一、简答

1. 平均值为区间和除以区间长度。单点更新、区间查询时，我们只需要在查询输出时除以区间长度即可。
2. 研究数对的偏序关系。

```
bool cmp(pair<int, int> a, pair<int, int> b) {
    if (a.first != b.first) return a.first < b.first;
    return a.second < b.second;
}
```

首先依据第一关键字优先，第二关键字次之排序。扫描数组，同时用树状数组维护已经访问过数对的第二关键字（桶），每个数对的得分更新前区间查询前缀和即可。时间复杂度为 $O(nlogn + n * log(1e6))$.

3.树状数组或线段树维护区间最值。注意查询区间由当前菜品数量和屏幕可显示菜品数量决定。时间复杂度为 $O(mlogm)$.

> 线段树维护区间最值:
> 将求和转换为求取最大值即可。

> 树状数组维护区间最值:
> 不同于线段树，树状数组不保存外部节点（原数组信息）。虽然在维护区间和时可以通过查询获得，但是在维护区间最值时就无法做到了。区间 A 的 max 为 a，将其分为A1，A2两个区间后无法求出各个区间的 max 。（不满足区间减法）因此树状数组在维护区间最值时要保存原始数组。

> 单点更新:
> 更新后，遍历直接相连的孩子节点，重新确定该节点的值。这里归纳一个性质：节点 x 的直接孩子为 $x - i$,其中 $i = 2^k < lowbit(x)$ 。

```
// A[]为原始数组
// 赋值操作:  A[x] = y
void update(int x, int y) {
    while (x <= N) {
        h[x] = y;
        // 重新求解区间(x - lowbit(x), x]最大值
        for (int i = 1; i < lowbit(x); i <<= 1) {
            h[x] = max(h[x], h[x - i]);
        }
        x += lowbit(x);
    }
```

> 区间查询:
> 维护最值时，区间减的性质无了，但是区间加仍满足！

```
int query(int x, int y)
{
        int ans = 0;
        while (y >= x)
        {
                ans = max(A[y], ans);
                y--;
                for (; y - lowbit(y) >= x; y -= lowbit(y))
                        ans = max(h[y], ans);
        }
        return ans;
}
```