Paradigmas de Resolução de Problemas

Busca Completa – *Meet in the Middle*: Exercícios Resolvidos

Prof. Edson Alves - UnB/FGA

2020

Sumário

LA 3506 – 4 values whose sum is 0

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 $b - a$, onde $[a, b)$ é o intervalo de índices de elementos y_i em ys tais que $ys_i = -x$
- 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 $O(N^2 \log N)$

```
21
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;
39
```

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  }
```

SPOJ SUBSUMS – Subset Sums

Problema

Given a sequence of N $(1 \leq N \leq 34)$ numbers $S_1, \ldots, S_N$ $(-20,000,000 \leq S_i \leq 20,000,000)$, determine how many subsets of S (including the empty one) have a sum between A and B $(-500,000,000 \leq A \leq B \leq 500,000,000)$, inclusive.

Entrada e saída

Input

The first line of standard input contains the three integers N, A, and B. The following N lines contain S_1 through S_N, in order.

Output

Print a single integer to standard output representing the number of subsets satisfying the above property. Note that the answer may overflow a 32-bit integer.

Exemplo de entradas e saídas

Sample Input

```
3 -1 2
1
-2
3
```

Sample Output

```
5
```

Solução $O(2^{N/2} \log N)$

- No pior caso, há 2^{34} subconjuntos a serem avaliados, de modo que uma solução que olhe todos eles individualmente resultará em um TLE
- A técnica do encontro no meio pode ser usada para dividir a entrada em dois grupos de aproximadamente $N/2$ elementos
- Para cada um destes grupos, é preciso computar as somas dos elementos de seus subconjuntos (listas S_1 e S_2, respectivamente)
- Para cada elemento $s \in S_1$, é preciso identificar todos os elementos $r \in S_2$ tais que $A \leq s + r \leq B$
- Se S_2 estiver ordenado, este intervalo de valores pode ser computado por meio de duas buscas binários, ou através das funções `lower_bound` e `upper_bound` da STL do C++

Solução $O(2^{N/2} \log N)$

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

using namespace std;
using ll = long long;

vector<ll> subset_sum(const vector<ll>& xs)
{
    vector<ll> s;

    for (size_t i = 0; i < (1ul << xs.size()); ++i) {
        ll sum = 0;

        for (size_t j = 0; j < xs.size(); ++j)
            if ((1 << j) & i)
                sum += xs[j];

       s.push_back(sum);
    }

    return s;
}
```

Solução $O(2^{N/2} \log N)$

```
22
23 ll solve(ll N, ll A, ll B, const vector<ll>& xs)
24 {
25     vector<ll> g1(xs.begin(), xs.begin() + N/2);
26     vector<ll> g2(xs.begin() + N/2, xs.end());
27
28     auto s1 = subset_sum(g1), s2 = subset_sum(g2);
29     sort(s2.begin(), s2.end());
30
31     ll ans = 0;
32
33     for (auto s : s1)
34     {
35         auto it = lower_bound(s2.begin(), s2.end(), A - s);
36         auto jt = upper_bound(s2.begin(), s2.end(), B - s);
37         ans += (jt - it);
38     }
39
40     return ans;
41 }
42
```

Solução $O(2^{N/2} \log N)$

```
43 int main()
44 {
45     ios::sync_with_stdio(false);
46
47     ll N, A, B;
48     cin >> N >> A >> B;
49
50     vector<ll> xs(N);
51
52     for (ll i = 0; i < N; ++i)
53         cin >> xs[i];
54
55     cout << solve(N, A, B, xs) << endl;
56
57     return 0;
58 }
```

Referências

1. Live Archive 3506 – 4 Values whose Sum if 0
2. SPOJ SUBSUMS – Subset Sums