模拟退火

Simulate Annealing Arithmetic

（参考自 oi-wiki）

Idea：一句话概括：如果新状态的解更优，则更改当前答案；否则以一定概率更改答案。

定义当前温度为 T，新状态与当前状态之间的能量差为 $|\Delta E|$，则发生转移的概率为：

$$P(\Delta E)=\begin{cases}1 & \text{新状态更优}\\ e^{-\frac{\Delta E}{T}} & \text{新状态更劣}\end{cases}$$

设置初始温度 T_0，降温系数 d，终止温度 T_k，则退火过程如下：

1. 使温度 $T=T_0$；
2. 进行转移尝试，随后降温至 $T:=dT$；
3. 重复过程 2 直至 $T<T_k$.

注意，为了使解更精确，取整个退火过程中的最优解，而非最终解。

Trick：

- 重复退火直到快超时：`while((double)clock()/CLOCKS_PER_SEC < MAX_TIME) simulateAnneal();`
- 为了让解更好，可以在退火结束后以当前温度在当前最优解附近随机多次，尝试得到更优的解。

Code（以 [JSOI2004]平衡点为例）：

```cpp
#define T0 3000
#define Tk 1e-15
#define cool 0.996
#define MAX_TIME 0.75
inline double Rand(){ return (double)rand() / RAND_MAX; }
inline double calc(double xx, double yy){
    double res = 0;
    for(int i = 1; i <= n; i++)
        res += sqrt((xx-x[i])*(xx-x[i])+(yy-y[i])*(yy-y[i])) * w[i];
    if(res < ans)   ans = res, ansx = xx, ansy = yy;
    return res;
}
void simulateAnneal(){
    double T = T0;
    double nowx = ansx, nowy = ansy;
    while(T > Tk){
        double nxtx = nowx + T * (rand()*2-RAND_MAX);
        double nxty = nowy + T * (rand()*2-RAND_MAX);
        double deltaE = calc(nxtx, nxty) - calc(nowx, nowy);
        if(Rand() < exp(-deltaE / T))   nowx = nxtx, nowy = nxty;
        T *= cool;
    }
    for(int i = 1; i <= 1000; i++){
        double nxtx = ansx + T * (2*Rand()-1);
        double nxty = ansy + T * (2*Rand()-1);
        calc(nxtx, nxty);
    }
}

int main(){
    // ...
    while((double)clock()/CLOCKS_PER_SEC < MAX_TIME)
        simulateAnneal();
    // ...
}
```