1.

可以使用基数排序

根据基数排序的定理，若有n个数，整数的2进制数有b个bit的话，可以将一个数拆成 $ceil(b/r)$个 r bit 的数，则 $T(n,b)=\Theta(\frac{b}{r}(n+2^r))$

当取值范围到n^3 时，则$b=\Theta(logn^3)=\Theta(3logn)$

当取$r=logn$ 时， $T(n,b)=\Theta(\frac{3logn}{logn}(n+2^{logn}))=\Theta(6n)=\Theta(n)$

或者说，当范围到n^3 时，一个数可以表示成 an^2+bn+c $(a,b,c<n)$, 可以按照c,b,a 的顺序，对这n个数进行计数排序，每次复杂度为$\Theta(n+n)=\Theta(n)$，因此三次排序后复杂度仍为$\Theta(n)$

2.

a.这个序列是一个单调不减序列

b. 1 2 3 4 5 6 7 8 10 9

c. 由定义：$\frac{\sum_{j=i}^{i+k-1}A[j]}{k}\le\frac{\sum_{j=i+1}^{i+k}A[j]}{k}$ $(i=1,2...n-k)$

则$\sum_{j=i}^{i+k-1}A[j]\le\sum_{j=i+1}^{i+k}A[j]$， 化简得$A[i]=A[i+k]$ $(i=1,2...n-k)$

d. 可以将n个数分成k个集合，第i个集合为$A[i],A[k+i],A[2k+i]....$

则每个集合有n/k 个数，使一个集合有序需要$O(\frac{n}{k}lg(n/k))$, 则k个集合需要$O(nlog(n/k))$

e.当一个序列是k-sort的时候，则可以看成这个序列分成了k个有序的序列，将k个有序的序列归并，可以建立一个k大小的堆，堆里面是每个有序序列最小的一个，每一个数需要$O(logk)$ 的时间，则总复杂度为$O(nlogk)$

f.运用决策树，相当于获得k个大小不超过$ceil(n/k)$ 的序列，每个序列有（ $n/k)!$ 个节点，总共有 k（ $n/k)!$个节点

设树的高度为h，则k（ $n/k)!\le 2^h$， 则$h\ge logk+n/klog(n/k)=logk+n/k(logn-logk)$

则$h=\Omega(logk+n/k(logn-logk))$， 当k为常数，$h=\Omega(n(logn-logk))$ = $h=\Omega(nlogn)$

3.

将题意的Weighted median理解成带权中位数(较小中位数)， median理解成中位数(较小中位数)

a.设较小中位数为x_k, 则$k=(n+1)/2$,(/ 理解为整除)

则$\sum_{x_i<x_k}w_i=(n-1)/2*1/n<1/2$, $\sum_{x_i>x_k}w_i=(n-\frac{n+1}{2})*\frac{1}{n}\le 2/n*n/1=1/2$

b. 用$O(nlogn)$ 的时间先按照xi的值排序，然后从第一个元素的w 值进行累加，使累加值第一次超过或等于1/2的xi 为 Weighted median

因为它是第一个让累加值超过或等于1/2的，所以小于它的元素加起来肯定小于1/2，而之后的元素加起来必然小于或者等于1/2

c.若是有线性时间确定中位数的算法，则可以进行二分，一开始$L=1$，$R=n$, 每次查找下标在$[L，R]$ 中数的中位数

若是小于它的x的w累加值小于1/2 并且 大于它的x的w的累加值小于等于x，则说明找到了带权中位数

若是前者的累加值是大于等于1/2的，则令R = xi， 若是后者的累加值大于1/2,则L =xi

这样的话每次查找需要线性的复杂度，其递推表达式是$T(n) = T(n/2) + \Theta(n)$

经过迭代，$T(n) = \Theta(n + n/2 + n/4 + ...) = \Theta(2n) = \Theta(n)$

d.设Weighted median为p_k, 设将邮局设在x处的累加值为f(x)

假设y不等于p_k, 若$y > p_k$:

则对于$p_i > y, \sum w_i * d(p_k, p_i) = \sum w_i(p_i - p_k)$, 同样地$\sum w_i * d(y, p_i) = \sum w_i(p_i - y)$

而对于$p_k < p_i \le y, \sum w_i * d(p_k, p_i) = \sum w_i(p_i - p_k)$， 而 $\sum w_i * d(y, p_i) = \sum w_i|p_i - y| \ge \sum w_i(p_i - y)$

所以对于$p_k < p_i$，$f(p_k) - f(y) \le \sum w_i(p_i - p_k - pi + y) = \sum w_i(y - p_k)$

对于$p_i < p_k, \sum w_i * d(p_k, p_i) = \sum w_i(p_k - p_i)$, 同样地$\sum w_i * d(y, p_i) = \sum w_i(y - p_i)$

$f(p_k) - f(y) \le \sum w_i(p_k - p_i - y + p_i) - (y - p_k) + \sum w_i(y - p_k) = (y - p_k)(W_2 - w_k - W1)$

其中$W2$，$W1$分别表示大于和小于p_k 的x_i 的w累加值，根据定义可知$f(p_k) - f(y) \le 0$

同理当$y < p_k$, 也有$f(p_k) - f(y) \le 0$

所以Weighted median为最小值点

e.可以先将所有横坐标排序，找到以横坐标为序列的weighted median为x_m， 然后类似处理纵坐标得到y_m，则邮局建在(x_m, y_m), 因为距离是曼哈顿距离，将横纵坐标分开计算这个点都是最优的，所以合起来一定也是最优的

4.

(1)使用两个指针，分别指向A，B数组的起始元素，每次找到这两个指针中最小的那一个元素并移动指针，若m+n为偶数，则找到第(m+n)/2和(m+n)/2+1个元素取平均，若是m+n为奇数，则找到第(m+n+1)/2个元素

这样复杂度为$\Theta((m + n)/2) = \Theta(m + n)$

(2)对于数组A,B，每次花$\Theta(1)$寻找这两个数组的中位数a, b, 若是a = b, 则为答案

若是a < b, 则 a < =ans <= b, 可以将A数组小于a的元素删去，B数组大于b数组的元素删去(保证删去的元素要一样)，再次寻找；

若是a > b, 则b < =ans <= a, 可以将A数组大于a的元素删去，B数组小于b数组的元素删去(保证删去的元素要一样)，再次寻找；

由于每次寻找都会剪去大概一半的元素，则递归表达式$T(n) = T(n/2) + \Theta(1)$, 递归迭代得$T(n) = \Theta(logn)$

(3)当m = n 时，由(2)可得答案

当m≠n时，不妨设 m > n, 若 m + n 为偶数，则 m - n 也为偶数，可以剪去A数组开头的(m - n) / 2 个元素，末尾的（m - n) / 2个元素，这样剪后n个元素的A数组和n个元素的B数组的中位数也一定是整体的中位数

若是m + n为奇数，则m - n 也是奇数，A数组开头和结尾分别剪去(m - n - 1)/2个元素，这样A数组还剩下(n+1)个元素，A,B数组总共有奇数个元素，则中位数肯定是某个数x_i，若是再加上一个负无穷大的元素，则中位数肯定往右偏，成为最中间两个元素x_i, x_{i+1}的平均值，用二分查找$O(logn)$的复杂度来找到大于等于这个平均值的最大数x_{i+1}, 即为所求

以上讨论的复杂度为$O(logn)$

同理，当 m < n, 复杂度为$O(logm)$

具体代码实现在附件中（已过EOJ）