三分

/3的均摊复杂度最低, /2的漏点的概率小

规定递归层数的稳定, 反之动态时间复杂度, 但可能精度误差大

浮点数三分

写法1:

```
while (l <= r) {
    double lmid = (l + r) / 2, rmid = (lmid + r) / 2;
    double lck = calc(lmid), rck = calc(rmid);
    ans = min({ans, lck, rck});
    if (lck < rck) r = rmid - eps;
    else l = lmid + eps;
}
```

写法2:

```
for (int _ = 100; _; _--) {
    double lmid = (l + r) / 2, rmid = (lmid + r) / 2;
    double lck = calc(lmid), rck = calc(rmid);
    ans = min({ans, lck, rck});
    if (lck < rck) r = rmid;
    else l = lmid;
}
```

写法3:

```
while (l <= r) {
    double lmid = l + (r - l) / 3, rmid = r - (r - l) / 3;
    double lck = calc(lmid), rck = calc(rmid);
    ans = min({ans, lck, rck});
    if (lck < rck) r = rmid - eps;
    else l = lmid + eps;
}
```

写法4:

```
for (int _ = 100; _; _--) {
    double lmid = l + (r - l) / 3, rmid = r - (r - l) / 3;
    double lck = calc(lmid), rck = calc(rmid);
    ans = min({ans, lck, rck});
    if (lck < rck) r = rmid;
    else l = lmid;
}
```

整数三分

```
for (int _ = 100; _; _--) {
    int lmid = (r + l) / 2, rmid = (r + lmid) / 2;
    double lck = calc(lmid), rck = calc(rmid);
    ans = min({ans, lck, rck});
    if (lck < rck) r = rmid;
    else l = lmid;
}
```