一.偏序

证: $\forall x, y, z \in \mathrm{N}$

1. $x|x$ 显然成立(因为$x = x \cdot 1$).
2. 如果 $x|y$ 且 $y|x$,则设 $y = c \cdot x, x = d \cdot y(c, d \in \mathrm{N})$,因而有 $x = c \cdot d \cdot x$,因而有 $c \cdot d = 1$,因此 $c = d = 1$。由此，我们可知 $x = y$
3. 如果 $x|y$ 且 $y|z$,则设 $y = c \cdot x, z = d \cdot y(c, d \in \mathrm{N})$。此时，$z = c \cdot d \cdot x$,而 $c, d \in \mathrm{N}$,因而 $c \cdot d \in |\mathrm{N}$,因而有$x|z$。

由以上三条性质，可以知道自然数上整除是偏序。

二.对角线方法

证: case 0: 当A的元素个数有限，设为 n 个。 此时，$|A| = n, |pow(A)| = 2^n$ 当 $n = 0$, $|A| = 0 < 1 = |pow(A)|$ 当 $n \in \mathrm{N}, 2^n \geq n + 1$(由二项式展开) $> n$，所以$|A| < |pow(A)|$。

case 1: 当A的元素个数无限且为可列集。 此时，若存在由 A 到 $pow(A)$ 的双射，记为 f 。设 A 中的元素的一个排列为 $\{a_n\}$，记 $f(a_i) = B_i$ 。 设 $b_{i,j} = 1$(当B_i中含有a_i)或 0 (当B_i中不含有a_i) 。 考虑集合 C : 若 $b_{i,i} = 1$ 则 C 中不含有 a_i 。反之，则 C 中含有 a_i 。此时，C与B_i在a_i元素上不同 ，因此 $!\exists B_i\ s.t.\ C = B_i$ ，所以 $C \subseteq A$且$C \subseteq pow(A)$ ，与 $pow(A)$ 定义矛盾。 因此假设不成立，$|A| = |pow(A)|$

case 2:当 A 的元素无限且为不可列集。 同理若存在，则设 f 是一个满足的双射。 此时，构造集合 $B = \{x | x \in A, x \in f(x)\}$。$\forall C \subseteq A$，由定义 $\exists y = f^{-1}(C) \in A$。 此时，若 $y \in B$ 则 $y \in f(y) = C$ ，若 $y \in B$，则 $y \in f(y) = C$。所以 B和C 在元素 y 上不同，$B = C$。因此 $!\exists C \in pow(A)\ s.t.\ C = B \subseteq A$ ，与 $pow(A)$ 定义矛盾。 因此假设不成立，$|A| = |pow(A)|$

三.基于比较的排序

(1).证: 由数分课上讲过的斯特林公式

$$\lim_{n \to \infty} \frac{n!}{\sqrt{2\pi n}(\frac{n}{e})^n} = 1$$

$$\text{可知} \lim_{n \to \infty} \frac{\log(n!)}{\log(\sqrt{2\pi n}(\frac{n}{e})^n)} = 1$$

所以 $\log(n!) = \Theta(n\log n - n + 0.5\log 2\pi n) = \Theta(n\log n)$

补:证明方法2(放缩法) 我们默认log以e为底数

$$\int_1^n \log x dx \le \log(n!) \le 1 + \int_1^{n+1} \log x dx$$

$$\text{其中} \int_1^n \log n = n\log n - n - 1$$

因此 $\log(n!) = \Theta(n\log n)$

(2). 解:It's trivial that thy answer is $n!$.

(3). 证: 先定义 a < b 为 a 排在 b 前面。 在最坏情况下，假设你比较两个数字a和b， 若a,b的相对大小已经确定，那么你不能消除任何一种情况；若a,b的相对大小未确定，那么将目前可能的排列的集合 X 分为a排在b前面 和 b 排在 a前面两类排列的集合 Y，Z，有 $|Y| + |Z| = |X|$，所以$\max\{|Y|, |Z|\} >= 0.5|X|$。因此，无论怎么设计算法选取a,b，每次比较后最坏的情况是可能的排列数量减小为原来的 0.5 倍。所以最坏情况下，最优算法至少要比较 t 次 其中 $2^t \ge n!(\text{排列数})$ 所以 $t \ge \Omega(\log_2(n!)) = \Omega(n\log n)$。

下证明该最坏情况下最优可以取到等号。 考虑最简单的归并排序，如果你还不会，请看这个or 附 C++ 代码(那你不太行啊)

```
// C++ version
void merge(int l, int r) {
    if (r - l <= 1) return;
    int mid = l + ((r - l) >> 1);
    merge(l, mid), merge(mid, r);
    for (int i = l, j = mid, k = l; k < r; ++k) {
        if (j == r || (i < mid && a[i] <= a[j])) tmp[k] = a[i++];
    else tmp[k] = a[j++];
  }
  for (int i = l; i < r; ++i) a[i] = tmp[i];
}
```

过程如下:将一个数列长度补全到 2^N(补无穷小)，再进行如下操作。先将原数列划分为 2^{N-1} 个长度为2的子区间，将每个子区间内比较一次完成排序。再将原数列划分为 2^{N-2} 个长度为 4 的子区间，每个长度为4的子区间由两个已经排序好的、长度为2的子区间合并而来。之后类似的合并长度为4的子区间，直到所有区间合并为长 2^N 的数列再取出无穷小即为排序完成的数列。

在合并任意两个排序完的、长度为n和m的区间的时候，我们反复比较这两个区间的排在前面的最小的元素，将更小的一个取出区间并放到临时数列中。因为已经排序完，所以取出的元素就是两个区间的最小的元素，仅需1次比较，转变为长度为 n - 1 和 m 或 n 和 m - 1 的区间。重复操作，我们最多经过 n + m - 1 次的过程，就可以将两个区间合并为排序完的、长 n + m 的区间。

因此，每次合并长 2^n 的区间的比较次数不会超过 $2^{N-n-1} \cdot 2 \cdot 2^n = 2^N$次。

合并过程最多执行N次，所以比较操作不会超过 $N \cdot 2^N$ 次。我们可以补为 $2^{N-1} <$ 数列长度 $\leq 2^N$ 的数列。此时，比较次数不超过 $2 \cdot n \cdot \log_2(n) = O(n \log n)$次。所以 $t \geq \Omega(n \log n)$ 等号可以取到。

综上，基于比较的排序需要的比较次数是 $\Omega(n \log n)$的。

四.密码学

1.RSA加密算法

(1). 解: 公钥: p * q = 3127 和 e = 11 私钥: p * q = 3127 和 d = 1371(exgcd的解)

(2). 解: ALEI

(3). 解: 不可以。为了保证 ex + (p-1)(q-1) y = 1 有解，需要e和(p-1)(q-1)(即公钥的欧拉函数的值)互质。13和3016不互质，所以不行。

(4). 解: 因为还原出解的过程是在 mod p * q意义下进行的，所以最终解密出的密文是 实际文字的数值 在 mod p * q 意义下的值。为了保证在mod p * q意义下信息不会改边，需要保证输入原始的密文的最大值不超过 p * q，即为题目所描述的。

五.复杂度

1.搜索 SAT 的解

假设是一个n-SAT问题，则变元个数 $m \leq n \cdot |x|$ ，设为 $\{a_m\}$。对于已有的 $|x|$ 个约束，我们对其额外的添加约束。首先添加单约束 $a_1 = 0$，并对新的约束，用 M 加以验证。如果新的约束不可能满足，那么因为有解性，所以 $a_1 = 1$ 时必定有解，保留单约束$a_1 = 1$；如果新的约束能够满足，那就保留单约束 $a_1 = 0$，而有解性不会丧失 。然后类似的，从小到大添加 $a_2, a_3, ..., a_m$ 的单约束并用 M 的结果决定保留的单约束。最

终，只需调用 $m = O(|x|)$ 次 M ，即可得到一组可行的解。(每个变元的值即为新增的 m 个单约束中这个变元保留的单约束的值)。