分治

分治法思想

将一个难以直接解决的大问题，分割成一些规模较小的相同问题。

分治策略

对于一个规模为n的问题，若该问题可以容易地解决（比如说规模n较小）则直接解决，否则将其分解为k个规模较小的子问题，这些子问题互相独立且与原问题形式相同，递归地解这些子问题，然后将各子问题的解合并得到原问题的解。这种算法设计策略叫做分治法。

分治法适用的情况

1. 该问题的规模缩小到一定的程度就可以容易地解决
2. 该问题可以分解为若干个规模较小的相同问题，即该问题具有最优子结构性质。
3. 利用该问题分解出的子问题的解可以合并为该问题的解；
4. 该问题所分解出的各个子问题是相互独立的，即子问题之间不包含公共的子子问题

分治之二分

每次查找都将求解规模缩小了一半

必须是有序的数组

模板

```c
int binary_search(int *num, int n, int x) {
    int l = 0, r = n - 1, mid;
    while (l <= r) {
        mid = (l + r) >> 1;
        if (num[mid] == x) return mid;
        if (num[mid] > x) r = mid - 1;
        else l = mid + 1;
    }
    return -1;
}
```

延伸问题

- 最小值最大化
- 最大值最小化

练习题

P2440 木材加工

解题思路

最小值最大

代码

```c
#include<iostream>
using namespace std;

int arr[100005] = {0};

int main() {
    int n, k, l = 1, r = 0;
    cin >> n >> k;
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
        r = max(r, arr[i]);
    }
    while (l <= r) {
        int mid = (l + r) >> 1, cnt = 0;
        for (int i = 0; i < n; i++) cnt += arr[i] / mid;
        if (cnt >= k) l = mid + 1;
```

```
        else r = mid - 1;
    }
    if (r < 1) r = 0;
    cout << r << endl;
    return 0;
}
```

P1226 【模板】快速幂||取余运算

代码

```
#include <iostream>
#include <cstdio>
using namespace std;

typedef long long LL;

LL quick_pow(LL b, LL p, LL k) {
    LL ret = 1;
    while (p) {
        if (p & 1) ret = ret * b % k;
        b = b * b % k;
        p >>= 1;
    }
    return ret % k;
}

int main() {
    LL b, p, k;
    cin >> b >> p >> k;
    printf("%lld^%lld mod %lld=%lld\n", b, p, k, quick_pow(b, p, k));
    return 0;
}
```

P1908 逆序对

解题思路

归并排序原理

原数组 arr，从数组左半部分取出一个元素a_i，从数组右半部分取出一个元素a_j，如果 a_i <= a_j，则将 a_i 放到数组 temp 中，依次类推，再将数组 temp 拷贝回数组的相应位置。

解题方法

采用归并排序，可以利用分治的思想，将数组分为左右两部分，因为我们采用归并排序，所以左右两部分都是相对有序的，例如：

left	right
5, 6, 7, 8	1, 2, 3, 4

数组中 1 比左数组中所有元素都小，所以把 1 放入 temp 数组，此时左数组中所有元素都可以和 1 组成逆序对，所以逆序对总数加上左数组中没有放入 temp 数组中元素的个数。

代码

```
#include <iostream>
#include <cstdio>
using namespace std;

typedef long long LL;
const int LEN = 500010;

int arr[LEN] = {0}, temp[LEN] = {0};
LL ans = 0;

// 快速读入
inline int read() {
    int x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') { if (ch == '-') f = -1; ch = getchar(); }
    while (ch >= '0' && ch <= '9') { x = x * 10 + (ch - '0'); ch = getchar(); }
    return x * f;
}

// 归并排序
void merge_sort(int l, int r) {
    if (r - l < 2) return ;
```

```
    int mid = (l + r) / 2;
    merge_sort(l, mid);
    merge_sort(mid, r);
    int i = l, j = mid, k = 0;
    while (i < mid || j < r) {
        if (j >= r || (i < mid && arr[i] <= arr[j])) {
            temp[k++] = arr[i++];
        } else {
            temp[k++] = arr[j++];
            // 此时左侧没有放入 temp 数组中的元素都大于 arr[j]
            // 所以逆序对的总数加上这些元素的个数
            ans = ans + (mid - i);
        }
    }
    for (i = l; i < r; ++i) arr[i] = temp[i - l];
    return ;
}

int main() {
    int n = read();
    for (int i = 0; i < n; ++i) arr[i] = read();
    merge_sort(0, n);
    printf("%lld\n", ans);
    return 0;
}
```

P1010 幂次方

解题思想

将一个数字表示为 `2(0) 2 2(2)` 加和嵌套的形式，我们只需用递归一直拆分即可

代码

```
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

const int LEN = 16;
int arr_pow[20] = {0};
string arr_ch[3] = {"2(0)", "2", "2(2)"};
string ans;

// 将 2^0 ~ 2^16 打表
```

```cpp
void init() {
    arr_pow[0] = 1;
    for (int i = 1; i <= LEN; i++) {
        arr_pow[i] = arr_pow[i - 1] * 2;
    }
    return ;
}

// 递归判断 n 该如何展开
void dfs(int n, int k) {
    int now = floor(log2(n)); // now 最大等于 2 的几次幂
    if (now == 2) ans += arr_ch[2];
    else if (now == 1) ans += arr_ch[1];
    else if (now == 0) ans += arr_ch[0];
    else {
        // now 还需要继续展开
        ans += "2(";
        dfs(now, now);
        ans += ')';
    }

    // 展开 n - arr_pow[now] 的剩余部分
    if (n - arr_pow[now] > 0) {
        ans += '+';
        dfs(n - arr_pow[now], now);
    }
    return ;
}

int main() {
    init();
    int n;
    cin >> n;
    dfs(n, LEN);
    cout << ans << endl;
    return 0;
}
```