E5I 小水獭和紧急演习
题目描述
加帕里幼儿园正在预防出现宇宙爆炸！
加帕里幼儿园可以被抽象为一个 m维欧氏空间，其中有 n 个实验室，第 i 个实验室的坐标为 $(xi,1,xi,2,\dots,xi,m)$。
加帕里幼儿园即将紧急演习，为计算财政支出，他们请小水獭帮忙计算一下加帕里幼儿园的直径，即任意两所实验室之间曼哈顿距离的最大值。
在 m维欧氏空间中，两点 $(x1,x2,\dots,xm)$ 和 $(y1,y2,\dots,ym)$ 的曼哈顿距离为：

$$\sum_{i=1}^{m}|xi-yi|$$

输入格式
第一行一个正整数 $t(1\le t\le 20)$，表示数据组数。
对于每组数据，第一行两个正整数 $n,m(2\le n\le 10^4$，$1\le m\le 5)$，含义同题目描述。
接下来 n行，第 $i(1\le i\le n)$行 m 个整数 $xi_1,xi_2,\dots,xi_m(-10^9\le xi,j\le 10^9)$，表示第 i个实验室的坐标。保证没有两个不同的实验室位于相同的位置。

输出格式
对于每组数据，输出一行一个正整数，表示任意两所实验室之间曼哈顿距离的最大值。

题解思路

首先我们来分析题目的要求：给定n个点，求这n个点的最大曼哈顿距离。
对于任意两个点，若其维度为m，则其曼哈顿距离可以直接用公式算出，时间复杂度为$O(m)$；
若采用朴素遍历的方式进行查找，则时间复杂度为$O(m*n^2)$, 有很大的可能超时，因此需要选择时间更快的方法。

考虑分析曼哈顿距离公式：

$$d=\sum_{k=1}^{m}|x_{i_k}-x_{j_k}|$$

对于其中的每一项，若去掉绝对值符号，则其只有两种可能：

$$1*(x_{i_k}-x_{j_k})\text{或}-1*(x_{i_k}-x_{j_k})\text{。}$$

设第i维的符号位为b_i, 则有

$$d=\sum_{k=1}^{m}b_i*(x_{i_k}-x_{j_k})$$

由于预先无法知道具体的符号位，因此将每一维度的**可能**的符号位（-1 或 1）写为长度为m的序列$\{b_{k_i}\}, i=1,2,\dots,m$.
易知这样的序列共有2^m种, 即$k=1,2,3,\dots,2^m$。

例如，当$m = 2$时，可能的序列有$\{-1,-1\},\{-1,1\},\{1,-1\},\{1,1\}$, 共 4 种。

定义$\{b_{\max(x,y)_m}\}$为点x, y的曼哈顿距离的符号序列, 其中$\max(x,y)$为点x, y对应符号序列的下标。则对于任意的$j = 1,2,\dots,m$，都满足

$$b_{\max(x,y)_j} * (x_j - y_j) = |x_j - y_j| = \max\{b_{k_j} * (x_j - y_j)\}, k = 1,2,3,\dots,2^m$$

因此对于每一个序列$\{b_{k_m}\}$, 对于任意两点x, y都有

$$\sum_{i=1}^{m} b_{k_i} * (x_i - y_i) \leq \sum_{i=1}^{m} b_{\max(x,y)_i} * (x_i - y_i) = d_{xy}$$

因此只需要将这$\boldsymbol{2^m}$**种可能**的序列带入两个点的坐标并**找出**$\sum_{i=1}^{m} b_{k_i} * (x_i - y_i)$的**最大值**，即为这两个点间的曼哈顿距离。

然而到这一步后仍然需要求$O(n^2)$次曼哈顿距离，没有真正的解决问题。

实际上，若将公式中的括号去掉，可以改写为

$$d_{xy} = \sum_{i=1}^{m} b_{\max(x,y)_i} * (x_i - y_i)$$

$$= \sum_{i=1}^{m} b_{\max(x,y)_i} * x_i - \sum_{i=1}^{m} b_{\max(x,y)_i} * y_i$$

可以看出，只要我们分别对于每一个点$P_j(x_{j_1}, x_{j_2}, \dots, x_{j_m})$求出对应所有序列$\{b_{k_i}\}$的值

$$p_{j_k} = \sum_{i=1}^{m} b_{k_i} * x_{j_i}$$

再将每一个序列$\{b_{k_i}\}$对应的所有点的p_{j_k}中的**最大值减去最小值**得到

$$q_k = p_{max_k} - p_{min_k}$$

然后再取所有序列中q的最大值

$$q_{max} = \max\{q_k\}, k = 1,2,3,\dots,2^m$$

就可以得出这n个点的最大曼哈顿距离$d_{max} = q_{max}$。

证明如下：

1. q_{max}是某两个点$P_i(x_{i_1}, x_{i_2}, \dots, x_{i_m}), P_j(x_{j_1}, x_{j_2}, \dots, x_{j_m})$的曼哈顿距离
   由等式

$$\sum_{i=1}^{m} b_{k_i} * (x_j - y_j) \leq \sum_{i=1}^{m} b_{\max(x,y)_i} * (x_j - y_j) = d$$

   可知，若q_{max}不为P_i, P_j的曼哈顿距离，则存在属于某一个序列$b_{\max(x,y)}$的值$p_{i_{\max(i,j)}} - p_{j_{\max(i,j)}} \geq q_{max}$为这两个点的曼哈顿距离。

i. 若$p_{i_{\max(i,j)}} - p_{j_{\max(i,j)}} = q_{max}$，则$q_{max}$即为这两个点的曼哈顿距离；

ii. 若 $p_{i_{\max(i,j)}} - p_{j_{\max(i,j)}} > q_{max}$ ，则有 $q_{\max(i,j)} \geq p_{i_{\max(i,j)}} - p_{j_{\max(i,j)}} > q_{max}$，即$q_{max}$不为所有$q$的最大值，与假设矛盾。

因此q_{max}是某两个点的曼哈顿距离；

2. q_{max}是最大曼哈顿距离

考虑所有的曼哈顿距离：

对于任意两点$P_i(x_{i_1}, x_{i_2}, \dots, x_{i_m}), P_j(x_{j_1}, x_{j_2}, \dots, x_{j_m})$，总存在一种序列$b_{\max(i,j)_m}$满足其曼哈顿距离

$$d_{ij} = \sum_{k=1}^{m} b_{\max(i,j)_k} * (x_{i_k} - x_{j_k}) = p_{i_{max_{ij}}} - p_{j_{max_{ij}}}$$

对于序列$\{b_{\max(i,j)_m}\}$，显然有$q_{\max(i,j)} \geq p_{i_{max_{ij}}} - p_{j_{max_{ij}}} = d_{ij}$.

因此对于任意两点间的曼哈顿距离d_{ij}，都有$q_{max} \geq q_{\max(i,j)} \geq d_{ij}$.

因此q_{max}是最大曼哈顿距离.

结合上述论述，我们需要做到：

1. 根据维数m生成对应的2^m个符号位序列$\{b_{k_m}\}$；
2. 对于所有的n个点P_j，生成所有2^m个符号位序列$\{b_{k_m}\}$对应的p_{j_k}，并分别放置于不同的2^m个数组中；
3. 对于每一个数组找到其最大值p_{max_k}和最小值p_{min_k}，取$q_k = p_{max_k} - p_{min_k}$；
4. 取$q_{max} = \max\{q_k\}$，则最大曼哈顿距离即为q_{max}。

因此，所需的时间复杂度应为$O(2^m * m) + O(2^m) * O(nm) + O(2^m) * O(n) + O(2^m) = O(2^m * mn)$。

对于本题的数据范围来说($2 \leq n \leq 10^4$，$1 \leq m \leq 5$),采用这种方法会更快。

代码

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<stdlib.h>
#include<algorithm>
#include<string.h>
long long p[100][50000] = { 0 };//低位存储，最多2^5个（先不同顺序，再不同点）
long long pmax[100];
long long pmin[100];
```

```cpp
int seq[100][10] = { 0 };//低位存储顺序(+或-),最多2^5个
using namespace std;
void initialize_seq(int p) {//2^p
    int temp = 0;
    for (int i = 0; i < (1 << p-1); i++) {
        for (int j = 0; j < p; j++) {
            seq[i][j] = (temp & (1 << j)) >> j;
            if (!seq[i][j])seq[i][j] = -1;
        }
        temp++;
    }//遍历组合
}
int main() {
    int t;
    scanf("%d", &t);
    initialize_seq(5);//初始化5维顺序,不同维度可以混用
    while (t--) {
        int n, m;
        long long v;
        long long M = 0;
        for (int i = 0; i < 100; i++) {
            pmax[i] = -1e20;
            pmin[i] = 1e20;
        }
        memset(p, 0, sizeof(p));
        //初始化
        scanf("%d%d", &n, &m);
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                scanf("%lld", &v);
                for (int k = 0; k < (1 << m - 1);k++) {
                    p[k][i] += seq[k][j] * v;
                }
            }
        }//输入点坐标

        for (int i = 0; i < (1 << m - 1); i++) {//不同顺序
            for (int j = 0; j < n; j++) {//不同点
                pmax[i] = max(pmax[i], p[i][j]);
                pmin[i] = min(pmin[i], p[i][j]);
            }
        }//对于每一个顶点，计算其所有顺序的p,并求出每个顺序的pmin和pmax
        for (int i = 0; i < (1 << m-1); i++) {
            M =max(M, pmax[i] - pmin[i]);
```

```
        }//求出q的最大值
        printf("%lld\n", M);
    }
}
```