# LiveArchive 3506

*4 values whose sum is 0*

---

Prof. Edson Alves – UnB/FGA

# Problema

The SUM problem can be formulated as follows: given four lists $A, B, C, D$ of integer values, compute how many quadruplet $(a, b, c, d)$ belongs to $A \times B \times C \times D$ are such that $a + b + c + d = 0$. In the following, we assume that all lists have the same size $n$.

# Entrada e saída

**Input**

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line of the input file contains the size of the lists $n$ (this value can be as large as 4000). We then have $n$ lines containing four integer values (with absolute value as large as $2^{28}$) that belong respectively to $A, B, C$ and $D$.

**Output**

For each test case, your program has to write the number quadruplets whose sum is zero.

The outputs of two consecutive cases will be separated by a blank line.

# Exemplo de entradas e saídas

**Sample Input**

```
1

6
-45 22 42 -16
-41 -27 56 30
-36 53 -37 77
-36 30 -75 -46
26 -38 -10 62
-32 -54 -6 45
```

**Sample Output**

```
5
```

# Solução com complexidade $O(N^2 \log N)$

- O produto cartesiano $A \times B \times C \times D$ tem $4000^4 = 256 \times 10^{12}$, o que inviabiliza uma solução *naive* $O(N^4)$
- É possível utilizar a técnica *meet in the middle*, observando que $a + b = -(c + d)$
- Assim, é preciso computar as somas $xs$ e $ys$ dos pares $A \times B$ e $C \times D$, respectivamente
- As somas registradas em $ys$ devem ser ordenadas, de modo que seja possível utilizar a busca binária
- Para cada valor $x \in xs$, a resposta será incrementada em $L - R$, onde $[L, R)$ é o intervalo de índices de elementos $y_i$ em $ys$ tais que $y_i = -x$
- Este intervalo pode ser computado através da função `equal_range()` da STL da linguagem C++

# Solução com complexidade $O(N^2 \log N)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

const int MAX { 4010 };

ll as[MAX], bs[MAX], cs[MAX], ds[MAX], xs[MAX*MAX], ys[MAX*MAX];

ll solve(int N)
{
    ll ans = 0;

    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            xs[j + i*N] = as[i] + bs[j];

    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            ys[j + i*N] = cs[i] + ds[j];
```

# Solução com complexidade $O(N^2 \log N)$

```
22      sort(ys, ys + N*N);
23
24      for (int i = 0; i < N*N; i++)
25      {
26          auto p = equal_range(ys, ys + N*N, -xs[i]);
27          ans += (p.second - p.first);
28      }
29
30      return ans;
31  }
32
33  int main()
34  {
35      ios::sync_with_stdio(false);
36
37      int T;
38      cin >> T;
```

# Solução com complexidade $O(N^2 \log N)$

```
40      for (int test = 0; test < T; ++test)
41      {
42          int N;
43          cin >> N;
44
45          for (int i = 0; i < N; ++i)
46              cin >> as[i] >> bs[i] >> cs[i] >> ds[i];
47
48          if (test)
49              cout << endl;
50
51          cout << solve(N) << endl;
52      }
53
54      return 0;
55  }
```