问题描述： 有一棵无限的树 T：它每个节点 u 有 r 个儿子；u 到第 i 个孩子的边的费用是 c_i。假定 $c_1<=c_2<=\ldots<=c_r$。节点 u 的费用是根到 u 的路径上所有边的费用之和（根节点费用为 0）。求 T 的有 n 个叶子的子树，叶子的费用之和最小。

为了方便，树 T 中的节点按费用从小到大编号：根节点编号为 1，根的第 1 个儿子为 2，等等（多个节点的费用相同时它们之间按任意顺序进行编号即可）。

假定最优解有 m 个内部节点（注意 m 是未知的）。（内部节点=非叶子节点）
我们知道 $m(r-1)+1>=n$ 。因此 $m(r-1)>=(n-1)$。
定义 $m_{min}=(n-1)/(r-1)$取上整。有 $m >= m_{min}$。

对 $x>=m_{min}$，定义 **T[x]**为如下的一棵树：它的内部节点为{1,…,x}。它的叶子节点为 Leave{1,…,x}中费用最小的 n 个节点。（注意 T[x]是 T 的子树）
Leave{V}={ u | u 是 V 中某个节点的儿子，且 u 不属于 V}。

容易证明，**T[m]是最优解**。（因为最优解内部节点不妨选 1..m）。
根据这一结论，算法一是正确的。

算法一：枚举 x 从 m_{min} 到 n-1，计算出 T[x]的费用。找到其中最小的即为答案。
（很显然，在最优解中内部节点个数<=n-1，因为内部节点有 2 个或以上儿子）
举例。r=3.n=3. C=(1,1,5)。$m_{min}=1$。
T[1] 包含 1 个非叶子和 3 个叶子节点 a,b,c。费用为 1+1+5=7。
T[2] 包含 2 个非叶子和 3 个叶子 aa,ab,b。费用为 2+2+1=5。为最优解。

提前终止条件 I：当 T[x]的节点 x 少于 2 个儿子在 T[x]，可终止。（正确性不难证明。读者请思考。）使用“提前终止条件 II”也能 AC。
提前终止条件 II： 注意到最小的那个叶子的费用是递增的。如果 n 倍该费用超过了当前的最优值，那么可以终止。（正确性显然）。

如何实现算法一？
借助一个**虚拟的**表格进行思考比较简单。这个表中第 i 行、第 j 列的元素的值为 c_i+b_j。b_j 表示编号为 j 节点的费用。举例如下。c=(2,3,5)，n=8。
从第 1 列开始构造表格(注意 $b_1=0$)。然后，从第 1 列中选择其中最小的一个数，即为 b_2。然后在前 2 列中找第 2 小的数，即为 b_3，以此类推。

$c_1=2$	**2**	**4**	5	6
$c_2=3$	**3**	5	6	7
$c_3=5$	5	7	8	9
	$b_1=0$	$b_2=2$	$b_3=3$	$b_4=4$

x=4 时，8 个叶子的费用为 5+5+5+6+6+7+7+8。表格中的 9 淘汰，以后也不会选了。
x=5 时，表格中最后一列 8 和 10 被淘汰（如下）。费用增加（5 换成了 7）。

$c_1=2$	**2**	**4**	5	6	7
$c_2=3$	**3**	**5**	6	7	8
$c_3=5$	5	7	8	9	Ignore
	$b_1=0$	$b_2=2$	$b_3=3$	$b_4=4$	$b_5=5$

注意中间节点 x=5 只有一个叶子被选中了，因此根据终止条件 I，算法可停下来了。

黄色格子（内部节点）红色格子（多余的被淘汰的叶子）的维护

上表黄色为内部节点，红色为被淘汰了的叶子节点，白色为当前的 n 个叶子节点。

维护 r 个数 $d_1,...,d_r$ 用来表示每行第 1 个非黄色节点的位置。将第$(1,d_1),\ldots(r,d_r)$ 这 r 个元素放入优先队列中。每次选择最小的一个(i,d_i)作为 b_{x+1}，并将 d_i 修改为 d_i+1。

红色的节点也有类似的处理方法。假设 1~h 行无红色格子，h+1~r 行有。对于每个 i>h，我们维护 e_i 表示第 i 行最后一个非红色格子的位置。当白色格子数超过 n 时，需将最大的一个改成红色。为此，只要从$(h+1,e_{h+1}),(h+2,e_{h+2}),\ldots,(r,e_r)$这些格子以及第 h 行最后一个元素中，找最大的元素即可。同样的，借助一个优先队列来实现。

时间复杂度的分析。

变量 x 的枚举量为 O(n)。 染红次数为 O(n r)。因此总复杂度为 O(n r log r)。

下面证明，染红次数其实为 O(n log r)这个级别。因此总复杂度为 O(n log r log r)。

事实上，n*r 个格子有很多都不会被染红——比如说，一行中出现了一个红色格子，那么它右边的那些始终都是红色的。(比如上表右下方的那个“Ignore”元素)。

可以发现，只有满足 i*j<=2n 的格子才是有可能成为白格的。对于(i,j)>2n 这个格子，(1,1)-(i,j)这个矩形区域内有足够多的格子，即使删掉 n-1 个内部节点（黄色格子），也不需要选(i,j)这个节点。统计一下这种 j<=2n/i 的格子点，为 O(n log r)。

后续的更快的算法需要用到的一个引理

单峰性引理： $cost(T[m_{min}])>cost(T[m_{min}+1])>\ldots cost(T[m])<=cost(T[m+1])<=\ldots$

(这个引理的证明不是 trivial 的，但是时间关系，本题解中跳过😐)

事实上，cost 满足**凸性：** cost(T[x+1])-cost(T[x]) >= cost(T[x])-cost(T[x-1])

这个凸性显然蕴含着上面的单峰性。证明见<SiCOMP96>的 OBSERVATION 3.6.

提前终止条件 3：当 cost(T[x])>=cost(T[x-1])时，可提前结束。（并不改变复杂度）

算法二(by Jinkai)

利用上面提到的单峰性，配合分组思想，可给出 O(n log r log log r)的算法。

上面提到 ij<=2n 的元素有 O(n log r)个。将这 O(n log r)个待访问元素按列进行分组，每组 O(n)个元素，一共 O(log r)组。（注意找到 b_1~b_n 只需要 O(n log r)时间，可预处理）

利用上面单调性，通过二分找出了最优解在哪一组中（log log r 次二分，每次 O(nlogr)）。

找到包含最优解的分组后，再用 O(n)次堆操作即可在这个分组中找到最优的一个解。

算法三(by Choi and Golin) (终极解法).

Choi 和 Golin 给出了最优解的精确的结构性质（关键是 m 的取值）。这里介绍一下。

令 $h_i=(c_1+\ldots+c_i)/(i-1)$ $(1<i<=r)$。令 $h=\min_i\{h_i\}=h_k$。设 T 中节点的不同的费用值从小到达依次为 l_0 $(l_0=0),l_1(l_1=c_1),l_2,l_3,\ldots$。堆某个 j >=0。假设费用<=l_j 的节点数为 x。设 Leave{1,…,x}中费用<=l_j+h 的个数为 a_j。费用<=l_{j+1}+h 的个数为 b_j。Choi 和 Golin 证明了，

当 $a_j\le n\le b_j$ 时 m=x。当 $b_j<n<a_{j+1}$ 时 $m\in\{x+p,x+p+1\}$，其中 $p=(n-b_j)\%\ (k-1)$。

此结论刻画了 m 的取值，**避免枚举或二分 m**。此结论证明不算特别复杂，是 case by case 的，主要利用了单峰性（证明见 Algorithmica01 第 2 和 4 章）。根据上述结论，不难设计出 O(n log r)的算法（可参考 ICALP96 的 4.2 节）。它**避免了无畏的删除步骤**。

SiCOMP96= <PREFIX CODES: EQUIPROBABLE WORDS, UNEQUAL LETTER COSTS>
ICALP96=<Lopsided Trees: Analyses, Algorithms, and Applications >
Algorithmica01=<Lopsided Trees, I: Analyses1 >