9-10 Класс. Задачи 1-3

Hex-операция

Условие.

Мальчик Петя очень любит шестнадцатеричные числа и решил подарить своему другу одно из них. Число оказалось слишком большим, поэтому Петя решил упростить его по следующему алгоритму: нужно складывать все цифры числа до тех пор, пока не получится число, состоящее из одной цифры.
Помогите Пете найти итоговое число.

Формат ввода.

В единственной строке дана строка S $(1 \leq |S| \leq 2 * 10^6)$ состоящая из символов $0123456789abcdef$ - изначальное число Пети.
Гарантируется, что изначальное число не содержит лидирующих нулей.

Формат вывода.

Выведите одну шестнадцатеричную цифру, которую Петя подарит своему другу.

Пример 1

Ввод	Вывод
f	f

Пример 2

Ввод	Вывод
aa	5

Примечание

В первом примере число уже состоит из одной цифры и ничего делать не нужно.

Во втором примере будет следующая цепочка преобразований: aa -> 14 -> 5.

Решения

Язык Python

```
s = input()
n = 0
for c in s:
        n += int(c, 16)

while n >= 16:
        new_n = 0
        while n > 0:
                new_n += n % 16
                n //= 16
        n = new_n

print(hex(n)[2:])
```

Порвать массив

Условие.

Мальчик Петя получил на день рождения массив из чисел, хотя хотел получить машинку. От обиды он решил порвать этот массив, но хочет сделать это, следуя определённым правилам.

За один шаг Петя:

1. Выбирает любой элемент массива, кроме первого и последнего.
2. Добавляет к ответу произведение выбранного числа на сумму всех остальных чисел массива.
3. Рвёт массив на две части - до и после выбранного элемента.
4. Повторяет шаг с каждой получившейся частью массива.

Помогите Пете найти максимальный возможный ответ.

Формат ввода.

В единственной строке дано число N $(1 \leq N \leq 400)$.
Во второй строке дано N чисел a_i $(1 \leq a_i \leq 10^6)$.

Формат вывода.

Выведите одно число - максимальный возможный ответ.

Пример 1

Ввод	Вывод
7 1 2 3 9 10 5 1	293

Пример 2

Ввод	Вывод
2 8 1	0

Примечание

В первом примере выгодно взять числа в следующем порядке:

1. 5, это добавит к ответу 130.
2. 2, это добавит к ответу 46.
3. 9, это добавит к ответу 117.

Во втором примере массив имеет длину 2, поэтому ни одной операции не произойдёт.

Решения

Язык C++

```
#include <algorithm>
#include <iostream>
#include <map>
#include <numeric>
#include <vector>

using namespace std;
using ll = long long;

int main() {
```

```
  int n;
  cin >> n;
  vector<ll> a(n);
  for (auto &el : a)
    cin >> el;
  vector<vector<ll>> dp(n + 1, vector<ll>(n + 1)); // [l, r)
  for (int len = 0; len <= n; ++len) {
    for (int l = 0; l + len <= n; ++l) {
      if (len < 3) {
        dp[l][l + len] = 0;
        continue;
      }
      ll sm = accumulate(begin(a) + l, begin(a) + l + len, 0ll);
      ll best = 0;
      for (int mid = l + 1; mid < l + len - 1; ++mid) {
        best = max(best,
                   dp[l][mid] + dp[mid + 1][l + len] + a[mid] * (sm -
a[mid]));
      }
      dp[l][l + len] = best;
    }
  }
  cout << dp[0][n] << endl;
}
```

Язык Python

```
n = int(input())
a = list(map(int, input().split()))
dp = [[0] * (n + 1) for i in range(n + 1)]
for len in range(0, n + 1):
    for l in range(0, n + 1 - len):
        if len < 3:
            dp[l][l + len] = 0
            continue
        sm = sum(a[l:l+len])
        best = 0
        for mid in range(l + 1, l + len - 1):
            best = max(best, dp[l][mid] + dp[mid + 1][l + len] + a[mid] *
(sm - a[mid]))
        dp[l][l + len] = best
print(dp[0][n])
```

Умная записная книжка

Условие.

После очередного обновления телефона у Поликарпа появилась новая функция в записной книжке. Теперь при наборе номера телефона система показывает все контакты, номера которых начинаются с набранных цифр.

Поликарп хочет узнать, какое минимальное количество дополнительных цифр ему нужно набрать после введенного префикса, чтобы система точно определила, какой контакт он ищет (то есть нашла ровно один номер).

Формат ввода.

В первой строке находятся два целых числа N и Q $(1 \leq N, Q \leq 10^5)$ — количество номеров в записной книжке и количество запросов соответственно.

Следующие N строк содержат телефонные номера. Каждый номер — это непустая строка, состоящая из цифр (0123456789). Длина каждого номера не превышает 15 символов. Все номера различны.

Следующие Q строк содержат запросы. Каждый запрос — это непустая строка, состоящая из цифр (префикс номера). Длина каждого запроса не превышает 15 символов.

Формат вывода.

Для каждого запроса выведите в отдельной строке одно целое число — минимальное количество цифр, которое нужно дописать к префиксу, чтобы однозначно определить номер из записной книжки. Если такого количества не существует, выведите -1.

Пример.

Ввод	Вывод
3 3 012345 0276 987 0 03 98	1 -1 0

Примечание

Первый запрос: к префиксу 0 достаточно дописать одну цифру (например, 1).
Второй запрос: номера с префиксом 03 не существует.
Третий запрос: с префиксом 98 существует только один номер, ничего дописывать не нужно.

Решения

Язык C++

```cpp
#include <iostream>
#include <map>
#include <vector>

using namespace std;

struct Trie {
  vector<map<char, int>> next;
  vector<int> dp, ends;
  int sz;

  int node() {
    next.emplace_back();
    dp.emplace_back(1000);
    ends.emplace_back();
    return sz++;
  }

  Trie() : sz(0) { node(); }

  int add(int v, char c) {
    if (!next[v].count(c))
      next[v][c] = node();
    return next[v][c];
  }

  void dfs(int v) {
    for (auto [c, u] : next[v]) {
      dfs(u);
      ends[v] += ends[u];
    }
  }

  void precalc(int v) {
```

```
    for (auto [c, u] : next[v]) {
      precalc(u);
      dp[v] = min(dp[v], dp[u] + 1);
    }

    if (ends[v] == 1) {
      dp[v] = 0;
    }
  }

  int go(int v, int c) {
    if (!next[v].count(c))
      return -1;
    return next[v][c];
  }
};

int main() {
  int n, q;
  cin >> n >> q;
  Trie trie;
  for (int i = 0; i < n; ++i) {
    string s;
    cin >> s;
    int v = 0;
    for (auto c : s) {
      v = trie.add(v, c);
    }
    trie.ends[v] = 1;
  }
  trie.dfs(0);
  trie.precalc(0);
  while (q--) {
    string s;
    cin >> s;
    int v = 0;
    for (auto c : s) {
      v = trie.go(v, c);
      if (v == -1) {
        cout << -1 << endl;
        break;
      }
    }
    if (v != -1) {
      cout << trie.dp[v] << endl;
    }
```

```
  }
}
```

Язык Python

```python
class Trie:
    def __init__(self):
        # Initialize with empty lists and first node
        self.next = [{}]  # List of dictionaries instead of vector<map>
        self.dp = [1000]  # Initialize dp with 1000 for first node
        self.ends = [0]   # Initialize ends with 0 for first node
        self.sz = 1       # Size counter

    def node(self):
        # Create new node
        self.next.append({})
        self.dp.append(1000)
        self.ends.append(0)
        self.sz += 1
        return self.sz - 1

    def add(self, v, c):
        # Add character c to node v
        if c not in self.next[v]:
            self.next[v][c] = self.node()
        return self.next[v][c]

    def dfs(self, v):
        # Depth-first search
        for c, u in self.next[v].items():
            self.dfs(u)
            self.ends[v] += self.ends[u]

    def precalc(self, v):
        # Precalculate minimum values
        for c, u in self.next[v].items():
            self.precalc(u)
            self.dp[v] = min(self.dp[v], self.dp[u] + 1)

        if self.ends[v] == 1:
            self.dp[v] = 0

    def go(self, v, c):
        # Navigate trie
```

```
        return self.next[v].get(c, -1)

def main():
    # Get input
    n, q = map(int, input().split())

    # Create trie
    trie = Trie()

    # Add strings to trie
    for _ in range(n):
        s = input()
        v = 0
        for c in s:
            v = trie.add(v, c)
        trie.ends[v] = 1

    # Process trie
    trie.dfs(0)
    trie.precalc(0)

    # Process queries
    for _ in range(q):
        s = input()
        v = 0
        found = True

        for c in s:
            v = trie.go(v, c)
            if v == -1:
                print(-1)
                found = False
                break

        if found:
            print(trie.dp[v])

if __name__ == "__main__":
    main()
```