B. LAYER CAKE

PROBLEM DESCRIPTION

給 n 個蛋糕，各有不同的長與寬，蛋糕可以沿著邊切割，題目問在所有堆疊蛋糕都一樣形狀的情況下，最多可以疊出多大體積的蛋糕(以及長寬)。

SOLUTION TECHINQUES

排序、枚舉

SOLUTION SKETCHES

枚舉蛋糕的長，再看每種蛋糕可以提供的寬，將寬蒐集起來做排序，然後枚舉蛋糕的寬：體積 = 長 * 寬[i] * (寬.size() – i)。

似乎有更快的作法。

TIME COMPLEXITY

$O(N^2 logN)$，N 為蛋糕的數量。

SOLUTION PROGRAM FOR REFERENCE

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <vector>

using namespace std;
typedef long long ll;

const int N = 4002;
const int M = 1000002;

int a[N][2];
char ex[M];
vector<int> can;

int main()
{
	int i, j, k, n;
	ll ans = 0; int ah, aw;
	scanf("%d", &n);
	can.reserve(n);
	for (i = 0; i < n; i++)
	{
		scanf("%d%d", &a[i][0], &a[i][1]);
		if (a[i][0] > a[i][1]) swap(a[i][0], a[i][1]);
		ex[a[i][0]] = ex[a[i][1]] = 1;
	}
	for (i = 0; i < M; i++)
		if (ex[i])
		{
			can.clear();
			for (j = 0; j < n; j++)
				if (a[j][0] >= i)
					can.push_back(a[j][1]);
				else if (a[j][1] >= i)
					can.push_back(a[j][0]);
			sort(can.begin(), can.end());
			for (j = 0; j < can.size(); j++)
			{
				ll tmp = (ll)i * can[j] * (can.size() - j);
				if (tmp > ans)
				{
					ans = tmp;
					ah = i;
					aw = can[j];
				}
			}
		}
	printf("%I64d\n", ans);
	printf("%d %d\n", ah, aw);
	return 0;
}
```