Разборы задач №3

Бинарный поиск и перебор

Содержание

3 – Как можно быстрее – codeforces 701D

10 – Осада Вальгаллы – codeforces 975C

14 – Эклеры – codeforces 991C

18 – Планирование экспедиции – codeforces 1011B

23 – Лавочки – codeforces 1042B

26 – Шляпа – codeforces 1019B

Как можно быстрее – codeforces 701D

На каникулах n школьников решили сходить на экскурсию и собрались все вместе. Им необходимо преодолеть путь длиной l метров. Каждый из школьников будет идти со скоростью v_1. Чтобы быстрее добраться до экскурсии было принято решение арендовать автобус, вместимостью k человек (то есть одновременно в автобусе не может находиться более k человек) и скоростью v_2. Чтобы никого из школьников не укачало, каждый из них хочет сесть в автобус не более одного раза.

Перед вами стоит задача определить минимальное время, по истечении которого все n школьников смогут добраться до места проведения экскурсии. Считайте, что посадка и высадка пассажиров, а также разворот автобуса, происходят мгновенно

Как можно быстрее – codeforces 701D

Входные данные
В первой строке следует пять целых положительных чисел n, l, v_1, v_2 и k ($1 \le n \le 10\,000, 1 \le l \le 109, 1 \le v_1 < v_2 \le 109, 1 \le k \le n$) — количество школьников, расстояние от места сбора до места проведения экскурсии, скорость каждого школьника, скорость машины и количество мест в машине.

Выходные данные
Выведите вещественное число — минимальное время, по истечении которого все школьники доберутся до места проведения экскурсии. Относительная или абсолютная погрешность ответа не должна превышать 10 - 6.

Как можно быстрее – codeforces 701D

Примеры

Входные данные	Выходные данные
5 10 1 2 5	5.0000000000
3 6 1 2 1	4.7142857143

Как можно быстрее – codeforces 701D

Очевидно, что искомый ответ лежит в пределах от 0 до $n*l$.

Разобьем школьников на группы по вместимости автобуса: $(n+k-1)/k$.

Чтобы получить искомое время, мы можем использовать бинарный поиск по ответу. Если целевая функция будет возвращать истину, то мы будем сдвигать правую границу поиска, а если ложь – левую.

Как можно быстрее – codeforces 701D

Рассмотрим время, которое требуется проехать на автобусе, чтобы первая группа дошла как раз за время mid. это время равно $\frac{T-(v_2*T-(l-posM))}{v_2-v_1}$, где изначально T равно mid, $posM$ равно 0 (в этой позиции мы будем поддерживать позицию школьников, которые еще не ехали в автобусе). Затем нужно аккуратно пересчитывать T и posM для каждой следующей группы, учитывая, что автобус должен вернуться обратно, чтобы забрать следующую группу. Если в какой-то момент $v_1 * T$ стало меньше, чем $l - posM$, либо T стало меньше 0, нужно вернуть *false*. Если все группы школьников успеют добраться за время T, нужно вернуть *true*. Также нужно не забыть, что после того, как автобус отвезет последнюю группу школьников, возвращаться обратно ему не нужно.

Как можно быстрее – codeforces 701D

```
bool check(double T, int n,int l,int v1,int v2,int k)
{
 double start = 0;
 double t0 = 0;
 double left = n;
 if (T*v1 >= l)
   return true;

 while (left > 0) {
     double t = l-start+v1*(t0-T);
```

Золотая система счисления – codeforces 457A

```
#include <bits/stdc++.h>
using namespace std;
int main() {
   string a,b;
   cin>>a>>b;
   int n=max(a.size(),b.size());
   if(a.size()<n) a=string(n-a.size(),'0')+a;
   else if(b.size()<n) b=string(n-b.size(),'0')+b;
   int c[100002];
   for( int i=0;i<n;i++){
       c[i]=a[i]-b[i];
   }



   for( int i=0;i<n-2;i++){
       if(c[i]>2) {
           cout<<">";
```

Осада Вальгаллы – codeforces 975C

Ивар Бескостный — великий лидер. Он пытается захватить Каттегат, в данный момент находящийся под контролем Лагерты. Битва началась, и волны воинов Ивара гибнут одна за другой.

У Ивара n воинов, он выставляет их вдоль прямой напротив главных ворот так, что i-й воин стоит сразу за $(i-1)$-м воином. Первый воин возглавляет атаку.

Каждый атакующий воин может выдержать до a_i стрел, прежде чем он падёт, где a_i — сила i-го воина.

Лагерта приказывает своим воинам выпустить k_i стрел в течение i-й минуты, стрелы одна за одной поражают первого всё ещё стоящего воина. После того, как все воины Ивара падут и стрелы, находящиеся в воздухе в данный момент, пролетят, Тор бьёт по земле своим молотом и все воины Ивара получают свои силы назад и возвращаются в битву. Другими словами, если все воины умрут в минуту t, в конце этой минуты t они все встанут и будут сражаться.

Осада Вальгаллы – codeforces 975C

Входные данные
Первая строка содержит два целых числа n и q $(1 \leq n, q \leq 200000)$ — число воинов Ивара и длительность боя в минутах.
Вторая строка содержит n целых чисел $a_1, a_2,\ldots,a_n (1 \leq a_i \leq 109)$, обозначающих силы воинов Ивара.
Третья строка содержит q целых чисел $k_1, k_2,\ldots,k_q$ $(1 \leq k_i \leq 1014)$, i-е из которых означает число стрел k_i, которое будет выпущено в воинов Ивара по приказу Лагерты в минуту i.

Выходные данные
Выведите q строк, i-я из которых содержит число воинов Ивара, находящихся в строю после i-й минуты.

Осада Вальгаллы – codeforces 975C

Примеры

Входные данные	Выходные данные
5 5 1 2 1 2 1 3 10 1 1 1	3 5 4 4 3
4 4 1 2 3 4 9 1 10 6	1 4 4 1

Осада Вальгаллы – codeforces 975C

Сперва определимся с тем, какими будут совокупные силы воинов, для чего просуммируем силы i-того воина и всех предыдущих.

Аналогично поступим со стрелами.

Теперь мы знаем, что за i-тую минуту будет выпущено t стрел, которые убьют h воинов. Если совокупно стрел на i-той больше, чем совокупная сила всех воинов, то мы можем вернуть n - все воины умрут, а потом воскреснут.

Иначе мы можем определить, сколько умерло воинов, и вернуть в качестве i-того значения разницу между числом оставшихся воинов и числом умерших воинов.

Эклеры – codeforces 991C

После успешной сдачи всех зачетов Вася купил себе в подарок коробку, содержащую n сладких эклеров. Вася решил каждое утро есть некоторое одинаковое число эклеров, пока они все не закончатся. Однако сосед Васи, Петя, заметил принесенную Васей коробку и тоже решил насладиться вкусом эклеров.

Теперь процесс поедания эклеров выглядит следующим образом: сначала Вася выбирает число k, одинаковое для всех дней. Затем утром он съедает k эклеров из коробки (или доедает все эклеры, если их осталось меньше k), после этого Петя вечером съедает 10% оставшихся эклеров. Если эклеры еще не закончились, то на следующий день Вася опять съедает k эклеров, а Петя — *10%* от оставшихся и так далее.

Если число эклеров не делится на 10, то Петя округляет «свою» долю в меньшую сторону, например, если в коробке было 97 эклеров, то Петя съест только 9 из них. В частности, если в коробке уже меньше 10 эклеров, то Петя не будет их есть вообще.

Определите, какое наименьшее число k может выбрать Вася такое, что он съест не менее половины от всех n эклеров, которые были в коробке изначально. Заметьте, что число k должно быть натуральным.

Эклеры – codeforces 991C

Входные данные

В первой строке содержится натуральное число n ($1 \leq n \leq 10^{18}$) — изначальное количество эклеров.

Выходные данные

Вывести единственное число — наименьшее значение k, удовлетворяющее Васю.

Примеры

Входные данные	Выходные данные
68	3

Эклеры – codeforces 991C

Очевидно, что если для некоторого k условие задачи выполняется – Вася съест более половины эклеров, то и для любого большего k условие выполнится.

Тогда мы можем решить задачу с помощью бинарного поиска по ответу. Мы будем сдвигать левую границу, если данное k нас не удовлетворяет, и правую – если при данном k условие выполнится.

Эклеры – codeforces 991C

```
bool check(long long k, long long n) {
	long long sum = 0;
	long long cur = n;
	while (cur > 0) {
		long long o = min(cur, k);
		sum += o;
		cur -= o;
		cur -= cur / 10;
	}
	return sum * 2 >= n;
}
```

Планирование экспедиции – codeforces 1011B

Наташа планирует экспедицию на Марс для n человек. Важная задача — обеспечить питанием каждого участника в каждый из дней экспедиции.

Всего на складе доступны m суточных комплектов питания. Каждый комплект характеризуется своим типом a_i.

Каждый из участников в день должен съедать ровно один суточный комплект питания. По причине экстремальных нагрузок, каждый участник в каждый из дней экспедиции должен съедать комплект одного и того же типа. Для разных участников типы съедаемых комплектов могут как различаться, так и совпадать.

Формально, для каждого участника j Наташа должна выбрать тип питания b_j и каждый день j-й участник должен съедать комплект питания типа b_j. Значения b_j у разных участников могут как различаться, так и совпадать.

Какой максимальной продолжительности в днях можно спланировать экспедицию,

Планирование экспедиции – codeforces 1011B

Входные данные

В первой строке записаны два целых числа n и m ($1 \le n \le 100, 1 \le m \le 100$) — количество участников экспедиции и количество суточных комплектов питания на складе.

Во второй строке записана последовательность целых чисел $a_1, a_2, \ldots, a_n (1 \le a_i \le 100)$, где a_i — тип i-го комплекта питания.

Выходные данные

Выведите максимальное количество дней, сколько может продолжаться экспедиция. Если невозможно спланировать экспедицию даже на один день, то выведите 0.

Two's complement – 1 – informatics 750

Примеры

Входные данные	Выходные данные
4 10 1 5 2 1 1 1 2 5 7 2	2
100 1 1	0
2 5 5 4 3 2 1	1
3 9 42 42 42 42 42 42 42 42 42	3

Планирование экспедиции – codeforces 1011B

Сперва определимся с тем, сколько каких комплектов нам досталось.

Очевидно, что ответ не может превышать количество еды.

Тогда мы можем реализовать бинарный поиск.

Будем рассматривать величину $p = \sum_{i=1}^{n} \frac{F[i]}{\frac{l+r}{2}}$, где *F[i]* - кол-во комплектов, *l, r* - нижняя и верхняя граница дней в экспедиции. В первой итерации $l=0, r=m$;

По сути мы рассматриваем то, на людей хватит комплектов питания по группам для количества дней *r*.

Если $p<n$, то сдвигаем r влево, иначе сдвигаем l вправо

Если $r-l=1$, то то делаем то же, что и в пункте а, если $p<n$, то выводим в качестве ответа *l*, иначе - *r*.

Планирование экспедиции – codeforces 1011B

```
int find_res1(int people, int l,  int r, vector<int>F){
int p = 0;
    if (r - l == 1)
    {
        for (unsigned int i = 0; i < F.size(); i++)
            p += F[i] / r;
        if (p >= people)
            return r;
        else
            return l;
    }
    int m = (l + r)/2;
    for (unsigned int i = 0; i < F.size(); i++)
        p += F[i]  / m;

    if (p >= people) return find_res1(people, m, r, F);
    else return find_res1(people, l, m, F);}
```

Лавочки – codeforces 1019B

В берляндском парке есть n лавочек. Про каждую лавочку известно количество людей a_i, которые уже сидят на i-й лавочке. Известно, что в ближайшее время в парк придут ещё m человек, каждый из которых сядет на одну из n лавочек.

Пусть k — это максимальное количество человек, которые будут сидеть на одной лавочке после прихода в парк ещё m человек. Определите минимально возможную величину k и максимально возможную величину k.

Считайте, что никто из посетителей парка не будет вставать с лавочек.

Лавочки – codeforces 1042A

Входные данные

В первой строке следует целое число n ($1 \leq n \leq 100$) — количество лавочек в парке.

Во второй строке следует целое число m ($1 \leq m \leq 10000$) — количество людей, которые ещё придут в парк и сядут на лавочки.

В следующих n строках следует по одному целому числу a_i ($1 \leq a_i \leq 100$) — количество людей, которые изначально сидят на i-й лавочке.

Выходные данные

Выведите минимально возможную величину k и максимально возможную величину k, где k — это максимальное количество человек, которые будут

Лавочки – codeforces 1042A

Примеры

Входные данные	Выходные данные
4 6 1 1 1 1	3 7
1 10 5	15 15
3 6 1 6 5	6 12
3 7 1 6 5	7 13

Лавочки – codeforces 1042A

Для максимума находим наибольшее число людей на лавочке и добавляем туда всех пришедших.

Для минимума в цикле распределяем всех пришедших по наименее занятым лавочкам.

```
int main(){
unsigned int n, m;
   cin>>n>>m;
   int *a=new int [n];
   for (int i=0;i<n;i++)
       cin>>a[i];
   int max=0;
   for (int i=0;i<n;i++)
       if (a[max]<a[i])
           max=i;
   max=a[max]+m;

   while (m>0)
```

Шляпа – codeforces 1019B

Это интерактивная задача.
Имур Ихаков организует клуб по шляпе. На клуб пришло принять участие n человек, где n чётно. Имур усадил их всех в круг и провёл жеребьевку, чтобы разбить ребят на пары, но что-то пошло не так. Участники пронумерованы так, что участники i и $i+1$ ($1 \le i \le n-1$) сидят рядом, а также участники n и 1 сидят рядом. Каждому был выдан листочек с числом так, что у ребят, сидящих рядом, эти числа отличаются ровно на единицу. Предполагалось, что игроки с одинаковыми числами образуют пару, но оказалось, что не все числа встречались ровно дважды.
Как известно, удобнее всего объяснять слова партнёру, когда он сидит напротив. Участники с номерами i и сидят напротив друг друга. Имуру интересно, есть ли люди, сидящие напротив друг друга и имеющие одинаковые числа на своих листочках. Помогите ему найти такую пару людей, если она есть.
Вы можете задавать вопросы вида «какое число написано на листочке у школьника i?». Ваша цель — выяснить, есть ли требуемая пара, сделав не более 60 вопросов.

Шляпа – codeforces 1019B

Входные данные: на вход подаётся одно целое чётное число n ($2 \leq n \leq 100\,000$) — число участников, пришедших на клуб Имура. Вам разрешается задать не более 60 вопросов.

Выходные данные: для того, чтобы узнать число i-го участника ($1 \leq i \leq n$), нужно вывести «? i». После этого ваша программа на вход получит целое число a_i ($-109 \leq ai \leq 109$) — число на листочке i-го человека.

Чтобы сообщить, что ответ найден, требуется вывести «*! i*», где i — номер любого участника из пары ($1 \leq i \leq n$). Если такой пары участников не существует, выведите «! -1». После этого программа должна завершиться. Запрос на вывод ответа не входит в ограничение на 60 запросов. Не забывайте сбрасывать буфер после каждого запроса. Например, на C++ надо использовать функцию *fflush(stdout)*, на Java вызов *System.out.flush()*, на *Pascal flush(output)* и *stdout.flush()* для языка Python.

Шляпа – codeforces 1019B

Примеры

Входные данные	Выходные данные
8 2 2	? 4 ? 8 ! 4
6 1 2 3 2 1 0	? 1 ? 2 ? 3 ? 4 ? 5 ? 6 ! -1

Шляпа – codeforces 1019B

Пусть *a(i)* — листок на числе *i*-го школьника. Введем функцию $b(i) = a(i) - a(i + \frac{n}{2})$. Поскольку n четно, то *b(i)* корректно определена. Заметим два факта: во-первых, $b(i) = -b(i + \frac{n}{2})$, а во-вторых, $|b(i) - b(i+1)| \in \{2,0,2\}$. Задача заключается в поиске i такого, что b(i) = 0.

Из второго наблюдения можно заметить, что все *b(i)* имеют одинаковую четность. Поэтому, посчитаем *b(0)*, и если оно нечетно, то ответа нет, и нужно вывести - 1 — все *b(i)* имеют одинаковую четность (нечетную), и ноль, как число другой четности, в *b(i)* не появится.

Шляпа – codeforces 1019B

Иначе, пусть мы посчитали $b(0)$, и оно оказалось равным какому-то числу x (без ограничения общности $x > 0$). Заметим, что $b\left(0 + \frac{n}{2}\right) = -b(0) = -x$. Поскольку из второго наблюдения мы помним, что все b(i) имеют одинаковую четность, и соседние отличаются не больше, чем на 2, можно воспользоваться дискретной непрерывностью.

Лемма: если есть два индекса i, j со значениями $b(i)$, $b(j)$, то на отрезке между i и j есть все значения той же четности от $min(b(i), b(j))$ до $max(b(i), b(j))$. Действительно, поскольку соседние b изменяются не больше, чем на 2, мы не могли пропустить ни одно число той же четности.

Теперь можно применить двоичный поиск с границами 0 и $\frac{n}{2}$. Сдвиги границ будут производиться в зависимости от знака $b(m)$. Если $b(m)=0$, то поиск прекращается.

Шляпа – codeforces 1019B

```
while(true)
  {
     int mid=(l+r)/2;
     int x=query(mid+hn)-query(mid);
     if(x==0)
        answer(mid);
     if ((a < 0 && x > 0) || (a > 0 && x < 0))
     {      r = mid;
        b = x;
     }
     else{
        l = mid;
        a = x;
     }
  }
```

```
inline int query(int k)
{
   cout<<"? "<<k<<"\n";fflush(stdout);
   int x;cin>>x;
   return x;
}

inline void answer(int k)
{
   cout<<"! "<<k<<"\n";fflush(stdout);
   exit(0);
}
```