过分依赖题目所给的测试数据也会很容易陷入出题人的圈套中。数据还是和去年一样的温柔。比如最后一道题目 yy 大牛使用 integer 可以 ac。

不管怎么评价 noip 的题目，它也不能改变，只有让我们去努力适应。明年的题目是否会加大难度呢？数据依然会很让人很舒服么？总之可以说，作为纯普及性质的 noip，所需要的并不是特别高深的算法，而是需要一种弱题不失误，难题不手软的作风。