本题标答做法：使用堆，时间复杂度$O(m\log K\log m)$

对于本题，为了在最朴素的$O(n\log n)$的Huffman Coding算法上进一步优化复杂度，我们可以想象，假如一开始有非常多个频率相同的叶节点，Huffman Coding肯定是把它们两两合并完，再去合并频率更大的节点。正如题面Hint提到，

> The pair (10, 10) is somehow "equivalent" to (20, 5) .

说的是10个频率为10的节点在合并五次之后，就是5个频率为20的节点。

我们需要把所有Coding Tree上结构相同的子树同时进行批量的两两合并。具体地：

> 每个节点存(f,c)（频率，次数）二元组，每次循环从堆中pop频率最小的节点(f_1,c_1)后，
>
> - 如果$c_1>1$，则需要批量的两两合并
>   - push$(2f_1,\lfloor c_1/2\rfloor)$
>   - 如果c_1是奇数的话还有个落单的，再push$(f_1,1)$
> - 如果$c_1=1$，则该节点需要和频率第二小的节点(f_2,c_2)合并一次
>   - pop(f_2,c_2)
>   - push$(f_1+f_2,1)$
>   - 如果$c_2>1$还要再push(f_2,c_2-1)回去
>
> 如此重复，直到堆里只剩一个$c=1$的点。

接下来我们还要考虑算出答案，码长之和，也就是所有叶节点的频率×高度之和。标答的做法非常简单，因为每次将两个频率最小的节点合并时，码长之和的变化恰好就等于两个子树的频率之和。具体地：

> 每个节点存(f,c)（频率，次数）二元组，每次循环从堆中pop频率最小的节点(f_1,c_1)后，
>
> - 如果$c_1>1$，则需要批量的两两合并
>   - push$(2f_1,\lfloor c_1/2\rfloor)$
>   - 码长之和 += $2f_1\times\lfloor c_1/2\rfloor$
>   - 如果c_1是奇数的话还有个落单的，再push$(f_1,1)$
> - 如果$c_1=1$，则该节点需要和频率第二小的节点(f_2,c_2)合并一次
>   - pop(f_2,c_2)
>   - push$(f_1+f_2,1)$
>   - 码长之和 += f_1+f_2
>   - 如果$c_2>1$还要再push(f_2,c_2-1)回去
>
> 如此重复，直到堆里只剩一个$c=1$的点。return 码长之和。

如果你是用其他方法，比如把树形结构建出来，然后从根节点开始DFS的过程中算答案，则会更加复杂。

接下来我们初步分析标答的时间复杂度的上界。称大点为$c>1$的节点，而小点为$c=1$的节点。

- 如果堆顶是大点，那么大点的c减半又放回堆里，并且堆里至多会多一个小点
- 如果堆顶是小点，那么会和频率第二小的节点合并一次，如果和大点合并那堆里点数不变，如果和小点合并那堆里点数还会减一。

因此，堆里总共产生的节点数至多是所有大点减半的总次数。由于一个大点减半次数不超过$\log K_i$，那么设K是K_i的上界，所有大点减半的总次数总和为$O(m\log K)$。堆里初始有m个点，这不影响我们得出堆操作次数、堆节点个数上界都是$O(m\log K)$，那么总复杂度是$O(m\log K\log(m\log K))$。

稍后我们将进一步证明堆里的节点数上界是$\Theta(m)$，因此标答的做法复杂度是$O(m\log K\log m)$。

引子：有序频率序列的Huffman Coding的$\Theta(n)$算法

这一小节我们回过来讨论一下原始Huffman Coding问题的算法。

假设频率序列已经排好序，那么我们没必要用堆的$O(n\log n)$算法，而是用下面$\Theta(n)$的算法。

使用两个queue p, q存放节点，保证每个queue内部的频率都是有序的。

初始时p为n个叶子节点，而q为空。

循环，每次从两个queue中pop频率最小的节点，再从两个queue中pop频率最小的节点，然后合并push到q的最后。显然Huffman Coding过程中合并后的频率是单调不降的，所以放在q最后仍然有序。

如此重复，直到p为空而q只剩一个节点。

换句话说，即使初始频率序列没排好序，我们也可以不用堆，而是排个序之后再用这个算法，复杂度一样$O(n\log n)$。

本题改进做法：不用堆，时间复杂度$O(m\log m + m\log K)$

利用了引子的思路，先对初始频率序列排序，复杂度$O(m\log m)$。

之后也是两个队列p, q，但是q要用双端队列

> 每个节点存(f, c, s)（频率，次数，码长之和）三元组，初始所有点的$s=0$。每次循环从两个队列中pop频率最小的节点(f_1, c_1, s_1)后，
>
> - 如果$c_1 > 1$，则需要批量的两两合并
>   - q.push_back$(2f_1, c_1/2, 2s_1 + 2f_1)$
>   - 如果c_1是奇数的话还有个落单的，再q.push_front$(f_1, 1, s_1)$
> - 如果$c_1 = 1$，则该节点需要和频率第二小的节点(f_2, c_2, s_2)合并一次
>   - 从现在两个队列中选出频率最小的pop(f_2, c_2, s_2)
>   - q.push_back$(f_1 + f_2, 1, s_1 + s_2 + f_1 + f_2)$
>   - 如果$c_2 > 1$还要再q.push_front$(f_2, c_2 - 1, s_2)$回去
>
> 如此重复，直到p为空而q里只剩一个$c=1$的点，它的s就是答案。

因为队列的单次操作是$\Theta(1)$，所以这个循环的复杂度是$O(m\log K)$。

因此改进算法总时间复杂度$O(m\log m + m\log K)$，是对标答渐进意义上的严格改进。

利用改进做法的过程，证明待合并节点数上界为$\Theta(m)$

设q的频率序列为$f_1, f_2, \ldots, f_l$，该序列单调不降。

并且有的点是大点，用$\left[.\right]$表示；有的点是小点，用$(.)$表示，像这样：$\left[f_1\right](f_2)\ldots\left[f_l\right]$

我们把q想象成一个首尾相接的环：$\ldots\left[f_l\right]\Big\|\left[f_1\right](f_2)\ldots$，用双竖线表示队首和队尾的分界。

定义势能Φ为这个首尾相接的环里，相邻两个点都是大点的对数。上面f_l, f_1也算相邻大点。

接下来我们关注各种可能的情况对势能Φ和q的长度l的影响：

- f_1是大点，f_2是大点时，无论f_l是不是大点都有

$$\cdots \Bigg\| \Bigg[f_1 \Bigg] \Bigg[f_2 \Bigg] \cdots \begin{cases} c_1\text{为偶数，} \cdots \Bigg[2f_1\Bigg] \Bigg\| \Bigg[f_2\Bigg] \cdots & \Phi\text{不变，}l\text{不变，}\Phi+l\text{不变} \\ c_1\text{为奇数，}\cdots \Bigg[2f_1\Bigg]\Bigg\|(f_1)\Bigg[f_2\Bigg]\cdots & \Phi\text{减}1\text{，}l\text{加}1\text{，}\Phi+l\text{不变}\end{cases}$$

- f_1是大点，f_2是小点时，无论f_l是不是大点都有

$$\cdots \Bigg\| \Bigg[f_1 \Bigg] (f_2) \cdots \begin{cases} c_1\text{为偶数，} \cdots \Bigg[2f_1\Bigg] \Bigg\| (f_2) \cdots & \Phi\text{不变，}l\text{不变，}\Phi+l\text{不变} \\ c_1\text{为奇数，}\cdots \Bigg[2f_1\Bigg]\Bigg\|(f_1)(f_2)\cdots \Longrightarrow \cdots \Bigg[2f_1\Bigg](f_1+f_2)\Bigg\|\cdots & \Phi\text{不变，}l\text{不变，}\Phi+l\text{不变}\end{cases}$$

- f_1是小点，f_2是大点时，无论f_l是不是大点都有

$$\cdots \Bigg\| (f_1) \Bigg[f_2 \Bigg] \cdots \Longrightarrow \cdots (f_1+f_2) \Bigg\| \Bigg[f_2 \Bigg] \cdots \quad \Phi\text{不变，}l\text{不变，}\Phi+l\text{不变}$$

- f_1是小点，f_2是小点时，无论f_l是不是大点都有

$$\cdots \Bigg\| (f_1)(f_2) \cdots \Longrightarrow \cdots (f_1+f_2) \Bigg\| \cdots \quad \Phi\text{不变，}l\text{减}1\text{，}\Phi+l\text{减}1$$

上面的所有操作都不会让$\Phi+l$变大！

我们虽然没有讨论合并时c减小导致大点变成小点的情况，但是显然大点变成小点只可能让Φ更小或者不变。

我们只剩下一个地方要考虑了，就是当p的队首频率（记为p_1）比f_1小的时候。这种情况我们可以理解为，把大点$\Bigg[p_1\Bigg]$插进q的队首，则Φ至多加1（当f_l, f_1其中有一个大点的时候），l加1，那么$\Phi+l$至多加2。

因为至多从p往q插入m次，那么$\Phi+l$最大可以达到$2m$，因此$l \leq 2m$。需要注意的是f_1是大点，f_2是小点且c_1为奇数时，中间$\cdots\Bigg[2f_1\Bigg]\Bigg\|(f_1)(f_2)\cdots$的时刻，队列q的长度会临时多1，因此队列q的长度在整个改进算法过程中能到达的极限是$2m+1$，这不影响我们证明待合并节点数上界为$\Theta(m)$。

因为标答堆里的节点和改进算法p, q两个队列里的节点合起来在整个算法过程中是一一对应关系，都表示待合并节点的集合，所以我们也证明了堆里的节点数上界是$\Theta(m)$，因此标答的做法时间复杂度是$O(m \log K \log m)$。

同时我们也证明了两种算法的空间复杂度都可以做到$\Theta(m)$。

$\Theta(n)$算法中发现的一个性质，但最终没有帮助我们的证明

性质：设q的频率序列为$f_1, f_2, \ldots, f_l$，该序列单调不降，则整个算法过程中始终有$2f_1 \geq f_l$。

证明：设p的最小的两个频率为p_1, p_2，合并的时候有三种情况

- 合并p_1, p_2，那么说明$p_1 \leq p_2 \leq f_1$，则合并后队首、队尾分别为f_1, p_1+p_2，满足$2f_1 \geq p_1+p_2$；

- 合并f_1, f_2，那么说明$f_1 \le f_2 \le p_1$，则合并后队首、队尾分别为$f_3, f_1 + f_2$，满足$2f_3 \ge f_1 + f_2$；
- 合并p_1, f_1，那么说明$p_1, f_1 \le p_2, f_2$，则合并后队首、队尾分别为$p_2, p_1 + f_1$，满足$2p_2 \ge p_1 + f_1$。

边界情况是q中只剩一个点时，也满足性质$2f_1 \ge f_1$。

不知道这个性质能不能推导出其它有价值的结论，欢迎大家讨论。