



[خانه](#) > [توسعه‌دهنده](#) > [با کوثرابلاگ](#) > [مسابقات و رویدادها](#) > [راه حل‌های مسابقه الگوریتمی ۲۰۱۸](#)

## Newbies ۲۰۱۸

۱۱۱ ثانیه | ۱۴ دقیقه زمان مطالعه:

۱۴۰۱ بهمن ۱۴۰۲



سلام!

مسابقات الگوریتمی Newbies هر سال در دانشگاه شهید بهشتی برگزار می‌شود. سطح سوالات این مسابقه به نسبت مسابقات رسمی ACM آسون تن است و لی سبک سوالات حفظ شده و شرکت توش برای علاقه‌مندان الگوریتم، بجهه‌های المپیادی، دانشجوهای ترم اول و دوم و همه کسانی که می‌خواهند در مسابقات ACM-ICPC شرکت کنند، می‌توانند جذاب و چالشی باشند.

همون طور که می‌دونیم گرفتیم مسابقات سال‌های گذشته Newbies را به صورت آنلاین در کوثرابلاگ کنیم تا فرصتی برای تمرین و رقابت باشند. این هفته مسابقه‌ای الگوریتمی Newbies ۲۰۱۸ در کوثرابلاگ برگزار شد. این مسابقه در سال ۲۰۱۸ در دانشگاه شهید بهشتی با تلاش «محمد رضا محسنی»، «محمد نصیری‌فر»، «محمد حسن‌پور»، «عرفان علی‌محمدی»، «سامان خامسیان» و «رضا شیری» طراحی و برگزار شده بود.

راه حل‌های سوالات این مسابقه در ادامه توضیح داده شدند. در صورتی که متوجه راه حلی نشدید، می‌توانید در بخش نظرات، سوالات و ابهام‌های خودتون رو

طرح کنید. اگه راحل دیگهای برای سؤالات دارید، خوشحال می‌شیم که راه حلتون رو در بخش نظرات با ما و دوستانتون به اشتراک بذارید.

## A – AbulHassan’s Quest

اگر  $b \neq 0$  آنگاه:

$$q = \frac{a - r}{b}$$

سپس کافی است چک کنیم که آیا  $a - r$  مضرب  $b$  هست یا نه و اگر  $a - r = b$  آنگاه کافی است چک کنیم  $a$  برابر  $r$  است یا خیر.

پیچیدگی زمانی:

$$\mathcal{O}(T)$$

```

1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 int main()
7 {
8     int T;
9     cin >> T;
10    for(int test = 0; test < T; test++)
11    {
12        int a, b, r;
13        cin >> a >> b >> r;
14        if(b == 0)
15            cout << (a == r?"Yes\n":"No\n");
16        else
17            cout << ((a-r)%b == 0?"Yes\n":"No\n");
18    }
19 }
20
21 //Thank God

```

## B – Farshad’s Research

در ابتدا اندازهی هر مؤلفه همبندی را بدست می‌آوریم (این کار با `dfs` و `dsu` قابل انجام است). و سپس آنها را مرتب شده در آرایه‌ای مثل  $V$  ذخیره می‌کنیم. اگر  $f(x)$  را تعداد مؤلفه‌ها با سایز کمتر از  $x$  بگیریم آنگاه پاسخ برابر  $(l) - f(r + 1) - f(l)$  خواهد بود. همچنین برای محاسبه  $f(x)$  از روش جستجوی دودویی کمک می‌گیریم.

پیچیدگی زمانی:

$$\mathcal{O}(T \times (N + M \log N + Q \log N))$$

ما برای سادگی پیاده‌سازی از `dsu` با پیچیدگی زمانی  $M \log N$  استفاده می‌کنیم.

```

1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 int main()
7 {
8     ios_base::sync_with_stdio(false);
9     int T;
10    cin >> T;
11    for(int test = 0; test < T; test++)
12    {
13        int n, m, q;
14        cin >> n >> m >> q;
15        int head[n], sz[n];
16        for(int i = 0; i < n; i++)
17            head[i] = i, sz[i] = 1;
18        for(int u, v, i = 0; i < m; i++)
19        {
20            cin >> u >> v;
21            u--, v--;
22            while(u != head[u]) u = head[u];
23            while(v != head[v]) v = head[v];
24            if(u == v) continue;
25            if(sz[v] < sz[u]) swap(u, v);
26            sz[v] += sz[u];
27            head[u] = v;
28        }
29        vector<int> sizes;
30        for(int i = 0; i < n; i++)
31            if(i == head[i])
32                sizes.push_back(sz[i]);
33        sort(sizes.begin(), sizes.end());
34        for(int l, r, i = 0; i < q; i++)
35        {
36            cin >> l >> r;
37            l = lower_bound(sizes.begin(), sizes.end(), l)-sizes.begin();
38            r = upper_bound(sizes.begin(), sizes.end(), r)-sizes.begin();
39            cout << r-l << '\n';
40        }
41    }
42 }
43
44 //Thank God

```

## C – Shortest but Hardest

ادعا می‌کنیم پاسخ برابر کوچکترین عضو آرایه است (آن را از این به بعد  $m$  در نظر بگیرید)! مشخصاً زیردنباله‌ای که تنها از  $m$  ساخته شده این خاصیت را دارد. حال ثابت می‌کنیم هر زیردنباله‌ای مانند:

$$A_l, A_{l+1}, \dots, A_r$$

میانگین هندسی بیشتر مساوی  $m$  دارد.

$$\sqrt[r-l+1]{A_l \cdot A_{l+1} \cdots \cdot A_r} \geq \sqrt[r-l+1]{m \cdot m \cdots \cdot m} = m$$

پیچیدگی زمانی:

$$\mathcal{O}(T \times n)$$

```

1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 int main()
7 {
8     int T;
9     cin >> T;
10    for(int test = 0; test < T; test++)
11    {
12        int n;
13        cin >> n;
14        int A[n];
15        for(int i = 0; i < n; i++)
16            cin >> A[i];
17        cout << *min_element(A, A+n) << ".00" << '\n';
18    }
19 }
20
21 //Thank God

```

## D – Divisible Strings

برای حل این سؤال حریصانه عمل می‌کنیم، به این شکل که سعی می‌کنیم بزرگترین  $k$  را پیدا کنیم به طوری که با حذف برخی از حروف  $S$  حاصل  $T \times k$  شود و بدین صورت پاسخ برابر  $|S| - k \times |T|$  خواهد بود.

پس باید رشته  $S = s_1, s_2, \dots, s_{k*|T|}$  به  $S$  تبدیل شود به طوری که:

$$s_1 = T_1, s_2 = T_2, \dots, s_{|T|+1} = T_1, \dots, s_{|T|\times k} = T_{|T|}$$

حال از ابتدای  $S$  شروع کرده و آنقدر حرف از ابتدا حذف می‌کنیم تا به  $s_1$  برسیم و بعد از آن دوباره آنقدر حرف حذف می‌کنیم تا به  $s_2$  برسیم و ...

پیچیدگی زمانی:

$$\mathcal{O}(T \times (|S| + |T|))$$

```
1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 int main()
7 {
8     int T;
9     cin >> T;
10    getchar();
11    for(int test = 0; test < T; test++)
12    {
13        string s, t;
14        getline(cin, s);
15        getline(cin, t);
16        int ps = 0, pt = 0, k = 0;
17        while(ps < s.size())
18        {
19            if(s[ps] == t[pt])pt++;
20            ps++;
21            if(pt == t.size())k++, pt = 0;
22        }
23        cout << s.size()-k*t.size() << '\n';
24    }
25 }
26
27 //Thank God
```

## E – Tennis

برای این سؤال کافی است همه حالت‌های جایگذاری؟ را امتحان کنید و در هر حالت معتبر برنده را شناسایی کنید.

پیچیدگی زمانی: منظور از  $MAX_{length}$  در اینجا همان 11 است.

$$\mathcal{O}(T \times MAX_{length} \times 3^{MAX_{length}})$$

```
1 #include <iostream>
2 #include <cmath>
3 #include <algorithm>
4 #include <vector>
5 #include <cassert>
6
7 using namespace std;
8
9 const int NASIRIFAR = 1;
10 const int MORTEZA = 2;
11 const int INVALID = 0;
12
13 int pw(int a, int b) {
14     int ret = 1;
15     for(int i = 0 ;i < b; i++) {
16         ret *= a;
17     }
18     return ret;
19 }
20
21 string renderSequence(string sequence, const vector<char>& assignments) {
22     int assignmentPtr = 0;
23     for(int i = 0;i < sequence.size(); i++) {
24         if(sequence[i] == '?') {
25             sequence[i] = assignments[assignmentPtr];
26             assignmentPtr++;
27         }
28     }
29     return sequence;
30 }
31
32 enum State {
33     toServe,
34     ballInAirServe,
35     ballInAir,
36     groundOnce,
37 };
38
39 const char RACKET = 'R';
40 const char GROUND = 'G';
41 const char NET = 'N';
42
43 int otherPlayer(int player) {
44     if(player == NASIRIFAR) {
45         return MORTEZA;
46     }
47     return NASIRIFAR;
48 }
49
50 int getResult(const string& sequence) {
51     int scores[3] = {0};
52
53     int state = toServe;
54     int lastHit = 0;
55
56     for(auto event: sequence) {
57         if(state == toServe) {
58             if(event != RACKET) {
59                 return INVALID;
60             }
61             lastHit = NASIRIFAR;
62             state = ballInAirServe;
63         } else if(state == ballInAirServe) {
64             if(event == RACKET) {
65                 return INVALID;
66             } else if(event == NET) {
67                 scores[otherPlayer(lastHit)]++;
68                 state = toServe;
69             } else if(event == GROUND){
70                 state = groundOnce;
71             }
72         } else if(state == ballInAir) {
73             if(event == RACKET) {
74                 lastHit = otherPlayer(lastHit);
75             }
76         }
77     }
78 }
```

```
75         } else if(event == NET) {
76             scores[otherPlayer(lastHit)]++;
77             state = toServe;
78         } else if(event == GROUND){
79             state = groundOnce;
80         }
81     } else if (state == groundOnce) {
82         if(event == RACKET) {
83             lastHit = otherPlayer(lastHit);
84             state = ballInAir;
85         } else if(event == NET) {
86             return INVALID;
87         } else if(event == GROUND){
88             state = toServe;
89             scores[lastHit]++;
90         }
91     }
92 }
93
94 if(state != toServe) {
95     return INVALID;
96 }
97
98 if(scores[MORTEZA] >= scores[NASIRIFAR]) {
99     return MORTEZA;
100 }
101 return NASIRIFAR;
102 }
103
104 int main() {
105     int t;
106     cin >> t;
107     assert(t <= 50);
108     while(t --) {
109         string soundsSequence;
110         cin >> soundsSequence;
111
112         assert(soundsSequence.size() <= 11);
113
114         vector<char> choices = {RACKET, GROUND, NET};
115
116         int numQuestionMarks = count(soundsSequence.begin(), soundsSequence.end(), '?');
117         int numAllSequences = pw(3, numQuestionMarks);
118
119         int numMorteza = 0, numNasirifar = 0;
120
121         for(int coding = 0; coding < numAllSequences; coding++) {
122             vector<char> assignments;
123             int origCoding = coding;
124             for(int i = 0 ;i < numQuestionMarks; i++) {
125                 assignments.push_back(choices[coding%3]);
126                 coding /= 3;
127             }
128             coding = origCoding;
129             string renderedSequence = renderSequence(soundsSequence, assignments);
130             int result = getResult(renderedSequence);
131             if(result == NASIRIFAR) {
132                 numNasirifar++;
133             } else if (result == MORTEZA) {
134                 numMorteza++;
135             }else {
136                 // invalid sequence
137             }
138         }
139         assert(numMorteza + numNasirifar >= 0);
140
141         if(numMorteza >= numNasirifar) {
142             cout << "Morteza to doost dashtani hasti" << endl;
143         } else {
144             cout << "Nasirifar to doost dashtani hasti" << endl;
145         }
146     }
147
148     return 0;
149 }
```

## F – Profiling

این سؤال به چند بخش تقسیم می‌شود (اگر موردی در چند بخش بود بخش با اولویت بالاتر مد نظر است) :

۱. اگر  $k < n$  آنگاه جواب ۰ است.

۲. اگر  $n = k$  آنگاه جواب ۱ است.

۳. اگر  $k = 0$  آنگاه  $fib(0)$  به اندازه  $fib(2)$  صدای زده می‌شود پس می‌توانیم  $k$  را ۲ در نظر بگیریم.

۴. به استقرا می‌توان نشان داد جواب  $fib(n - k)$  است. پایه برای  $n = k + 1$  و  $n = k$  مشخص است (در این دو حالت جواب یک است یعنی همان  $fib(0) = fib(1) = 1$ ) و گام هم به آسانی اثبات می‌شود.

پیچیدگی زمانی:

$$\mathcal{O}(MAX_N + T)$$

```

1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 const int mod = 1'000'000'007;
7 const int N = 100'000 + 5;
8 int fib[N];
9
10 int main()
11 {
12     fib[0] = fib[1] = 1;
13     for(int i = 2; i < N; i++)
14         fib[i] = (fib[i-1]+fib[i-2])%mod;
15     int T;
16     cin >> T;
17     for(int test = 0; test < T; test++)
18     {
19         int n, k;
20         cin >> n >> k;
21         if(k == n)
22             cout << 1 << '\n';
23         else if(k > n or n < 2)
24             cout << 0 << '\n';
25         else if(k == 0)
26             cout << fib[n-2] << '\n';
27         else
28             cout << fib[n-k] << '\n';
29     }
30 }
31
32 //Thank God

```

## G – IMEI

برای بخش اول سؤال کافی است ۸ رقم اول را چاپ کنید و برای بخش دوم ابتدا `sumdigits` را طبق دستورالعمل حساب می‌کنیم و حال باقی مانده آن بر  $s$  در نظر بگیرید. اگر  $s = 0$  آنگاه  $checksum = 10 - s$  و در غیر اینصورت  $checksum = 0$  خواهد بود.

پیچیدگی زمانی:

$$\mathcal{O}(T)$$

```

1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 int main()
7 {
8     int T;
9     cin >> T;
10    for(int test = 0; test < T; test++)
11    {
12        string n;
13        cin >> n;
14        for(int i = 0; i < 8; i++)cout << n[i];
15        int sum = 0;
16        for(int i = 0; i < n.size(); i++)
17        {
18            if(i%2 == 0)sum += n[i]-'0';
19            else sum += (n[i]-'0')*2/10+(n[i]-'0')*2%10;
20        }
21        sum %= 10;
22        cout << ' ' << (sum == 0?sum:10-sum) << '\n';
23    }
24 }
25
26 //Thank God

```

## H – Fall in Love

اگر  $m$  عدد اول باشد که تنها کافی است بررسی کنیم  $m$  بزرگتر از  $n$  هست یا نه.  
حال فرض می‌کنیم  $m$  اول نباشد. پس:

$$m = \prod p_i^{q_i} (1 \leq p_i \leq \sqrt{m}, 0 \leq q_i)$$

حال اگر  $n!$  بر یک از  $p_i^{q_i}$  بخش‌پذیر باشد آنگاه  $n!$  بر  $m$  بخش‌پذیر خواهد بود. تعداد تکرارهای عامل اول  $p_i$  در  $n!$  به طریق زیر به دست می‌آید (با استقرار روی  $n$  می‌توانید ثابت کنید):

$$count_{p_i} = \sum_{k=1}^{\infty} \frac{n}{p_i^k}$$

پس کافی است چک کنیم آیا تعداد تکرارهای عوامل اول  $m$  همیشه در  $n$  بزرگتر مساوی هست یا نه.

پیچیدگی زمانی:  $N$  را حداقل  $m$  و  $n$  در نظر بگیرید که در سؤال <sup>31</sup> است.

$$\mathcal{O}(\sqrt{N} \log \log \sqrt{N} + T \times \frac{\sqrt{N}}{\log \sqrt{N}} \times \log \log N)$$

که  $\sqrt{N} \log \log \sqrt{N}$  پیچیدگی زمانی [غربال اعداد برای یافتن اعداد اول](#) است و نیز  $\frac{\sqrt{N}}{\log \sqrt{N}}$  تعداد اعداد اولی هست که می‌توانند عامل اول  $m$  در صورتی که اول نباشد، باشند.

```

1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 constexpr int N = 2147483647; // 2^31-1
7 constexpr int SQN = 31622; // sqrt(N)
8 bitset<SQN> is_prime;
9 vector<int> primes;
10
11 int main()
12 {
13     is_prime.set(), is_prime.reset(0), is_prime.reset(1);
14     for(int i = 0; i < SQN; i++)
15         if(is_prime[i])
16         {
17             primes.push_back(i);
18             for(int j = i+i; j < SQN; j += i)
19                 is_prime.reset(j);
20         }
21
22     int T;
23     cin >> T;
24     for(int test = 0; test < T; test++)
25     {
26         int n, m, m_copy;
27         cin >> n >> m, m_copy = m;
28         bool divides = true;
29         for(auto p: primes)
30         {
31             if(p*p > m or !divides)break;
32             int cnt_p = 0;
33             while (m%p == 0) cnt_p++, m /= p;
34             int cnt_n_factorial = 0, n_copy = n;
35             while(cnt_n_factorial < cnt_p and n_copy > 0)
36                 n_copy /= p, cnt_n_factorial += n_copy;
37             divides &= (cnt_n_factorial >= cnt_p);
38         }
39         if(m > 1 and n < m)divides = false;
40         cout << m_copy << (divides?" divides ":" does not divide ") << n << "!\n";
41     }
42 }
43
44 //Thank God

```

## I – Lightning Strike

برای حل سوال آن را به ۴ بخش تقسیم می‌کنیم. (با فرض داشتن برخورد) برخورد دو توپ پس از برخورد هر کدام به دیوار باشد یا برخورد هر دو قبل از برخورد به دیوار یا یکی قبل از برخورد به دیوار و دیگری پس از برخورد به دیوار (۲ حالت).

حال عملاً مساله به این تبدیل می‌شود که دو توپ دو بعدی با شعاع‌ها، مکان‌ها و سرعت‌های مختلف داریم آیا اینها به هم برخورد می‌کنند یا نه (توجه کنید باید معتبر بودن جواب هم چک کنیم که مثلاً از دیوار توپ رد نشود).؟!

برای اینکار از قضیه [سرعت نسبی](#) در فیزیک استفاده می‌کنیم. یعنی می‌توانیم فرض کنیم یک توپ ثابت است و فقط یک توپ با سرعتی جدید حرکت می‌کند. حال می‌خواهیم چک کنیم که آیا با هم برخورد دارند یا خیر.

می‌دانیم مسیر حرکت مرکز توپ تشکیل یک خط (نیمخط یا پاره خط در واقع) و مرکز توپ دیگر ساکن است. پس می‌توانیم با استفاده از [فرمول فاصله نقطه از خط](#) کمینه فاصله مرکز آن دو را بدست آوریم (از این پس آن را  $d$  در نظر می‌گیریم). حال اگر  $d \leq R_1 + R_2$  پس آنگاه آن ها با هم تماس دارند. همچنانی بجای فرمول فاصله نقطه از خط می‌توانید از الگوریتم‌هایی مثل [search ternary](#) هم استفاده کنید.

پیچیدگی زمانی:

$$\mathcal{O}(T)$$

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 const double eps = 1e-6;
6
7 class Vector;
8
9 class Point {
10 public:
11     double x, y;
12
13     Point(double x, double y): x(x), y(y) {
14     }
15
16     Point() {
17     }
18
19     Point translate(const Vector& v) const ;
20 };
21
22 class Vector {
23 public:
24     double x, y;
25     Vector(double x, double y): x(x), y(y) {
26     }
27
28     Vector() {
29     }
30
31     Vector scale(double factor) const;
32     Vector normalize() const;
33     double cross(const Vector& o) const;
34     Vector opposite() const;
35     double length() const;
36 };
37
38 ostream& operator<<(ostream& out, const Point& p) {
39     out << "(" << p.x << ", " << p.y << ")";
40     return out;
41 }
42
43 ostream& operator<<(ostream& out, const Vector& v) {
44     out << "(" << v.x << ", " << v.y << ")";
45     return out;
46 }
47
48 class Ball {
49 public:
50     Ball(const Point& pos, const Vector& displacement, double r, double velocity):
51         pos(pos), displacement(displacement.normalize()), r(r), velocity(velocity) {
52     }
53     Point pos;
54     Vector displacement;
55     double r;
56     double velocity;
57     Point posAfter(double t) const;
58 };
59
60 class LinearTrajectory {
61 public:
62     double minTime, maxTime;
63
64     Point start;
65     Vector displacement;
66     double v;
67
68     Point get(double) const;
69
70     LinearTrajectory(double minTime, double maxTime, const Point& start, const Vector& displacement, double v):
71         minTime(minTime), maxTime(maxTime), start(start), displacement(displacement.normalize()), v(v) {
72     }
73
74     LinearTrajectory() {
```

```
75     }
76
77 };
78
79 vector<LinearTrajectory> trajectories(const Ball& b, const vector<Point>& wall);
80 double findMin(const LinearTrajectory& l1, const LinearTrajectory& l2);
81 bool segmentInterLine(const Point& s1, const Point& s2, const Point& l1, const Point& l2);
82 int sgn(double a);
83 double dist(const Point& p1, const Point& p2);
84 double eq(double a, double b);
85 Vector vectorFromTo(const Point& from, const Point& to);
86
87 int main() {
88     int n;
89     cin >> n;
90
91     while(n--) {
92         vector<Ball> balls;
93         vector<Point> wall;
94
95         for(int i = 0 ;i < 2; i++) {
96             double x, y, dx, dy;
97             cin >> x >> y >> dx >> dy;
98             double velocity, r;
99             cin >> velocity >> r;
100            balls.push_back(Ball(Point(x, y), Vector(dx, dy).normalize(), r, velocity));
101        }
102
103        for(int i = 0 ;i < 2 ; i++) {
104            double x, y;
105            cin >> x >> y;
106            wall.push_back(Point(x, y));
107        }
108    }
109
110    auto traj0 = trajectories(balls[0], wall);
111    auto traj1 = trajectories(balls[1], wall);
112    bool hit = false;
113
114    for(auto t0: traj0) {
115        for(auto t1: traj1) {
116            auto minDist = findMin(t0, t1);
117            hit |= minDist < balls[0].r + balls[1].r;
118        }
119    }
120
121    if(hit) {
122        cout << "Lightning strike" << endl;
123    } else {
124        cout << "Not even a spark" << endl;
125    }
126 }
127
128 return 0;
129 }
130
131 vector<LinearTrajectory> trajectories(const Ball& b, const vector<Point>& wall) {
132     double st = 0;
133     double en = 1e12;
134
135     double lo = st, hi = en;
136     for(int i = 0 ;i < 1000; i++) {
137         double md = (hi + lo)/2;
138         if(segmentInterLine(b.pos, b.posAfter(md), wall[0], wall[1])) {
139             hi = md;
140         } else {
141             lo = md;
142         }
143     }
144
145     double timeOfImpact = (hi + lo)/2;
146     Point centerOfBallOnImpact = b.posAfter(timeOfImpact);
147     Point pointFromWall = wall[0];
148     if(eq(dist(pointFromWall, centerOfBallOnImpact), 0)) {
```

```
150     pointFromWall = wall[1];
151 }
152
153
154 Vector vWall = vectorFromTo(centerOfBallOnImpact, pointFromWall);
155 Vector vTraj = vectorFromTo(centerOfBallOnImpact, b.pos);
156
157 double sineOfDirections = fabs(vWall.cross(vTraj))/(vWall.length() * vTraj.length());
158 double displacementNeededInDirOfTraj = sineOfDirections * b.r;
159 Point realCenterOfBall = centerOfBallOnImpact.translate(
160     vectorFromTo(centerOfBallOnImpact, b.pos).normalize().scale(displacementNeededInDirOfTraj));
161 double cosineOfDirections = sqrt(1 - pow(sineOfDirections, 2));
162 double mirrorPointDissplacementFactor = dist(realCenterOfBall, centerOfBallOnImpact) * cosineOfDirections * 2;
163
164 Vector maybeDirectionToFindMirrorPoint = vectorFromTo(wall[0], wall[1]).normalize();
165 Point maybeMirrorPoint = realCenterOfBall.translate(maybeDirectionToFindMirrorPoint.scale(mirrorPointDissplacementFactor));
166 if(!eq(dist(maybeMirrorPoint, centerOfBallOnImpact), dist(realCenterOfBall, centerOfBallOnImpact))) {
167     maybeDirectionToFindMirrorPoint = maybeDirectionToFindMirrorPoint.opposite();
168     maybeMirrorPoint = realCenterOfBall.translate(maybeDirectionToFindMirrorPoint.scale(mirrorPointDissplacementFactor));
169 }
170
171 Vector dMirror = vectorFromTo(centerOfBallOnImpact, maybeMirrorPoint).normalize();
172
173 vector<LinearTrajectory> ret;
174
175 double realTimeOfImpact = dist(realCenterOfBall, b.pos)/b.velocity;
176
177 ret.push_back(LinearTrajectory(0, realTimeOfImpact, b.pos, b.displacement, b.velocity));
178 if(timeOfImpact < 1e12 - 1) {
179     ret.push_back(LinearTrajectory(realTimeOfImpact, en, realCenterOfBall, dMirror, b.velocity));
180 }
181
182 return ret;
183 }
184
185 double findMin(const LinearTrajectory& l1, const LinearTrajectory& l2) {
186     double lo = min(l1.minTime, l2.minTime);
187     double hi = max(l1.maxTime, l2.maxTime);
188
189     for(int i = 0 ;i< 1000; i++) {
190         if(lo > hi) {
191             return 1231231231.0;
192         }
193         double oneThird = lo + (hi - lo)/3;
194         double twoThirds = lo + 2*(hi - lo)/3;
195         if(dist(l1.get(oneThird), l2.get(oneThird)) < dist(l1.get(twoThirds), l2.get(twoThirds))) {
196             hi = twoThirds;
197         } else {
198             lo = oneThird;
199         }
200     }
201     return dist(l1.get((hi + lo)/2), l2.get((hi + lo)/2));
202 }
203
204 bool segmentInterLine(const Point& s1, const Point& s2, const Point& l1, const Point& l2) {
205     return sgn(vectorFromTo(s1, l1).cross(vectorFromTo(s1, l2))) != sgn(vectorFromTo(s2, l1).cross(vectorFromTo(s2, l2)));
206 }
207
208 double dist(const Point& p1, const Point& p2) {
209     return sqrt(pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2));
210 }
211
212 double eq(double a, double b) {
213     return fabs(a - b) < eps;
214 }
215
216 int sgn(double a) {
217     if(a < 0) {
218         return -1;
219     } else if(a > 0) {
220         return 1;
221     }
222     return 0;
223 }
```

```

225 Vector vectorFromTo(const Point& from, const Point& to) {
226     return Vector(to.x - from.x, to.y - from.y);
227 }
228
229 Point Ball::posAfter(double t) const {
230     return pos.translate(displacement.scale(t * velocity));
231 }
232
233 Point Point::translate(const Vector& v) const {
234     return Point(x + v.x, y + v.y);
235 }
236
237 Vector Vector::scale(double factor) const {
238     return Vector(factor * x, factor * y);
239 }
240
241 Point LinearTrajectory::get(double t) const {
242     double dt = t - minTime;
243     return start.translate(displacement.scale(dt * v));
244 }
245
246 Vector Vector::normalize() const {
247     double len = dist(Point(0, 0), Point(x, y));
248     return Vector(x/len, y/len);
249 }
250
251 Vector Vector::opposite() const {
252     return Vector(-x, -y);
253 }
254
255 double Vector::cross(const Vector& o) const {
256     return x * o.y - y * o.x;
257 }
258
259 double Vector::length() const {
260     return dist(Point(x, y), Point(0, 0));
261 }

```

## J – Jitter Minimization

اول آرایه را مرتب می‌کنیم پس از این به بعد داریم:

$$a_1 < a_2 < \dots < a_{2 \times N}$$

حال اگر عددی مثل  $dis$  وجود داشته باشد که پاسخ مسئله باشد آنگاه  $dis$  برابر  $a_x - a_1$  نیز هست که حتی می‌توان به آسانی نشان داد  $1 \leq x \leq N$  . پس به ازای همه حالات ممکن برای  $dis$  چک می‌کنیم که آیا معتبر هست یا نه. که این کار را می‌توان حریصانه انجام داد هر بار کوچکترین عضو جفت نشده را در نظر گرفت و با  $dis$  به علاوه آن جفت کرد. اگر همچین عضوی وجود نداشت پس آن نامعتبر است.

پیچیدگی زمانی:

$$\mathcal{O}(T \times (N \log N + N \times N \log N))$$

```

1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 int main()
7 {
8     ios_base::sync_with_stdio(false);
9     cin.tie(0);
10    while(true)
11    {
12        int N;
13        cin >> N;
14        if(N == 0) return 0;
15        int A[2*N];
16        bool paired[2*N];
17        for(int i = 0; i < 2*N; i++)
18            cin >> A[i];
19        sort(A, A+2*N);
20        for(int i = 1; ; i++)
21        {
22            if(i == N+1)
23            {
24                cout << "Unreliable Network\n";
25                break;
26            }
27            int dis = A[i]-A[0];
28            memset(paired, 0, sizeof(paired));
29            int p = 0, l = 0;
30            while (l < 2*N)
31            {
32                paired[l] = true;
33                auto it = lower_bound(A, A+2*N, dis+A[l]);
34                if(it == A+2*N or *it != dis+A[l])break;
35                paired[it-A] = true;
36                while(l < 2*N and paired[l])l++;
37                p++;
38            }
39
40            if(p == N)
41            {
42                cout << dis << '\n';
43                break;
44            }
45
46        }
47    }
48 } //Thank God

```

## K – Traffic Lights

برای حل سؤال از روش برنامه‌نویسی پویا استفاده می‌کنیم.

ما  $dp[i][1]$  را که  $1 \leq i \leq n$  را تعداد بیشینه روزهای انتخابی بین روز ۱ تا  $i$  در حالتی که آخرین روز سبز انتخاب کرده باشیم، در نظر می‌گیریم.

برای زرد و  $dp[i][2]$  برای قرمز به طور مشابه تعریف می‌شود. پاسخ مسئله برابر بیشینه  $dp[n][x]$  خواهد بود که  $0 \leq x \leq 2$

برای پایه قرار می‌دهیم  $dp[0][0] = dp[0][1] = dp[0][2] = 0$  روز نام سبز باشد داریم:

$$dp[i][0] = \max(dp[i-1][0], dp[i-1][2]) + 1$$

و نیز اگر روز  $i$  سبز نباشد داریم:

$$dp[i][0] = dp[i-1][0]$$

و به طور مشابه برای زرد و قرمز نیز چنین است.

پیچیدگی زمانی:

$$\mathcal{O}(T \times N)$$

```

1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 int main()
7 {
8     int T;
9     cin >> T;
10    for(int test = 0; test < T; test++)
11    {
12        int n;
13        cin >> n;
14        string s;
15        cin >> s;
16        int dp[n+1][3];
17        dp[0][0] = dp[0][1] = dp[0][2] = 0;
18        for(int i = 0; i < n; i++)
19        {
20            dp[i+1][0] = dp[i][0];
21            dp[i+1][1] = dp[i][1];
22            dp[i+1][2] = dp[i][2];
23            if(s[i] == 'G')dp[i+1][0] = max(dp[i][0],dp[i][2])+1;
24            else if(s[i] == 'Y')dp[i+1][1] = max(dp[i][0],dp[i][1])+1;
25            else dp[i+1][2] = max(dp[i][1],dp[i][2])+1;
26        }
27        cout << max(dp[n][0], max(dp[n][1], dp[n][2])) << '\n';
28    }
29 }
30
31 //Thank God

```

## L – Your House Have Ant

نکته سؤال این است که می‌توانید فرض کنید راهرو تنگی وجود ندارد و مورچه‌ها آزادانه در جهت دلخواه حرکت می‌کنند! پس صرفاً کافیست به ازای هر مورچه مسافتی را که باید طی کند تا از تونل خارج شون را با هم جمع کنید.

پیچیدگی زمانی:

$$\mathcal{O}(T \times |S|)$$

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int t;
6 string s;
7
8 int main()
9 {
10     cin >> t;
11     while (t--)
12     {
13         cin >> s;
14         int ans = 0;
15         int l = s.size();
16         for (int i = 0; i < l; i++)
17             if (s[i] == '<')
18                 ans += i;
19             else if (s[i] == '>')
20                 ans += l - i - 1;
21         cout << ans << endl;
22     }
23     return 0;
24 }
```

## M – Concert

در ابتدا فرض می‌کنیم جواب 1 – نباشد (حالی که جواب 1 – است به سادگی به دست می‌آید). حال می‌دانیم جواب مسئله  $K$  – مین عدد اول دوست) در مبنای 3 کمتر از  $M = 30 > \frac{\log(10^{13})}{\log(3)}$  رقم دارد و همچنین جمع ارقام آن کمتر از  $S \leq 2 \times M = 30$  خواهد بود.

حال به کمک رابطه بازگشتی جواب را محاسبه می‌کنیم. برای اینکار به تابع

$$f(K, CntDigits, Before)$$

نیاز داریم که برای ما محاسبه کند که  $K$  – مین عدد شباهولدوست که حداقل  $CntDigits$  رقم در مبنای 3 دارد، کدام است. به عددی شباهولدوست می‌گوییم که جمع ارقامش به علاوه  $Before$  عددی اول شود. توجه کنید که اگر  $Before = 0$  آنگاه ما  $K$  – مین عدد اول دوست را خواهیم داشت. پس  $f(K, M, 0)$  جواب مسئله خواهد بود.

همچنین ما برای حل مسئله به

$$cnt[i][j] (0 \leq i \leq M, 0 \leq j \leq S)$$

که تعداد اعداد حداقل  $i$  رقمی (قاعده‌تاً در مبنای 3) با جمع ارقام  $j$  را می‌شمارد، نیاز داریم که طریق به دست آوردن آن به شرح زیر است:

$$cnt[0][0] = 1, cnt[i][j] = cnt[i - 1][j] + cnt[i - 1][j - 1] + cnt[i - 1][j - 2]$$

برای محاسبه  $f(K, CntDigits, Before)$   $f$  ما هر بار تلاش می‌کنیم مقدار پارزش‌ترین رقم جواب را مشخص کنیم. به طور دقیق‌تر فرض کنید تعداد اعداد شباهولدوست در حالی که پارزش‌ترین رقم 0 است  $x$ , 1 است  $y$  و 2 است  $z$  باشد. حال اگر  $x \leq K \leq x + y$  و اگر  $y < K \leq x + z$  باشد. آنگاه پارزش‌ترین رقم 0 و اگر  $y < K \leq x + y + z$  پارزش‌ترین رقم 3 خواهد بود (در غیر این صورت جواب 1 – است).

برای اینکه بدانیم چند عدد شباهولدوست با پارزش‌ترین رقم  $w$  داریم کافی است عبارت زیر را حساب کنیم:

$$\sum_{p_i \in P} cnt[CntDigits - 1][p_i - w - before]$$

که منظور از  $P$  مجموعه اعداد اول بین 0 تا  $S$  است.

حال جواب بدین شکل بدست می‌آید:

$$\begin{aligned}
 w = 0 : f(K, CntDigits, before) &= f(K, CntDigits - 1, before) \\
 w = 1 : f(K, CntDigits, before) &= 3^{CntDigits-1} + f(K - x, CntDigits - 1, before + 1) \\
 w = 2 : f(K, CntDigits, before) &= 2 * 3^{CntDigits-1} + f(K - x - y, CntDigits - 1, before + 2)
 \end{aligned}$$

پیچیدگی زمانی:

$$\mathcal{O}(\log MAX_N^2 + T \times \log MAX_N \times \frac{\log MAX_N}{\log \log MAX_N})$$

نماینده تعداد اعداد اول ۱ تا  $\frac{\log MAX_N}{\log \log MAX_N}$  است.  $\log MAX_N$  برای محاسبه  $cnt[i][j]$  که

```

1 // In the name of Allah
2 #include<bits/stdc++.h>
3
4 using namespace std;
5
6 typedef long long ll;
7
8 const ll N = 30; // N = MAX_DIGIT_IN_BASE_3
9 vector<ll> primes = {2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59};
10 ll dp[N][N*2], n, k, ans;
11
12 ll Find(ll k, ll digit, ll before)
13 {
14     if(digit == -1)
15         return 0;
16     ll last_sum = 0, sum = 0;
17     for(ll d = 0; d < 3; d++)
18     {
19         for(auto p: primes)if(p >= d+before)
20             sum += dp[digit][p-d-before];
21         if(sum >= k)
22             return d*pow(3, digit)+Find(k-last_sum, digit-1, before+d);
23         last_sum = sum;
24     }
25     return -1LL;
26 }
27
28 int main()
29 {
30     memset(dp, 0, sizeof(dp));
31     dp[1][0] = dp[1][1] = dp[1][2] = 1;
32     for(int i = 0; i < N; i++)dp[i][0] = 1;
33     for(int i = 0; i < N; i++)dp[i][1] = i;
34     for(int i = 2; i < N; i++)
35         for(int j = 2; j < 2*N; j++)
36             dp[i][j] = dp[i-1][j]+dp[i-1][j-1]+dp[i-1][j-2];
37
38     int T;
39     cin >> T;
40     for(int test = 0; test < T; test++)
41     {
42         cin >> n >> k;
43         ans = Find(k, N-1, 0);
44         if(ans > n)
45             ans = -1;
46         cout << ans << '\n';
47     }
48 }
49
50 //Thank God

```

امیدوارم راه حلها مفید بوده باشند. اگر پیشنهاد یا سؤالی داشتین، حتماً در نظرات مطرح بفرمایید.

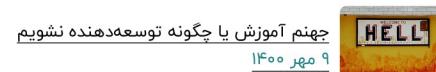


محمدپارسا الهی منش



ممکن است علاقهمند باشد

پریا زدیدترین‌ها



جدیدترین‌ها



eNAMAD.ir

جهانی

رویدادها

منابع

محصولات

[ماشین حساب حقوق برنامه نویسان](#)

[آمارهای دنیای برنامه نویسی](#)

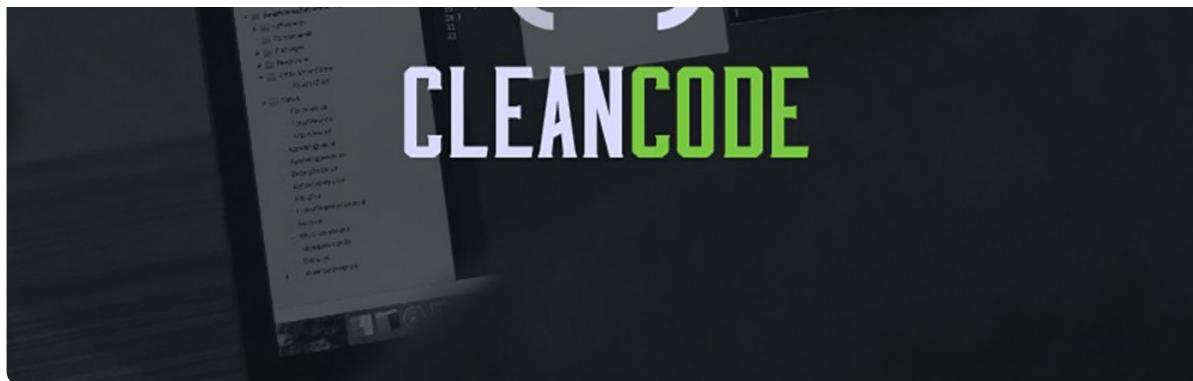
[عضویت در خبرنامه](#)

ساخته شده با افتخار در ایران | ۱۳۹۴ - ۱



Quera

[کدکاب](#)  
[آموزش برنامه‌نویسی](#)  
[نمایشگاه کارآموزشو](#)  
[آگهی‌های استخدام](#)  
[هکاتون کدآپ](#)  
[سؤالات برنامه‌نویسی](#)  
[مسابقات](#)  
[تربیس وی](#)  
[ب کوئرا](#)  
[کلاس‌ها](#)  
[همکاری با ما](#)  
[پلتفرم استخدامی](#)  
[تماس با ما](#)  
[درباره ما](#)  
[قوانین و مقررات](#)  
[همایت از مسابقات](#)



[در ستایش کد تمیز؛ +۱۰۰ توصیه از عمو باب](#)

۱ هفته پیش

نسرين نادری



# بوت‌کمپ‌های کوئرای‌کالج

## BootCamp

**Data Analysis**

**Software Engineering**

**Front-End Development**

- تسهیل روند استخدام
- تجربه‌ی کار تیمی
- تجربه‌ی صنعت محور
- منتورشیپ روزانه
- پروژه‌های واقعی
- محتوای جامع

۱۴۰۰ | مهلت ثبت‌نام: ۳۰ بهمن آغاز آنلاین

**Quera College**

[ثبت‌نام بوت‌کمپ‌های آنلاین برنامه‌نویسی کوئرای‌کالج شروع شد](#)

۱ هفته پیش

کوئرای بلاگ



A large, stylized graphic of binary code blocks forming a mountain-like shape against a dark background. The blocks are composed of binary digits (0s and 1s) and are arranged in a way that suggests a landscape or a city skyline.

卷之三



ویداد Database LevelUp

جذب ۲۰

اشتاك دا 

۱۳۹۱/۸/۲۷



قديمه (ت ٢٠١)



۳ دیدگاه

flu انتظار تصویب ① ۵ ثانیه قبل  متن اسفاره فرمت input سه عال A شتابید

فاطمه (ع) روز قبا

سلام، حتماً باید پاسخ ها به زبان سی یا سی پلاس باشن؟ من با پایتون نوشتم و تست کردم درست بود ولی برام پاسخ نادرست میزنه! این میتواند مشکل باشد

8

 ① همین الان flu انتظار تصویبپاسخ به  فاطمه

نه فرمت input رو بد نوشتن یا نه سر هر cin تو cpp هم میرفت خط بعد تو سوال ها. حواسمنون باید به اینجور مسائل باشه

 ④ روز قبل محمدپارسا الهیمنشپاسخ به  فاطمه

نویسنده

سلام و عرض وقت بخیر  
می تونین روی ارسال تان کلیک کنید و ببینید خطا از چه نوعی هست، خطا به خاطر محدودیت زمان با (Time Limit Exceeded) و خطا به خاطر محدودیت حافظه با (Memory Limit Exceeded) نشون داده میشه.  
در این آزمون B و L تنها سوالاتی هستند که ممکن است راه اصلی با محدودیت های زبان پایتون رو به رو شوند.

پاسخ   ۰  ① همین الان flu انتظار تصویب

پاسخ به محمدپارسا الهیمنش

متاسفانه فرمت input A اشتباه قرار گرفته بود، برای همین با پایتون هم قادر به پاسخگویی نبودیم. فرمت input در صورت سوال برای r a b در یک خط قرار گرفته، در صورتی که باید در سه خط پشت سر هم قرار میگرفت . اشکال ما از split رشته بود که متاسفانه فرمت input اشتباه منظور رو میرسوند

 ④ روز قبل فاطمه

پاسخ به محمدپارسا الهیمنش

خیلی ممنونم از پاسخگویی، کاش راه حل ها رو به زبان های دیگه از جمله پایتون هم بزارید

پاسخ   ۰ 