1.$T(n) = n + \sum_{i=1}^{n-1} T(i), T(1) = 1$

当$n \geq 3$，$T(n-1) = n - 1 + \sum_{i=1}^{n-2} T(i)$, 则$T(n) = 2 * T(n-1) + 1$

对于此递推式其特征方程为$x^2 = 2 * x$， 解得$x = 2$（除去零解）

由 $T(3) = 7, T(4) = 15$，配得系数$T(n) = 2^n - 1$ $(n \geq 3)$

而$T(1) = 1, T(2) = 3$, 也满足上式

所以$T(n) = 2^n - 1$

4.1

$a. T(n) = 2T(n/2) + n^4 = \Theta(n^4)$

由主定理， $n^{log_b a} = n < n^4$, 且对于某个常数c, 一定存在足够大的n，使得$2 * (n/2)^4 < c * n^4$

$b. T(n) = T(7n/10) + n = \Theta(n)$

由主定理， $n^{log_b a} = 1 < n$, 且对于某个常数c, 一定存在足够大的n，使得$7n/10 < c * n$

$c. T(n) = 16T(n/4) + n^2 = \Theta(n^2 logn)$

由主定理， $n^{log_b a} = n^2 = n^2$， 则$T(n) = \Theta(n^2 logn)$

$d. T(n) = 7T(n/3) + n^2 = \Theta(n^2)$

由主定理， $n^{log_b a} = n^{log_3 7} < n^2$, 且对于某个常数c, 一定存在足够大的n，使得$7(n/3)^2 < c * n^2$

$e. T(n) = 7T(n/2) + n^2 = \Theta(n^{log_2 7})$

由主定理， $n^{log_b a} = n^{log_2 7} > n^2$，则$T(n) = \Theta(n^{log_2 7})$

$f. T(n) = 2T(n/4) + \sqrt{n} = \Theta(\sqrt{n} logn)$

由主定理， $n^{log_b a} = n^{1/2} = \sqrt{n}$， 则$T(n) = \Theta(\sqrt{n} logn)$

$f. T(n) = T(n-2) + n^2 = \Theta(n^3)$

当$n = 2k, T(n) = T(2k) = T(2(k-1)) + (2k)^2 = T(2(k-2)) + (2(k-1))^2 + (2k)^2$

$= ... = 4(1^2 + 2^2 + 3^2 + .. + k^2) = 4 * \frac{k(k+1)(2k+1)}{6} = \Theta(k^3) = \Theta(n^3)$

同样可以得出$n = 2k + 1$的情况

所以$T(n) = \Theta(n^3)$

4.3

$a. T(n) = 4T(n/3) + nlgn = \Theta(n^{lg_3 4})$

由主定理，$n^{log_b a} = n^{log_3 4} > nlgn(polynomially\ large)$, 则$T(n) = \Theta(n^{lg_3 4})$

$b. T(n) = 3T(n/3) + n/lgn = \Theta(nlglgn)$

记$H(k) = \sum_{i=1}^{k} 1/i$, 则$H(k) = \Theta(lgk)$, 得到 $H(\lceil lgk \rceil) = H(\lfloor lgk \rfloor) = \Theta(lglgk)$

将原式迭代，得到$T(n) = n(\frac{1}{lgn} + \frac{1}{lgn-lg3} + \frac{1}{lgn-2*lg3} + ... + \frac{1}{c})$(c为某个足够小的整数)

记$G(n) = \frac{1}{lgn} + \frac{1}{lgn-lg3} + \frac{1}{lgn-2*lg3} + ... + \frac{1}{c}$, 则 $T(n) = n * G(n)$

令$lgn = t$，$G(n) < \int_{c-1}^{t} \frac{1}{i} = lnt - ln(c-1) < lnt = lglgn/lge$

又$G(n) > \int_{c}^{t+1} \frac{1}{i} = ln(t+1) - ln(c) = lg(lgn+1)/lge - ln(c-1)$

可以得到 $G(n) = \Theta(lglgn)$，$T(n) = \Theta(nlglgn)$

$c. T(n) = 4T(n/2) + n^2\sqrt{n} = \Theta(n^2\sqrt{n})$

由主定理，$n^{log_b a} = n^{log_2 4} < n^2\sqrt{n}(polynomially\ small)$，且$4(n/2)^2\sqrt{n/2} < n^2\sqrt{n}$, 则 $T(n) = \Theta(n^2\sqrt{n})$

$d. T(n) = 3T(n/3 - 2) + n/2 = O(nlgn)$

由递归树$T(n) = n/2 + (n-6)/2 + ...$，且层数小于$lg_3 n$层，则$T(n) = O(nlgn)$

$e. T(n) = 2T(n/2) + n/lgn = \Theta(nlglgn)$

记$H(k) = \sum_{i=1}^{k} 1/i$, 则$H(k) = \Theta(lgk)$, 得到 $H(\lceil lgk \rceil) = H(\lfloor lgk \rfloor) = \Theta(lglgk)$

将原式迭代，得到$T(n) = n(\frac{1}{lgn} + \frac{1}{lgn-1} + \frac{1}{lgn-2} + ... + \frac{1}{c})$(c为某个足够小的整数)

记$G(n) = \frac{1}{lgn} + \frac{1}{lgn-1} + \frac{1}{lgn-2} + ... + \frac{1}{c}$, 则 $T(n) = n * G(n)$

令$lgn = t$，$G(n) < \int_{c-1}^{t} \frac{1}{i} = lnt - ln(c-1) < lnt = lglgn/lge$

又$G(n) > \int_{c}^{t+1} \frac{1}{i} = ln(t+1) - ln(c) = lg(lgn+1)/lge - ln(c-1)$

可以得到 $G(n) = \Theta(lglgn)$，$T(n) = \Theta(nlglgn)$

$f. T(n) = T(n/2) + T(n/4) + T(n/8) + n = \Theta(n)$

假设当n < k时，对于某个c, $T(n) < cn$对充分大的n成立，则$n = k$时，
$T(n) < cn/2 + cn/4 + cn/8 + n = (1 + 7c/8)n$， 则$T(n) = O(n)$

而$T(n) = T(n/2) + T(n/4) + T(n/8) + n > n$, 得$T(n) = \Omega(n)$

由上可得$T(n) = \Theta(n)$

$g. T(n) = T(n-1) + 1/n = \Theta(lgn)$

由$\sum_{i=1}^{n} 1/i = \Theta(lgn)$, 经过迭代，$T(n) = 1/n + 1/(n-1) + ... + 1 = \Theta(lgn)$

$h. T(n) = T(n-1) + 1/lgn = \Theta(nlgn)$

假设当n < k时，存在c, 当n充分大时，$T(n) < cnlgn$, 则当n = k 时，
$T(n) < c(n-1)lg(n-1) + lgn = cnlg(n-1) - clgn + c + lgn < cnlgn - clgn + c + lgn$

当$-clgn + c + lgn < 0$时，即$lgn > c/(c-1)$时，$T(n) < cnlgn$，$T(n) = O(nlgn)$

假设当n < k时，存在c, 当n充分大时，$T(n) > cnlgn + dn$, 则当n = k 时，
$T(n) > c(n-1)lg(n-1) + d(n-1) + lgn = cnlg(n-1) - clg(n-1) + dn - d + lgn$

$> cnlg(n/2) - clg(n-1) + dn - d + lgn = cnlgn - cn - clg(n-1) + dn - d + lgn$

$= cnlgn - cn - clg(n-1) + dn - d$

当$-cn - clg(n-1) + dn - d > 0$, 即$(d-c)n > (c-1)lg(n-1) + d$时

有$T(n) > cnlgn, T(n) = \Omega(nlgn)$

所以)$T(n) = \Theta(nlgn)$

$i. T(n) = T(n-2) + 2lgn = \Theta(nlgn)$

经过迭代，$T(n) = 2lgn + 2lg(n-2) + ... < n/2 * 2lgn = nlgn$, 则$T(n) = O(nlgn)$

假设$n < k$时， 存在c, 当n充分大时, 有$T(n) < cnlgn$, 则$n = k$ 时，
$T(n) < c(n-2)lg(n-2) + 2lgn < c(n-2)lgn + 2lgn = (cn - 2c + 2)lgn$, 当$c > 1$,

$T(n) < cnlgn$， 可得$T(n) = \Omega(nlgn)$

所以$T(n) = \Theta(nlgn)$

$j. T(n) = \sqrt{n}T(\sqrt{n}) + n = \Theta(n)$

假设n < k时，存在c, 当n足够大时，有$T(n) < cn$, 则当$n = k$时, 有
$T(n) < \sqrt{n}\sqrt{cn} + n = (\sqrt{c} + 1)n$, 可得$T(n) = O(n)$

而$T(n) = \sqrt{n}T(\sqrt{n}) + n > n$, 可得$T(n) = \Omega(n)$

所以$T(n) = \Theta(n)$

2.4

a.$(1,5),(2,5),(3,4),(3,5),(4,5)$

b.当为{n, n - 1, ... 1}时，任意两个数都是逆序对，为最多， 有$n(n-1)/2$个

c. 逆序对的个数越小，插入排序的交换次数(相对而言，并非完全线性), 若一个数不在任何逆序对中，则它左边没有比它小的元素，右边没有比它大的元素，则这个元素是不需要交换的，每次交换后可以保证被交换的到正确位置的元素不在逆序对中。

d.

归并排序的复杂度为$O(nlgn)$, 在归并排序的过程中，若是后一个数组的数被归并到，则前一个数组剩余的数都比这个数大，都可以和这个数构成逆序对，所以逆序对可以在这个过程中求出。

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;
typedef long long ll;
const int maxn = 1e6 + 3;
const int INF = 0x3f3f3f3f;
const int mod = 1e9 + 7;

int n, num[maxn], A[maxn], B[maxn];
ll cnt;

void merge_sort(int l, int r, int* A, int* B)
{
    if(l == r)
    {
        A[l] = num[l];
        return;
    }
    int mid = (l + r) / 2, p1 = l, p2 = mid + 1, i = l;
    merge_sort(l, mid, B, A);
    merge_sort(mid + 1, r, B, A);
    while(p1 <= mid && p2 <= r)
    {
        if(B[p1] <= B[p2]) A[i++] = B[p1++];
        else
        {
            A[i++] = B[p2++];
            cnt += mid - p1 + 1;
        }
    }
    while(p1 <= mid) A[i++] = B[p1++];
    while(p2 <= r) A[i++] = B[p2++];
}

int main()
{
    scanf("%d", &n);
    for(int i = 1; i <= n; i++)
        scanf("%d", &num[i]);

    merge_sort(1, n, A, B);
    printf("%lld\n", cnt);
}
```