优先队列

[NOIP2004 提高组] 合并果子 / [USACO06NOV] Fence Repair G

题目描述

在一个果园里，多多已经将所有的果子打了下来，而且按果子的不同种类分成了不同的堆。多多决定把所有的果子合成一堆。

每一次合并，多多可以把两堆果子合并到一起，消耗的体力等于两堆果子的重量之和。可以看出，所有的果子经过 $n-1$ 次合并之后， 就只剩下一堆了。多多在合并果子时总共消耗的体力等于每次合并所耗体力之和。

因为还要花大力气把这些果子搬回家，所以多多在合并果子时要尽可能地节省体力。假定每个果子重量都为 1 ，并且已知果子的种类 数和每种果子的数目，你的任务是设计出合并的次序方案，使多多耗费的体力最少，并输出这个最小的体力耗费值。

例如有 3 种果子，数目依次为 1 ， 2 ， 9 。可以先将 1 、 2 堆合并，新堆数目为 3 ，耗费体力为 3 。接着，将新堆与原先的第三堆合并，又得到新的堆，数目为 12 ，耗费体力为 12 。所以多多总共耗费体力 $=3+12=15$ 。可以证明 15 为最小的体力耗费值。

输入格式

共两行。
第一行是一个整数 $n(1 \leq n \leq 10000)$ ，表示果子的种类数。

第二行包含 n 个整数，用空格分隔，第 i 个整数 $a_i(1 \leq a_i \leq 20000)$ 是第 i 种果子的数目。

输出格式

一个整数，也就是最小的体力耗费值。输入数据保证这个值小于 2^{31} 。

样例 #1

样例输入 #1

```
3
1 2 9
```

样例输出 #1

```
15
```

提示

对于 30% 的数据，保证有 $n \leq 1000$;

对于 50% 的数据，保证有 $n \leq 5000$;

对于全部的数据，保证有 $n \leq 10000$。

题目分析

考虑贪心策略，每次取场上最小的两堆并起来，一直操作到只剩下一堆果子为止。过程中耗费的体力值即答案。之前的《合并果子》使用了两个队列维护未合并过的果子的最小值，以及已经合并过的果子的最小值的做法，这样可以高效地寻找哪两堆果子最小。既然如此，为什么不维护一个堆，存放所有果子的重量，每次合并时去除两个最小的，然后合并后再放回去？

幸运的是，$C++$ 提供了优先队列这个数据结构，也就是 STL 中的 $priority\ \ queue$，底层就是由刚刚介绍过的堆实现的。要使用优先队列，需要包含 $queue$ 头文件，优先队列支持的基础操作如下：

1. priority _ queue q: 新建一个保存int型变量的优先队列q，默认是大根堆。
2. priority_queue<int,vector,greater >q: 新建一个小根堆。
3. $q.top()$：优先队列查询最大值（或最小值）；
4. $q.pop()$：将最大值（或最小值）弹出队列；
5. $q.push(x)$：将 x 加入优先队列；

和大多数 STL 容器一样，$queue$ 使用 $q.empty()$ 判断是否为空，用 $q.size()$ 获取大小。

回到本题，只需先把所有的数都扔进优先队列，然后对于每次合并，取出最小值，然后弹出；再取出最小值，然后弹出；将之前取出的两个值之和，加入优先队列，同时统计答案。直到队列中只剩下一个元素。

```cpp
#include<cstdio>
#include<algorithm>
#include<queue>
using namespace std;

priority_queue<int,vector<int>,greater<int> >q; //维护最小值

int main()
{
    int n,x,y,ans=0,cost;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        scanf("%d",&x),q.push(x);
    while(q.size()>1)
    {
        x=q.top();
        q.pop();
        y=q.top();
        q.pop();
        cost=x+y;
        q.push(cost);
        ans+=cost;
    }
    printf("%d\n",ans);
    return 0;
 }
```

请注意，上述代码在定义优先队列时，在最后一个">" 前加入一个空格，刻意规避了 $\cdots greater<int>>$ 的写法。这是一个值得推荐的代码习惯，某些 $C++$ 编译器中，不加空格会被认为是右移运算。

[NOIP2016 提高组] 蚯蚓

题目描述

本题中，我们将用符号 $\lfloor c \rfloor$ 表示对 c 向下取整，例如：$\lfloor 3.0 \rfloor = \lfloor 3.1 \rfloor = \lfloor 3.9 \rfloor = 3$。

蛐蛐国最近蚯蚓成灾了！隔壁跳蚤国的跳蚤也拿蚯蚓们没办法，蛐蛐国王只好去请神刀手来帮他们消灭蚯蚓。

蛐蛐国里现在共有 n 只蚯蚓（n 为正整数）。每只蚯蚓拥有长度，我们设第 i 只蚯蚓的长度为 a_i（$i = 1, 2, \ldots, n$），并保证所有的长度都是非负整数（即：可能存在长度为 0 的蚯蚓）。

每一秒，神刀手会在所有的蚯蚓中，准确地找到最长的那一只（如有多个则任选一个）将其切成两半。神刀手切开蚯蚓的位置由常数 p（是满足 $0 < p < 1$ 的有理数）决定，设这只蚯蚓长度为 x，神刀手会将其切成两只长度分别为 $\lfloor px \rfloor$ 和 $x - \lfloor px \rfloor$ 的蚯蚓。特殊地，如果这两个数的其中一个等于 0，则这个长度为 0 的蚯蚓也会被保留。此外，除了刚刚产生的两只新蚯蚓，其余蚯蚓的长度都会增加 q（是一个非负整常数）。

蛐蛐国王知道这样不是长久之计，因为蚯蚓不仅会越来越多，还会越来越长。蛐蛐国王决定求助于一位有着洪荒之力的神秘人物，但是救兵还需要 m 秒才能到来……（m 为非负整数）

蛐蛐国王希望知道这 m 秒内的战况。具体来说，他希望知道：

- m 秒内，每一秒被切断的蚯蚓被切断前的长度（有 m 个数）；
- m 秒后，所有蚯蚓的长度（有 $n + m$ 个数）。

蛐蛐国王当然知道怎么做啦！但是他想考考你……

输入格式

第一行包含六个整数 n, m, q, u, v, t，其中：n, m, q 的意义见【问题描述】；u, v, t 均为正整数；你需要自己计算 $p = u/v$（保证 $0 < u < v$）；t 是输出参数，其含义将会在【输出格式】中解释。

第二行包含 n 个非负整数，为 $a_1, a_2, \ldots, a_n$，即初始时 n 只蚯蚓的长度。

同一行中相邻的两个数之间，恰好用一个空格隔开。

保证 $1 \leq n \leq 10^5$，$0 \leq m \leq 7 \times 10^6$，$0 < u < v \leq 10^9$，$0 \leq q \leq 200$，$1 \leq t \leq 71$，$0 \leq a_i \leq 10^8$。

输出格式

第一行输出 $\left\lfloor \frac{m}{t} \right\rfloor$ 个整数，按时间顺序，依次输出第 t 秒，第 $2t$ 秒，第 $3t$ 秒，……被切断蚯蚓（在被切断前）的长度。

第二行输出 $\left\lfloor \frac{n+m}{t} \right\rfloor$ 个整数，输出 m 秒后蚯蚓的长度；需要按从大到小的顺序，依次输出排名第 t，第 $2t$，第 $3t$，……的长度。

同一行中相邻的两个数之间，恰好用一个空格隔开。即使某一行没有任何数需要输出，你也应输出一个空行。

请阅读样例来更好地理解这个格式。

样例 #1

样例输入 #1

```
3 7 1 1 3 1
3 3 2
```

样例输出 #1

```
3 4 4 4 5 5 6
6 6 6 5 5 4 4 3 2 2
```

样例 #2

样例输入 #2

```
3 7 1 1 3 2
3 3 2
```

样例输出 #2

```
4 4 5
6 5 4 3 2
```

样例 #3

样例输入 #3

```
3 7 1 1 3 9
3 3 2
```

样例输出 #3

```
//空行
2
```

提示

【样例解释1】

在神刀手到来前：3只蚯蚓的长度为$3,3,2$。

1秒后：一只长度为3的蚯蚓被切成了两只长度分别为1和2的蚯蚓，其余蚯蚓的长度增加了1。最终4只蚯蚓的长度分别为$(1,2),4,3$。括号表示这个位置刚刚有一只蚯蚓被切断

2秒后：一只长度为4的蚯蚓被切成了1和3。5只蚯蚓的长度分别为：$2, 3, (1, 3), 4$。

3秒后：一只长度为4的蚯蚓被切断。6只蚯蚓的长度分别为：$3, 4, 2, 4, (1, 3)$。

4秒后：一只长度为4的蚯蚓被切断。7只蚯蚓的长度分别为：$4, (1, 3), 3, 5, 2, 4$。

5秒后：一只长度为5的蚯蚓被切断。8只蚯蚓的长度分别为：$5, 2, 4, 4, (1, 4), 3, 5$。

6秒后：一只长度为5的蚯蚓被切断。9只蚯蚓的长度分别为：$(1, 4), 3, 5, 5, 2, 5, 4, 6$。

7秒后：一只长度为6的蚯蚓被切断。10只蚯蚓的长度分别为：$2, 5, 4, 6, 6, 3, 6, 5, (2, 4)$。所以，7秒内被切断的蚯蚓的长度依次为$3, 4, 4, 4, 5, 5, 6$。7秒后，所有蚯蚓长度从大到小排序为
$6, 6, 6, 5, 5, 4, 4, 3, 2, 2$

【样例解释2】

这个数据中只有$t = 2$与上个数据不同。只需在每行都改为每两个数输出一个数即可。

虽然第一行最后有一个6没有被输出，但是第二行仍然要重新从第二个数再开始输出。

【样例解释3】

这个数据中只有$t = 9$与上个数据不同。

注意第一行没有数要输出，但也要输出一个空行。

【数据范围】

<table>
<tr><th>测试点</th><th>n</th><th>m</th><th>t</th><th>a_i</th><th>v</th><th>q</th></tr>
<tr><td>1</td><td>$=1$</td><td rowspan="3">$=0$</td><td rowspan="8">$=1$</td><td rowspan="14">$\le 10^6$</td><td rowspan="13">≤ 2</td><td rowspan="5">$=0$</td></tr>
<tr><td>2</td><td>$=10^3$</td></tr>
<tr><td>3</td><td>$=10^5$</td></tr>
<tr><td>4</td><td>$=1$</td><td rowspan="4">$=10^3$</td></tr>
<tr><td>5</td><td>$=10^3$</td></tr>
<tr><td>6</td><td>$=1$</td><td rowspan="2">≤ 200</td></tr>
<tr><td>7</td><td>$=10^3$</td></tr>
<tr><td>8</td><td>$=5\times10^4$</td><td>$=5\times10^4$</td><td rowspan="7">$=0$</td></tr>
<tr><td>9</td><td rowspan="6">$=10^5$</td><td>$=10^5$</td><td>$=2$</td></tr>
<tr><td>10</td><td>$=2\times10^6$</td><td>$=21$</td></tr>
<tr><td>11</td><td>$=2.5\times10^6$</td><td>$=26$</td></tr>
<tr><td>12</td><td>$=3.5\times10^6$</td><td>$=36$</td><td rowspan="2">$\le 10^7$</td></tr>
<tr><td>13</td><td>$=5\times10^6$</td><td>$=51$</td><td rowspan="2">$\le 10^9$</td></tr>
<tr><td>14</td><td>$=7\times10^6$</td><td>$=71$</td><td rowspan="6">$\le 10^8$</td></tr>
<tr><td>15</td><td rowspan="2">$=5\times10^4$</td><td>$=5\times10^4$</td><td>$=1$</td><td>≤ 2</td><td rowspan="6">≤ 200</td></tr>
<tr><td>16</td><td>$=1.5\times10^5$</td><td>$=2$</td></tr>
<tr><td>17</td><td rowspan="4">$=10^5$</td><td>$=10^5$</td><td>$=3$</td><td rowspan="4">$\le 10^9$</td></tr>
<tr><td>18</td><td>$=3\times10^5$</td><td>$=4$</td></tr>
<tr><td>19</td><td>$=3.5\times10^6$</td><td>$=36$</td></tr>
<tr><td>20</td><td>$=7\times10^6$</td><td>$=71$</td></tr>
</table>

题目分析

参考合并果子，每次从蚯蚓堆中找到最大的一条，把它切掉，放回去就行。但是每次放回时，其他的蚯蚓全部都要变长 q ,怎么处理呢？除了这两条蚯蚓以外，大家都增加长度 q ，相当于这两条蚯蚓减少 q （计算长度的时候还要全部加上若干个 q ）

```
#include<cstdio>
#include<queue>
```

```cpp
using namespace std;
priority_queue<int>ans;//存储蚯蚓长度的优先队列
int main()
{
    int n,m,q,u,v,t,tmp;
    int alldec=0; //所有蚯蚓被减了偏移量
    scanf("%d%d%d%d%d%d",&n,&m,&q,&u,&v,&t);
    for(int i=1;i<=n;++i)
    {
        scanf("%d",&tmp);
        ans.push(tmp);
     }
    for(int i=1;i<=m;++i)
    {
        int top=ans.top()+alldec;//从蚯蚓堆中取最长的，加上偏移量
        ans.pop();
        int q1,q2; //被切后的两只
        q1=int(1.0*u/v*top);
        q2=top-q1;
        alldec+=q; //累加偏移量
        q1-=alldec; //减掉偏移量
        q2-=alldec;
        ans.push(q1);//放回第一只
        ans.push(q2);
        if(i%t==0)
            printf("%d ",top); // 根据题目要去输出的第一行
    }
    printf("\n");
    for(int i=1;ans.size();++i)
    {
        if(i%t==0)
            printf("%d ",ans.top()+alldec);//要求输出的第二行
        ans.pop(); //弹出
    }
    return 0;
 }
```

还能继续改进吗?

这种解法进行 m 次操作，单次操作的时间复杂度是 $O(logn)$, 所以总的时间复杂度是 $O(mlogn)$ ，稍稍大了一点，会有少量测试点无法在规定的时间内运行完毕，必须优化时间复杂度。

还记得"合并果子"中使用两个队列的技巧吗？本题也是可以的。先被切掉的两条蚯蚓一定比后面切掉的对应的两条蚯蚓长。假设两只蚯蚓长度分别为 $a_1, a_2(a_1 > a_2)$,第一条被分成了 $a_{11} = pa_1$ 和 $a_{12} = (1-p)a_1$ 两条。一段时间后，所有蚯蚓都变长了 s ，之前被切的两条长度为 $a_{22} = (1-p)(a_1+s)$,可以得到 $a_{11} > a_{21}$ 且 $a_{12} > a_{22}$ 。

因此，可以准备三个队列，第一个队列存放原来的蚯蚓，第二个队列寸每次切蚯蚓后的第一段，第三个队列存第二段。这三个队列都是单调不增的。比较这三个队列的队首，选取最大的一个来进行切割。注意也要上一段代码中的长度偏移量。

```cpp
#include<cstdio>
#include<queue>
#include<algorithm>
#define inf 2100000000;
```

```cpp
using namespace std;
const int maxn=1e5+20;
int n,m,q,u,v,t;
int a[maxn],x,y;
queue<int>q1,q2,q3;

bool cmp(int A,int B)
{
    return A>B;
 }

int findmax()
{
    int x,x1,x2,x3;
    if(!q1.empty()) x1=q1.front();else x1= -inf;
    if(!q2.empty()) x2=q2.front();else x2= -inf;
    if(!q3.empty()) x3=q3.front();else x3= -inf;
    if(x1>x2 && x1>x3)
    {
        x=x1;
        q1.pop();
    }
    else
        if(x2>x3)
        {
            x=x2;
            q2.pop();
        }
        else
            {
                x=x3;
                q3.pop();
            }
    return x;
}

int main()
{
    scanf("%d%d%d%d%d%d",&n,&m,&q,&u,&v,&t);
    for(int i=1;i<=n;i++)scanf("%d",&a[i]);
    sort(a+1,a+n+1,cmp);
    for(int i=1;i<=n;i++)q1.push(a[i]);
    for(int i=1;i<=m;i++)
    {
        x=findmax()+q*(i-1); //原蚯蚓长度
        if(i%t==0)printf("%d ",x);
        y=1.0*x*u/v; // 第一只
        x=x-y;// 第二只
        y-=q*i;q2.push(y);
        x-=q*i;q3.push(x);
    }
    printf("\n");
    for(int i=1;i<=n+m;i++)
    {
        x=findmax()+q*m;
        if(i%t==0)
```

```
61              printf("%d ",x);
62      }
63      return 0;
64  }
```