Разбор задач предметного тура

Математика 9 класс

Задача 1

Числом Кармайкла будем называть всякое составное число n, которое удовлетворяет сравнению $b \equiv b \pmod{n}$ для всех целых b.

а) (4 балла) Докажите, что число n не является числом Кармайкла, если делится на квадрат простого числа, т.е. существует $p \in P$, что $n \,.\, p^2$.

б) (8 баллов) Докажите, что n является числом Кармайкла, если для каждого его простого делителя p выполнены следующие два условия:

(1) n не делится на p^2;

(2) $n - 1$ делится на $p - 1$.

в) (8 баллов) Пусть n — натуральное число. Положим $p = 6n + 1$, $q = 12n + 1$ и $r = 18n + 1$. Докажите, что если p, q, r — простые числа, то произведение $p \cdot q \cdot r$ является числом Кармайкла.

Решение:

а) Так как n — число Кармайкла, то противоречие получится, если мы найдем такое целое b, для которого $b^n \not\equiv b \pmod{n}$. Пусть $b = p$, тогда $p^n \equiv p \pmod{n} \Rightarrow p^n \equiv p \pmod{p^2}$. Противоречие, так как $p^n \equiv 0 \pmod{p^2}$, но $p \not\equiv 0 \pmod{p^2}$.

б) Предположим, что p — простой делитель числа n, покажем тогда $b^n \equiv b \pmod{p}$. Если $b \,.\, p$, то очевидно. Иначе, по малой теореме Ферма $b^{p-1} \equiv 1 \pmod{p}$. Пусть $n = (p - 1) \cdot q + 1$, тогда

$$b^n \equiv (b^{p-1})^q \cdot b \equiv b \pmod{p}.$$

Так как $n = p_1 \cdot p_2 \cdot \ldots \cdot p_k$, следовательно $b^n - b \,.\, p_i$. Так как все p_i различны, то

$$b^n - b \,.\, p_1 \cdot p_2 \cdot \ldots \cdot p_k = n.$$

Значит n является числом Кармайкла.

в) Докажем, что выполняются свойства пункта (б).

$pqr - 1 = (18n + 1)(12n + 1)(6n + 1) - 1 = 18n(12n + 1)(6n + 1) + 18n(4n + 1) \,.\, 18n.$

$pqr - 1 = (18n + 1)(12n + 1)(6n + 1) - 1 = 12n(18n + 1)(6n + 1) + 12n(9n + 2) \,.\, 12n.$ $pqr - 1 =$

$= (18n + 1)(12n + 1)(6n + 1) - 1 = 6n(18n + 1)(12n + 1) + 6n(36n + 5) \,.\, 6n.$

Задача 2.

На бесконечную шахматную доску со стороной клетки $2a$ наудачу брошена монета радиуса $r < a$ (центр монеты лежит на доске). Найти вероятность того, что:

а) (5 баллов) монета попадет целиком внутрь одной клетки;

б) (5 баллов) монета пересечет не более одной стороны одной клетки.

Решение:

Посмотрим в какой области может лежать центр монеты внутри каждой клетки:

а) Для того, чтобы монета попадала целиком внутрь одной клетки, ее центр должен лежать внутри центрированного квадрата со стороной $2(a - r)$. Следовательно, вероятность для всей доски будет

$$P = \frac{64 \times 4(a-r)^2}{64 \times 4 \times a^2} = \frac{(a-r)^2}{a^2}$$

б) Для того, чтобы монета пересекла не более одной стороны одной клетки, ее центр должен лежать в одном из четырех прямоугольников (по прямоугольнику $2(a - r) \times r$ возле каждой из сторон). Следовательно, вероятность для всей доски будет

$$P = \frac{64 \times 4 \times 2r(a-r)^2}{64 \times 4 \times a^2} = \frac{2r(a-r)^2}{a^2}$$

Задача 3

На доске $n \times n$ одна из клеток закрашена невидимыми чернилами. Разрешается выделить любой прямоугольник и спросить, есть ли в нем выделенная клетка. За какое минимальное число вопросов можно найти выделенную клетку, если

а) (7 баллов) $n = 4$;

б) (8 баллов) $n = 5$.

Решение:

Закодируем каждую клетку двоичным кодом. Задавая вопрос про любой прямоугольник, мы получаем не больше одного бита (получаем ответ да или нет).

а) Чтобы закодировать 16 клеток, нам нужно хотя бы 4 бита. Следовательно, быстрее чем за 4 вопроса мы не можем найти закрашенную клетку. (пример легко строится: за два вопроса узнаем в каком из четырех квадратов 2×2 лежит закрашенная клетка; за остальные 2 вопроса узнаем уже точное расположение клетки в квадрате 2×2).

б) Чтобы закодировать 25 клеток, нам нужно хотя бы 5 битов. Следовательно, быстрее чем за 5 вопросов мы не можем найти закрашенную клетку. (пример легко строится: выделяем 4×4; если клетка в нем, то по предыдущему пункту находим за 4 вопроса; иначе в оставшейся области выделяем 1×4 и 1×5 за один вопрос определяем в какой части лежит клетка и за оставшиеся 3 вопроса находим точное расположение клетки).

Математика 10 – 11 класс

Задача 1

Числом Кармайкла будем называть всякое составное число n, которое удовлетворяет сравнению $b^n \equiv b \pmod{n}$ для всех целых b.

а) (5 баллов) Докажите, что n является числом Кармайкла, если для каждого его простого делителя p выполнены следующие два условия:

(1) n не делится на p^2;

(2) $n - 1$ делится на $p - 1$.

б) (6 баллов) Докажите, что не существует чисел Кармайкла вида $n = p \cdot q$.

в) (7 баллов) Найдите все числа Кармайкла вида $n = 3 \cdot p \cdot q$, где p и q — простые числа.

Решение:

а) Предположим, что p — простой делитель числа n, покажем тогда $b^n \equiv b \pmod{p}$. Если $b \,.\, p$, то очевидно. Иначе, по малой теореме Ферма $b^{p-1} \equiv 1 \pmod{p}$. Пусть $n = (p - 1) \cdot q + 1$, тогда

$$b^n \equiv (b^{p-1})^q \cdot b \equiv b \pmod{p}.$$

Так как $n = p_1 \cdot p_2 \cdot \ldots \cdot p_k$, следовательно $b^n - b \,.\, p_i$. Так как все p_i различны, то

$$b^n - b \,.\, p_1 \cdot p_2 \cdot \ldots \cdot p_k = n.$$

Значит n является числом Кармайкла.

б) Для того чтобы $n = p \cdot q$ было числом Кармайкла необходимо и достаточно, чтобы $p \not\equiv q$, $(p \cdot q - 1) \,.\, (p - 1)$ и $(p \cdot q - 1) \,.\, (q - 1)$. Получаем

$(p - 1) \cdot q + (q - 1) \,.\, (p - 1) \Longrightarrow (q - 1) \,.\, (p - 1)$.

Аналогично

$p \cdot (q - 1) + (p - 1) \,.\, (q - 1) \Longrightarrow (p - 1) \,.\, (q - 1)$.

Следовательно, $(p - 1) = (q - 1)$. Противоречие.

в) Для того чтобы $n = 3 \cdot p \cdot q$ было числом Кармайкла необходимо и достаточно, чтобы $p \not\equiv q \not\equiv 3$, $(3 \cdot p \cdot q - 1) \,.\, (p - 1)$, $(3 \cdot p \cdot q - 1) \,.\, (q - 1)$ и $(3 \cdot p \cdot q - 1) \,.\, (3 - 1)$. Из последней делимости следует, что p, q – нечетные простые числа. Будем считать, что $p > q > 3$. Получаем

$3(p - 1)q + (3q - 1) \,.\, (p - 1) \Longrightarrow (3q - 1) \,.\, (p - 1)$.

Так как $p > q$, то возможны только два варианта:

1) $3q - 1 = p - 1 \Longrightarrow 3q = p$. Противоречие.

2) $3q - 1 = 2(p - 1) \Longrightarrow 3q + 1 = 2p$. Аналогично можно получить

$(3p - 1) \cdot (q - 1) \Longrightarrow 2(3p - 1) \cdot (q - 1) \Longrightarrow (9q + 1) \cdot (q - 1) \Longrightarrow 10q \cdot (q - 1)$

А так как $(q, q - 1) = 1$, то $10 \cdot q - 1$. Получаем единственный возможный вариант $q = 11$, $p = 17$ и $n = 561$.

Задача 2

Через специальный кабель Алиса передает случайную последовательность битов $x_1x_2 \cdots x_n$ своему однокласснику Бобу, который на выходе получает последователь- ность $y_1y_2 \cdots y_n$. Боб получает искаженное сообщение в следствии того, что кабель неидеальный (имеется вероятностный шум):

(1) бит 0 с вероятностью ε изменяется на 1;

(2) бит 1 с вероятностью ε изменяется на 0.

Обозначим через α — вероятностное распределение, передаваемых Алисой символов (будем считать, что P (бит a_i равен 0) = p и P (бит a_i равен 1) = $1 - p$).

а) (4 балла) Выразите через p и ε вероятностное распределение β, получаемых Бобом битов;

б) (4 балла) Вычислите величину $Q = \max_\alpha H(\beta)$, где H — энтропия Шеннона;

в) (6 баллов) Вычислите пропускную способность кабеля $R = \max_\alpha I(\alpha : \beta)$ и выясните при каких ε она максимальна и минимальна (I – взаимная информация).

Решение:

а)

$$P(b_i = 0) = P(a_i = 0) \cdot (1 - \varepsilon) + P(a_i = 1) \cdot \varepsilon = p \cdot (1 - \varepsilon) + (1 - p) \cdot \varepsilon.$$

$$P(b_i = 1) = P(a_i = 1) \cdot (1 - \varepsilon) + P(a_i = 0) \cdot \varepsilon = (1 - p) \cdot (1 - \varepsilon) + p \cdot \varepsilon.$$

б)

$$Q = \max_\alpha H(\beta) = \max_p H(\beta)$$

При $p = 1$ получаем $P(b_i = 0) = P(b_i = 1) = 0{,}5$, следовательно? при таком p энтропия будет максимальна

$$H(\beta) = \log_2 2 = 1. \text{ Значит } Q = 1.$$

в) $R = \max_\alpha I(\alpha : \beta) = \max_\alpha\big(H(\beta) - H(\beta|\alpha)\big) = \max_\alpha\big(H(\beta) - H(\beta|a_i = 0)\, P(a_i = 0) - H(\beta|a_i = 1) P(a_i = 1)\big) = \max_\alpha H(\beta) - \varepsilon \log_2 \frac{1}{\varepsilon} - (\varepsilon - 1) \log_2 \frac{1}{\varepsilon - 1} = 1 - h(\varepsilon)$

Функция $h(\varepsilon)$ принимает минимальное значение при $\varepsilon = 0$ или 1 и максимальное значение при $\varepsilon = 0{,}5$.

Задача 3.

Пусть p и q — различные простые числа и $n = p \cdot q$. Блок b будем называть неподвижным относительно RSA-системы с открытым ключом (n, e), если он не изменяется после кодирования, т.е. $E(b) = b$. Определите число неподвижных блоков относительно RSA-системы в случае:

а) (8 баллов) $e = 3$, $p = 3$ и $q > 3$;

б) (10 баллов) $e = 5$, $p > 5$ и $q > 5$.

Решение:

Так как b неподвижный блок относительно RSA-системы с открытым ключом (n, e), то $b^e \equiv b \pmod n$. Следовательно, $b^e \equiv b \pmod p$ и $b^e \equiv b \pmod q$. Решим каждое из этих сравнений и применим китайскую теорему об остатках. Получим, что общее число неподвижных блоков — это произведение числа решений $b^e \equiv b \pmod p$ и $b^e \equiv b \pmod q$.

а) $b^3 \equiv b \pmod 3$ имеет ровно 3 решения (любой остаток по модулю 3 подходит); $b^3 \equiv b \pmod q$ имеет также ровно 3 решения ($b(b - 1)(b + 1) \cdot q \Longrightarrow b \equiv 0, \pm 1 \pmod q$) В итоге получаем 9 неподвижных блоков.

б) $b^5 \equiv b \pmod p \Longrightarrow b(b - 1)(b + 1)(b^2 + 1) \cdot p$. Получаем 3 решения точно имеются $b \equiv 0, \pm 1 \pmod p$. Остальные два зависят от того является ли -1 квадратичным вычетом по модулю p. Если $p = 4k + 1$, то -1 —

квадратичный вычет ⟹ решений будет 5, а если $p = 4k + 3$, то −1 — квадратичный невычет ⟹ решений будет 3. Аналогичное утверждение можно сделать для делителя q.
В итоге, в зависимости от того какие остатки дают p и q при делении на 4, получаем 9, 15 или 25 неподвижных блоков.

Информатика

Задача 1: Платежные каналы

В технологии блокчейн Probocoin для передачи средств используются односторонние платежные каналы, каждый из которых связывает два узла сети ProboNetwork.

Средства из узла *a* в узел *b* в этой сети передаются либо через прямой платежный канал, либо через цепочку платежных каналов.

Используя данные о существующих платежных каналах, ответьте на следующий вопрос:

Есть ли возможность передать средства от человека *X* человеку *Y*, не создавая новые каналы?

Человек *X* представляет собой узел *s*, человек *Y* представляет собой узел *t*.

Формат входных данных

В первой строке вводятся два целых числа n, m ($1 \le n \le 10^5$, $1 \le m \le 2\ 10^5$) — количество узлов и каналов в системе.

В следующих m строках вводятся по два целых числа a_i и b_i ($1 \le a_i, b_i \le n$, a_i b_i) — начало и конец i-го канала.

В последней строке вводятся два целых числа s и t ($1 \le s, t \le n$, s ƒ= t).

Формат выходных данных

Выведите *Yes*, если можно передать средства от человека *X* человеку *Y*, не создавая новые каналы. В противном случае выведите *No*.

Примеры

Пример №1

Стандартный ввод
3 1 1 3 1 3
Стандартный вывод
Yes

Пример №2

Стандартный ввод
3 1 3 1 1 3
Стандартный вывод
No

Решение

Первая задача сводится к тому, что у нас есть ориентированный граф состоящий из n вершин и m ребер и нам нужно проверить есть ли путь из вершины s в вершину t.

Это стандартная задача и решается она с помощью алгоритма DFS или алгоритма BFS.

Время работы: $O(n + m)$

Память: $O(n + m)$

Пример программы

Ниже представлено решение на языке C++

```
#include<bits/stdc++.h>
usingnamespacestd;
typedefpair<int,int>pii;
#define pb push_back
#define sz(x) ((int)x.size())
#define fi first
#define se second



constintMAXN=100000+10;



vector<int>g[MAXN];
charused[MAXN];
voiddfs(intv){
	used[v]=1;
	for(intto:g[v]){
		if(!used[to]){
			dfs(to);
		}
	}
}

voidsolve(){
	intn,m;
	cin>>n>>m;
	for(inti=1;i<=m;++i){
		inta,b;
		cin>>a>>b;
		g[a].pb(b);
	}
	ints,t;
	cin>>s>>t;
	dfs(s);
	if(used[t]){
		cout<<"Yes"<<endl;
	}else{
		cout<<"No"<<endl;
	}

}



intmain(){
	ios_base::sync_with_stdio(0);
	cin.tie(0);
	cout.tie(0);
	solve();

}
```

Задача 2: Комиссия

В сети ProboNetwork каждый платежный канал имеет свою комиссию, которая взимается при передаче средств по каналу. Эта комиссия не зависит от переводимой по платежному каналу суммы и одинакова для всех переводов средств по платежному каналу.

Также в сети ProboNetwork можно добавлять новые платежные каналы. За добавление нового канала потребуется заплатить q_0. Но добавлять можно не все каналы, а только те, которые разрешены системой. Всего существует q таких каналов.

Используя данные о существующих платежных каналах и списке каналов, которые разрешены к добавлению системой, ответьте на следующий вопрос:

Какова минимальная сумма, которую придется заплатить, чтобы произвести передачу денег из узла s в узел t.

Формат входных данных

В первой строке вводятся два целых числа n, m ($1 \le n \le 10^5$, $0 \le m \le 10^5$) — количество узлов и каналов в системе.

В следующих m строках вводятся по три целых числа a_i, b_i, c_i ($1 \le a_i, b_i \le n$, $a_i \neq b_i$, $1 \le c_i \le 10^4$) — начало, конец и комиссия i-го канала.

В следующей строке вводятся два целых числа s и t ($1 \le s, t \le n$, $s \neq t$).

В следующей строке вводятся два целых числа q и q_0 ($1 \le q \le 2 \cdot 10^5$, $0 \le q_0 \le 1000$).

В следующих q строках вводятся по три целых числа a_i, b_i, c_i ($1 \le a_i, b_i \le n$, $a_i \neq b_i$, $1 \le c_i \le 10^4$) — начало, конец и комиссия i-го канала в списке.

Формат выходных данных:

Выведите ответ на вопрос или –1, если передать средства из узла s в узел t невозможно.

Примеры

Пример №1

Стандартный ввод
3 1 1 3 5 1 3 2 6 1 2 1 2 3 1
Стандартный вывод
5

Пример №2

Стандартный ввод
3 1 1 2 5 1 3 1 1 1 3 2
Стандартный вывод
3

Пример №3

Стандартный ввод
3 1 1 2 1 3 1 1 1 2 3 1
Стандартный вывод
-1

Решение

Можно заметить, что ребра, которые можно добавить в граф за стоимость q_0, можно представить в виде ориентированных ребер в графе со стоимостью $q_0 + c_i$.

Таким образом, у нас есть ориентированный взвешенный граф состоящий из n вершин и $m + q$ ребер. И нам нужно найти длину кратчайшего пути из вершины s в вершину t или сказать, что пути из s в t не существует.

Это стандартная задача и решается она с помощью алгоритма Дейкстры для разреженных графов.

Для решения первых двух подзадач можно было воспользоваться обычным алгоритмом Дейкстры, алгоритмом Форда-Беллмана или алгоритмом Левита.

Время работы: $O((m + q) \cdot \log2(n))$

Память: $O(n + m + q)$

Пример программы

Ниже представлено решение на языке C++

```
1   #include<bits/stdc++.h>
2   usingnamespacestd;
3   typedeflonglongll;
4   typedefpair<int,int>pii;
5   typedefpair<ll,ll>pll;
6   #define pb push_back
7   #define sz(x) ((int)x.size())
8   #define fi first
9   #define se second
10  constllinf_ll=2e18;
11
12  constintMAXN=100000+10;
13
```

```
14
15  vector<pii>g[MAXN];
16  lldis[MAXN];
17  voiddey(intst){
18          fill(dis,dis+MAXN,inf_ll);
19          dis[st]=0;
20          set<pll>s;
21
22          s.insert({0,st});
23          while(!s.empty()){
24                  intv=s.begin()->se;
25                  s.erase(s.begin());
26                  for(piix:g[v]){
27                          intto=x.fi;
28                          intlen=x.se;
29                          if(dis[to]>dis[v]+len){
30                                  s.erase({dis[to],to});
31                                  dis[to]=dis[v]+len;
32                                  s.insert({dis[to],to});
33                          }
34                  }
35          }
36  }
37
38
39  voidsolve(){
40  intn,m;
41          cin>>n>>m;
42          for(inti=1;i<=m;++i){
43                  inta,b,c;
44                  cin>>a>>b>>c;
45                  g[a].pb({b,c});
46          }
47          ints,t;
48          cin>>s>>t;
49          intq,q0;
50          cin>>q>>q0;
51          for(inti=1;i<=q;++i){
52                  inta,b,c;
53                  cin>>a>>b>>c;
54                  g[a].pb({b,c+q0});
55          }
56          dey(s);
57          if(dis[t]==inf_ll){
58                  cout<<-1<<endl;
59          }else{
60                  cout<<dis[t]<<endl;
61          }
62
63  }
64
65  intmain(){
66          ios_base::sync_with_stdio(0);
67          cin.tie(0);
68          cout.tie(0);
69          solve();
70
71  }
```

Задача 3: Дерево блоков

Программист Иван создал свою версию *blockchain* технологии. Ему показалось странным, что для выстраивания цепочки блоков большинство используют линейный список, хотя по его мнению, использование дерева позволило бы добиться большей производительности *blockchain* сети.

В его версии *blockchain* в каждый определенный момент времени цепочка блоков состоит из n блоков, которые образуют дерево. Каждому ребру в дереве присваивают случайное число val_i. Для верификации блоков используют XOR всех значений на пути между текущей вершиной и другой случайной вершиной.

Система Ивана начала работать, но Иван стал замечать, что некоторых значения стали слишком часто совпадать. И теперь он боится, что совершил ошибку в коде.

Для этого он решил проверить задачу на специальных запросах: он выбирает вершину v_i и значение q_i и хочет узнать какое количество вершин u_i, таких что XOR всех значений на пути от v_i к u_i равен q_i.

Формат входных данных

В первой строке вводятся два целых числа n, q ($2 \le n \le 10^5$, $1 \le q \le 10^5$) — количество блоков и запросов.

В следующих $n - 1$ строках вводятся по три целых числа a_i, b_i, val_i ($1 \le a_i, b_i \le n$, $a_i = b_i$, $0 \le val_i \le 10^9$), которые означают, что между вершинами a_i и b_i в дереве есть ребро и ему присвоено число val_i.

Гарантируется, что ребра образуют дерево.

В следующих q строках вводятся по два целых числа v_i, q_i ($1 \le v_i \le n$, $0 \le q_i \le 10^9$).

Формат выходных данных

Выведите q чисел, i-е из которых — это количество вершин u_i, таких что XOR всех значений на пути от v_i к u_i равен q_i.

Примеры

Пример №1

Стандартный ввод
4 3 1 2 1 2 3 2 2 4 4 1 1 1 3 1 7
Стандартный вывод
1 1 0

Решение

Наивное решение:

Будем запускать DFS каждый раз во время запроса, и считать количество вершин во время обхода поддерживая текущее число xor.

Время работы: $O(q \cdot n)$

Память: $O(n + q)$

Полное решение:

Давайте выберем одну вершину корнем нашего дерева (например, 1). Теперь запустим DFS из корня и предподсчитаем значение *pref* [*v*].

pref [*v*] - *xor* всех ребер на пути, от корня до вершины *v*.

Для решения задачи нужно воспользоваться следующим свойством операции XOR:

a xor a = 0

Заметим, что если мы возьмем две вершины v_1 и v_2, то *xor* ребер на пути от v_1 до v_2 будет равен *pref* [v_1] *xor pref* [v_2]. Потому что ребра, которые не принадлежат пути, попадут два раза и не будут учитываться.

Заметим, что *pref* [v_1] *xor pref* [v_2] = q_i ⇔ *pref* [v_2] = q_i *xor pref* [v_1]

Следовательно, для каждого запроса нам нужно найти количество вершин, у которые значение *pref* равно q_i *xor pref* [v_1].

Это можно решить с помощью обычного словаря.

Время работы: $O((n + q) \cdot \log2(n)$

Память: $O(n + q)$

Пример программы

Ниже представлено решение на языке C++

```
1   #include<bits/stdc++.h>
2   usingnamespacestd;
3   typedeflonglongll;
4   typedefpair<int,int>pii;
5   typedefpair<ll,ll>pll;
6   #define pb push_back
7   #define sz(x) ((int)x.size())
8   #define fi first
9   #define se second
10  constllinf_ll=2e18;
11  constintMAXN=1e5+1;
12
13
14  vector<pii>g[MAXN];
15  intval[MAXN];
16
17  voiddfs(intv,intp){
18          for(piix:g[v]){
19                  intto=x.fi;
20                  if(to==p)
21                          continue;
22                  intlen=x.se;
23                  val[to]=val[v]^len;
24                  dfs(to,v);
25          }
26  }
27
28  voidsolve(){
29          intn,q;
30          cin>>n>>q;
31          for(inti=1;i<=n-1;++i){
32                  inta,b,c;
33                  cin>>a>>b>>c;
34                  g[a].pb({b,c});
35                  g[b].pb({a,c});
36          }
37          map<int,int>mp;
38          dfs(1,-1);
39
40          for(inti=1;i<=n;++i){
41                  mp[val[i]]++;
42          }
```

```
43
44              for(inti=1;i<=q;++i){
45                      intv1,q1;
46                      cin>>v1>>q1;
47                      intres=val[v1]^q1;
48
49                      cout<<mp[res]<<"\n";
50              }
51
52      }
53
54      intmain(){
55              ios_base::sync_with_stdio(0);
56              cin.tie(0);
57              cout.tie(0);
58              solve();
59
60      }
```