Three Sum

Submissions Attempted by: 229 | Solved by: 56 | Partially Solved by: 126 |

No tags Add Tags

Given an array of **N** integers **A[1..N]**, you are asked to count the probability that a randomly picked ordered triplet of indexes **(i, j, k)**, where **i < j < k**, satisfying the condition that **A[i] + A[j] + A[k] = m^3** for some integer **m**. We will call any such triplet a **valid triplet**.

First, notice that in order to compute that probability, we can first compute the number of valid triples, and divide that number by the number of all possible triplets, which equals **binomial(n, 3)**.

How to find the number of valid triplets? First, we can notice, that **each element of the array is a positive integer not greater than 2000**, so **the sum of any 3 elements of the array is not greater than 6000**. How does it help? Well, you can notice that there are **only 18 cubes** of integer numbers which are not greater than 6000, specifically: 1, 8, 27, 64, 125, 216, 343, 512, 729, 1000, 1331, 1728, 2197, 2744, 3375, 4096, 4913, 5832.

The first solution which can come to you might be the following: let's first compute a boolean table **cube[X]**, where **cube[X] = 1** if and only if **X** is a cube of integer number, for **X <= 6000**. Next, let's iterate over all triplets **(i, j, k)** and check if **cube[A[i] + A[j] + A[k]]** is **true**. This approach is perfectly fine, but unfortunatelly it is too slow for the last few test files, because in one input file, we have to handle at most 40 test cases.

Let's try to speed it up. Do we need to iterate over all triples? Let's think for a second. What if we fixed the first two indexes, **i** and **j**. Then, instead of iterating over all **k**, such that **j < k <= n**, we can iterate over all 18 possible cubes, and for each one, let's say **C**, check how many occurrences of **C - (A[i] + A[j])** are in **A[j + 1, ..., N]**. This would be great, but how to compute the number of occurrences of a number **X** in **A[j + 1, ..., N]** for any valid **X** and **j**? Since all numbers in **A** are smaller than 2000 and **N** is at most 500, we can compute the array **occ[i][v] := number of occurrences of v in A[i, ..., N]**. We do it starting from **i = N to i = 1**. The pseudocode for computing occ table can look like this:

```
for i = n down to 1:
    for v = 1 to 2000:
        if i == n:
            occ[i][j] = 0
        else:
            occ[i][j] = occ[i + 1][j]
    occ[i][A[i]] += 1
```

That is a very nice speed up, because rather than **O(N^3)** we have a solution working in **O(N^2 * 18)** with **O(N * 2000)** precomputation time.

Edit Editorial

IS THIS EDITORIAL HELPFUL?

Yes, it's helpful

No, it's not helpful

16 developer(s) found this editorial helpful.

Author Solution by Vinay Kumar

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

using namespace std;

void fillCubes(double cubes[])
{
        for (int i = 0; i < 20; ++i)
        {
                cubes[i] = (double)i*(double)i*(double)i;
        }
        return;
}

int main()
{
        int t;
        scanf("%d",&t);
        assert(t >= 1 && t <= 40);
        double cubes[20];
        fillCubes(cubes);
        while(t--)
        {
                int n;
                scanf("%d",&n);
                assert(n >= 3 && n <= 1000);
                double a[510]={0};
                map<double, int> mp;
                for (int i = 1; i <= n; ++i)
                {
                        scanf("%lf",&a[i]);
                        assert(a[i] >= 1 && a[i] <= 2000);
                        mp[a[i]]++;
                }
                sort(a,a+n+1);
                double sum,tmp;
                int t1 = 0,t2 = 0;
```

```
                int count = 0;
                for (int i = 1; i <= n-2;)
                {
                        t2 = 0;
                        for (int j = i+1; j <= n-1;)
                        {
                                sum = a[i]+a[j];
                                for (int k = 0; k < 20; ++k)
                                {
                                        tmp = cubes[k]-sum;
                                        if(tmp>=a[j] && mp.find(tmp
                                        {
                                                if(a[j] != tmp)
                                                        count+=mp[t
                                                else if(a[i] != a[j
                                                        count+=(mp[
                                                else if(a[i] == a[j
                                                        count+=(mp[
                                        }
                                }
                                ++j;
                                if(a[j] == a[j-1])
                                        t2++;
                                else
                                        t2=0;
                        }
                        ++i;
                        if(a[i] == a[i-1])
                                t1++;
                        else
                                t1=0;
                }
                double den = n*(n-1)*(n-2);
                den /= 6.0;
                printf("%.9lf\n", (double)count/den);
        }
        return 0;
}
```

Tester Solution by Deepankar Anil Kumar

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

using namespace std;

#define vi vector < int >
#define pii pair < int , int >
#define pb push_back
#define mp make_pair
#define ff first
#define ss second
#define foreach(it,v) for( __typeof((v).begin())it = (v).begin() ;
```

```
#define ll long long
#define llu unsigned long long
#define MOD 1000000007
#define INF 0x3f3f3f3f
#define dbg(x) { cout<< #x << ": " << (x) << endl; }
#define dbg2(x,y) { cout<< #x << ": " << (x) << " , " << #y << ":
#define all(x) x.begin(),x.end()
#define mset(x,v) memset(x, v, sizeof(x))
#define sz(x) (int)x.size()

int a[555];
int cnt[555][2001];
vi cubes;

void prec()
{
    int i = 1;
    while(i*i*i <= 6000)
    {
        cubes.pb(i*i*i);
        i++;
    }
}

int main()
{
    prec();
    int t;
    cin >> t;
    assert(1 <= t && t <= 40);
    while(t--)
    {
        int i,j,k;
        int n;
        cin >> n;
        assert(3 <= n && n <= 500);
        for(i=1;i<=n;i++)
        {
            cin >> a[i];
            assert(1 <= a[i] && a[i] <= 2000);
        }

        for(i=1;i<=n;i++)
        {
            for(j=1;j<=2000;j++)
            {
                cnt[i][j] = cnt[i-1][j] + (a[i] == j);
            }
        }

        int num = 0 , den = (n*(n-1)*(n-2))/6;


        for(i=1;i<=n;i++)
```

```
66.          {
67.               for(j=i+1;j<=n;j++)
68.               {
69.                    for(k=0;k<sz(cubes);k++)
70.                    {
71.                         int cube = cubes[k];
72.                         if(cube < a[i] + a[j])
73.                              continue;
74.                         int rem = cube - a[i] - a[j];
75.                         if(rem > 2000)
76.                           continue;
77.                         num += (cnt[n][rem] - cnt[j][rem]);
78.                    }
79.               }
80.          }
81.
82.          double ans = (double)(num)/(double)(den);
83.
84.          printf("%.9lf\n",ans);
85.      }
86.      return 0;
87. }
```

COMMENTS (0)

Start Discussion...

Cancel Post

PROFILE IMPACT

Complete Profile

*Excellent profile will increase your profile discoverability and keep you on top among others.

PROBLEMS SUGGESTED FOR YOU

Flip the words
Solved by 14

Hack Sequence
Solved by 279

Friends Everywhere
Solved by 147

more...

RECENT SUBMISSIONS

User	Result	Time	Lang
Bhawnesh...		5.639	C++
Bhawnesh...		2.0665	C++
Suparno ...		10.0155	C
Suparno ...		10.0331	C
Shantam ...		1.2175	C++
Nishank ...		1.379	C++
Nishank ...		4.2691	C++

View All

TRENDING NOTES

Number Theory - III
written by Boris Sokolov

Exact String Matching Algorithms
written by Alei Reyes

Binary Indexed Tree or Fenwick Tree
written by Chandan Mittal

Small tricks in for loop
written by Rangeesh

Strings And String Functions
written by Vinay Singh

more ...

DEVELOPERS TO FOLLOW

Deepa Panwar
14 followers

Sachin Gupta
6361 followers

Amit Mittal
2 followers

COMPANIES TO FOLLOW

Medlife International
2058 followers

Akamai Technologies
1927 followers

Intuit
2001 followers

RECOMMENDED CHALLENGES

Horlicks Hack 4 Fun

03 Sep 2015, 09:00 PM IST

Register

CODE-HUNT-2F

21 Oct 2015, 05:00 PM IST

Register

Zoomcar Ruby Challenge

23 Oct 2015, 06:00 PM IST

Register

Zomato Hiring Challenge

23 Oct 2015, 06:00 PM IST

Register

Diona iOS Developer Hiring Challenge

24 Oct 2015, 12:00 PM IST

Register

Tipstat Android Developer Hiring Challenge

24 Oct 2015, 12:00 PM IST

Register

D'code

SUBSCRIBE TO HACKEREARTH NEWS

bhawnesh.dipu@gmail.com

Subscribe

JOIN PROGRAMMING CLUB ON FACEBOOK

Join now