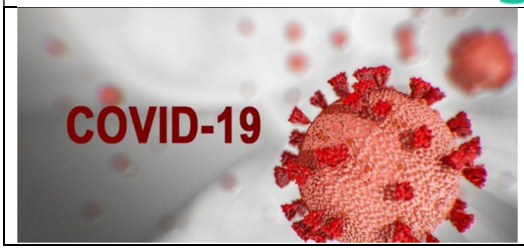
| | |
|---|---|
| Petlja | Школска 2019/2020. |
|   | **https://arena.petlja.org/** |

**Задаци за пети и шести разред**

## Брисање

| Vremensko ograničenje | Memorijsko ograničenje | ulaz | izlaz |
|---|---|---|---|
| 0,04 s | 8 MB | standardni ulaz | standardni izlaz |

Глиша је љубитељ непарних бројева. Једном је написао на табли све бројеве од A до B (укључујући и A и B), а потом избрисао све бројеве чији је збир цифара паран. Одредите колико бројева је остало на табли.

Улаз

Са стандардног улаза се уносе редом природни бројеви A и B у посебним редовима ($1 \leq A \leq B \leq 10^9$).

Излаз

Програм треба да испише један број - број бројева са непарним збројем цифара написаних на табли.

Пример 1

Улаз

10
12

Излаз

2

Пример 2

Улаз

11
14

Излаз

2

Пратећи текст задатка, можемо имплеменирати решење у коме се за све бројеве од A до B израчунава сума цифара и тестира парност. Ово решење се може учинити ефикаснијим ( и освојити већи број поена).

Уочимо да у поднизу на сваких десет узастопних бројева који почињу пуном десетицом постоји тачно 5 бројева који имају непаран збир цифара. (јер се бројеви разликују само у последњој цифри)

Ако би доња граница интервала (број А) био пуна десетица, а горња граница интервала (број Б) био број који је претходник пуне десетице, онда решење можемо добити у временско сложености O(1) тј. у броју корака који не зависи од величине распона интервала тј. разлике B-A +1. Дакле, довољно је пребројати колико има десетица и тај број помножити са 5 односно распон интервала B-A +1 поделити са 2.

```
a = int(input())
b = int(input())
odg =(b-a+1)//2
```

Потребно је додатно и урачунати корекцију доње и горње границе интервала у односу на учитане вредности бројева A и B и проверити парност збира цифара на границама интервала.

**print(odg)**

---

## Аеромитинг

| Vremensko ograničenje | Memorijsko ograničenje | ulaz | izlaz |
|---|---|---|---|
| 0,05 s | 8 MB | standardni ulaz | standardni izlaz |

Потпун квадрат је природан број који је једнак производу неког природног броја са самим собом. На пример, број 16 је потпун квадрат, а број 18 није. (јер 16=4×4).

На аеромитингу учествује m падобранаца. Најспектакуларнија формација падобранаца се добија онда када је број учесника формације потпун квадрат. Али пошто број m можда није потпун квадрат, дозвољено је да се падобранци поделе у неколико тимова, тако да сваки тим чини потпун квадрат падобранаца. Због лепоте приказа, сви тимови морају да буду исте величине, а команда аеромитинга жели да број падобранаца у сваком тиму буде што је могуће већи. Одредите максималну могућу величину једног тима падобранаца.

Улаз

Са стандардног улаза се уноси природан број m (1≤m≤2 000 000 000).

Излаз

Програм треба да испише један број - највећу могућу величину тима падобранаца.

Пример 1

Улаз
98
Излаз
49
Пример 2
Улаз
12
Излаз
4

**Опис решења**

Морамо пронаћи број ***K*** који би био делитељ броја **M** и потпун квадрат: $K=d^2$

Ефикасније решење тражи број ***K*** тако што множи природне бројеве ***d*** (почев од 1) са самим собом све док тај производ не постане већи од ***M*** и проверава дељивост броја ***M*** тим производом.

Ако мали петаци и шестаци познају појам корена, онда могу да имплементирају и решење које тражи број ***K*** тако што множи природне бројеве ***d*** (почев од бројева који нису већи од корена броја ***M***) са самим собом и проверавају дељивост броја ***M*** тим производом.

Иначе, решење које је имплементирано на сајту
https://www.geeksforgeeks.org/largest-factor-of-a-given-number-which-is-a-perfect-square/
може да користи нашим такмичарима као идеја. Решење је детаљно описано, али није коректно имплементирано и не саветујемо директну употребу тог решења без модификације.

```
m = int(input())
ans = 1
d = 2
while d * d <= m:
    if m % (d * d) == 0:
        ans = d * d
    d += 1
print(ans)
```

---

Сиракуза

| Vremensko ograničenje | Memorijsko ograničenje | ulaz | izlaz |
|---|---|---|---|
| 0,04 s | 12 MB | standardni ulaz | standardni izlaz |

У приморском граду Сиракузи на Сицилији, старогрчки математичар Архимед се већ прославио чувеним законом и открићем да краљев златар радије у круну ставља сребро уместо злата.

Зато је Архимеду поверено да растумачи износ кредита који је краљ Сиракузе неколико пута узео од Римљана. Сваки пут је банкар записао износ зајма на листу пергамента користећи римске бројеве. Али, авај, пергамент је скуп и сви бројеви су записани спојено једни за другим.

Када је краљ дошао да врати зајам, испоставило се да није једноставно успоставити поделу дугачког записа на бројеве. На пример, ако је на пергаменту написан зајам „XVIV“, он се може протумачити као збир римских бројева на различите начине, на пример, 16 + 5 = 21 или 15 + 4 = 19 (могуће су и друге варијанте истих вредности 10+5+4 или 10+5+1+5) .

Наравно, краљ жели да врати што је мање могуће новца, те жели поделити низ бројева на римске бројеве, тако да је збир свих бројева што је могуће мањи. Зато је позвао у помоћ Архимеда који зна да реши овај проблем.

Размишљајте и Ви као Архимед и напишите програм за израчунавање износа зајма.

Улаз

Са стандардног улаза се уноси низ чија дужина не прелази 250 знакова. Низ се састоји само од великих латиничних слова I, V, X, L, C, D, M.

Излаз

Програм треба да испише један број - најмању могућу суму која се може добити дељењем овог низа у низ правилно записаних римских бројева. Излаз записати цифрама декадног бројног система (0..9). Не заборавите да највећи број декадног бројног система који се може правилно записати римским цифрама има вредност 3999.

Пример 1

Улаз

XVIV

Излаз

19

Пример 2

Улаз

MMMMMMMMMMMMMMMMMMMMDM

Излаз

21500

**Опис решења**

Овај задатак је сличан задатку конверзије исправног записаног римског броја у вредност броја у декадном систему , али им се решења доста разликују, јер улаз у нашем проблему садржи низ исправно записаних римских бројева.
Због тога није било мудро предати решење са конверзијом римског записа зато што је оно тачно за одређен број тест примера.

Вредност исправно записаног кредита се може
одредити тако што читамо све цифре улазне ниске сдесна на лево и на текући збир
додељујемо вредност мање цифре (M има вредност 1000, D вредност 500, C вредпост 100, L вредност 50, X вредност 10, V вредност 5 и I вредност 1). Али, постоји изузетак од овог правила је случај када се у запису појави мања испред веће цифре и тада се текућој вредности зајма додаје мања цифра (CM има вредност 900, CD вредност 400, XC вредост 90, XL вредност 40, IX вредност 9, IV вредност 4 ).

На пример, израчунајмо вредност записа MMDMM.
На основу претходног описа обраде улазне ниске, вредност овог броја је 1000 + 1000 + 500 + 1000 + 1000 (тј. зајам је износио 1000+1000+500+1000+1000 или 2500+2000 или 2000+500+2000 или...)
Ако би се применио алгоритам конверзије броја записаног римским цифрама на неправилан запис MMDMM, онда би алгоритам конверзије дао број 1000+1000-500+1000+1000=3500 што јесте мање од 4500, али не можемо да кажемо да је зајам редом износио MM +DM + M зато што DM није исправан запис римским цифрама.

```
s = input()
ans = 0
digits = {"I": 1, "V": 5, "X": 10, "IV": 4, "IX": 9, "L": 50, "C": 100, "XL": 40, "XC": 90, "D": 500, "M": 1000, "CD": 400, "CM": 900}
i = len(s) - 1
while i >= 0:
    num2 = s[i - 1:i + 1]
    num1 = s[i]
    if i > 0 and num2 in digits:
        ans += digits[num2]
        i -= 2
    else:
        ans += digits[num1]
        i -= 1
print(ans)
```

==========

**4.**

Трансформације

| Vremensko ograničenje | Memorijsko ograničenje | ulaz | izlaz |
|---|---|---|---|
| 0,05 s | 64 MB | standardni ulaz | standardni izlaz |

Ика воли да се игра речима. Његова омиљена игра је следећа - дате су две речи једнаке дужине које не морају бити идентичне. Друга реч се мора добити од првог након примене одређеног броја операција "дељења".

Нека је дата непразна ниска $S$. Операција „дељење“ за ниску $S$ се састоји од следећег низа акција:

1. Раздвојите ниску $S$ на два непразне ниске $x$ и $y$ било које величине, тако да важи $S=xy$.
2. Заменимо места нискама $x$ и $y$.

На пример, ако је дата реч superspeed, онда описаном трансформацијом дељења добијамо реч speedsuper.

Ика може да изврши описану операцију тачно $K$ пута да би добио дату ниску $T$.

Напишите програм који ће израчунати на колико начина то Ика може да уради. Решење исписати по модулу $10^{31}$.

Улаз

У прва два реда стандардног улаза су дате две ниске знакова $S$ и $T$ ($2 \leq (|S|=|T|) \leq 1000$). У трећем реду је дат природан број $K$ ($K \leq 10^5$).

Излаз

У једином реду стандардног излаза исписати одговор.

Пример 1

Улаз

```
сесесе
сесесе
1
```

Излаз

```
2
```

Пример 2

Улаз

```
cg
gc
2
```

Излаз

```
0
```

Пример 3

Улаз

```
XnOAHhDwPjUBbKADHJXlS
nOAHhDwPjUBbKADHJXlSX
20888
```

Излаз

```
904761904761904761904761904761904761
```

**Опис решења**

Главна идеја задатка је рад са великим бројевима.
У задатку је корисно применити метод табеларног израчунавања броја начина за формирање ниске ***T***.

Уочимо да је свака операција дељења уствари цикличко померање полазног низа. Можемо прећи све цикличке промене у низу алгоритмом линеарне временске сложености и пребројати колико тих измена формира ниску ***T*** (и означити ту вредност са ***a***), као и колико тих измена не формира ниску ***T*** (и означиту ту вредност са ***b***).

Уведимо два низа: dpA[k+1] и dpB[k+1], која ће нам помоћи да израчунамо на колико начина можем да образујемо низ ***T*** и неки други низ ако употребимо ***i*** операција дељења ниске($0 \leq i \leq k$).
Број начина у оба случаја можемо представити рекурентним формулама:
dpS[i] = dpS[n - 1] * (a - 1) + dpT[n - 1] * a
dpT[i] = dpS[n - 1] * b + dpT[n - 1] * (b - 1)

За такмичаре који не користе Пајтон, важно је да знају да имплементирају аритметичке операције сабирања и множења у раду са великим бројевима (на сличан начин како се учи извођење ових операцијау млађим разредима основне школе).
Решење се заснива на бележењу остатака при дељењу са бројем $10^{31}$.
Сам модул је изабран тако да се поједностави имплементација младим такмичарима и избегне имплементација потпрограма за рад са остатком при дељењу са дугачким бројевима, и уместо тога једноставно се „пресеку“ вредности помоћних бројева на 31. позицији. То јест, ако број постане 32-цифрен, уклања се и последњи елемент низа вишецифрених бројева.

**Задаци за седми и осми разред**

## 1. Графика

| Vremensko ograničenje | Memorijsko ograničenje | ulaz | izlaz |
|---|---|---|---|
| 0,5 s | 256 MB | standardni ulaz | standardni izlaz |

Лана жели да на екрану нацрта квадрат величине n×nn×n пиксела у различитим бојама. Ланин монитор подржава 26 боја. За означавање боја можемо користити мала слова енглеске абецеде (од „a“ до z“).

Лана жели да сваки пиксел обоји одређеном бојом, што зависи од удаљености од пиксела од најближе дијагонале.

Наиме, Лана жели да обоји ћелије на дијагоналама квадрата бојом "a". Суседне ћелије ће обојити бојом "b", а суседне ћелије које још нису обојене обојиће бојом "c" и тако даље. Након боје "z", Лана се враћа у боју "a".

Напишите програм који за дато nn, исписује слику коју ће добити Лана.

Улаз

Са стандардног улаза се уноси цео број nn (1≤n≤1001≤n≤100).

Излаз

Испишите на стандардни излаз nn линија са по nn знакова - слику коју ће добити Лана.

Пример 1

Улаз

```
3
```

Излаз

```
aba
bab
aba
```

Пример 2

Улаз

```
11
```

Излаз

```
abcdefedcba
babcdedcbab
cbabcdcbabc
dcbabcbabcd
edcbababcde
fedcbabcdef
edcbababcde
dcbabcbabcd
cbabcdcbabc
babcdedcbab
abcdefedcba
```

**Решење**

**Јава**

```
import java.util.Arrays;
import java.util.Scanner;


public class diag {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                int d = Math.min(Math.abs(i - j), Math.abs(i + j - n + 1));
                System.out.print((char)('a' + (d % 26)));
            }
            System.out.println();
        }

    }
}
```

**Пајтон**

```
n = int(input())

a = [['?' for i in range(n)] for j in range(n)]

for i in range(n):
    for j in range(n):
        v = min(abs(i - j), abs(n - 1 - i - j))
        a[i][j] = chr(ord('a') + v % 26)

print("\n".join(["".join(x) for x in a]))
```

Тренирање

| Vremensko ograničenje | Memorijsko ograničenje | ulaz | izlaz |
|---|---|---|---|
| 0,7 s | 512 MB | standardni ulaz | standardni izlaz |

Светска спортска организација осмислила је нову стратегију вежбања за спортисте. У складу са овом методом, спортиста мора тренирати сваки дан, али смањење и повећање оптерећења морају наизменично да се смењују из дана у дан.

План тренинга је скуп позитивних целих бројева $a_1,a_2,..,a_m$, где $a_i$ описује оптерећење спортисте i-ог дана. Свака два суседна дана морају имати различито оптерећење тј. $a_i$ и $a_{i+1}$ имају различите вредности.

Да би се контролисао раст и смањење оптерећења спортисте, мора бити испуњен следећи услов: ако је $a_i < a_{i+1}$, онда је $a_{i+1} > a_{i+2}$, односно ако је $a_i > a_{i+1}$, онда је $a_{i+1}<a_{i+2}$ (за i које узима вредности од 1 до m−2).

Укупно оптерећење током извођења плана треба да буде n, то јест $a_1+a_2+...+a_m=n$.

У плану тренинга не постоји ограничење за број дана (вредност m може бити било која), али оптерећење првог дана тренинга мора бити фиксирано тј. важи да $a_1 = k$.

Пре тестирања нове методе, академија жели да открије колико различитих планова обуке задовољавају описана ограничења. Напишите програм који, за дате вредности n и k, одређује колико различитих планова обуке испуњавају описана ограничења и приказује остатак дељења броја таквих планова са $10^9+7$.

Улаз

Са стандардног улаза се уносе у првом реду цели бројеви n и k ($1\le n\le 5000$, $1\le k\le n$).

Излаз

На стандарни излаз исписати један број: остатак дељења броја планова обуке са $10^9+7$.

Пример 1

Улаз

5 2

Излаз

2

Објашњење: У првом примеру могући су следећи планови: [2,1,2], [2,3].

Пример 2

Улаз

7 3

Излаз

3

Објашњење: У другом примеру могући су следећи планови: [3,1,2,1], [3,1,3], [3,4].

Опис решења:

Да бисмо решили проблем за дато временско ограничење, онда за $n \le 10$, можемо применити рекурзивну претрагу, док за $n \le 30$ мора се користи претрага са одсецањем.

Да би смо решили подпроблеми за $n \le 500$ и $n \le 5000$, неопходно је применити динамичко програмирање.

Прво размотримо решење сложености $O(n^3)$ које решава у датом временском ограничењу проблем за за $n \le 500$.

Означимо са up[n][k] број планова тренинга који задовољавају описана ограничења, чија укупна оптерећења су n, оптерећења првог дана су k, а оптерећења другог дана су строго већа од оптерећења првог дана.
Слично ћемо увести и низ down[n][k] за број планова тренирања код којих је оптерећење другог дана строго мање од оптерећења првог дана.
Важи: up[n][n] = down[n][n] = 1 за k=n.
За k<n, вредности се могу израчунати помоћу формула:

up[n][k]= $ \sum_{i=k+1}^{n-k} down[n-k][i] $

down[n][k]= $ \sum_{i=1}^{k-1} up[n-k][i] $

Вредност коју тражимо једнака је up[n][k] + down[n][k]

Да би се убрзало решење, могу се користити два приступа.
Приступ 1: Уочимо да up[n][k]=up[n][k+1] + down[n-k][k+1]
Слично, важи да down[n][k]=down[n][k-1] + up[n][k-1]

Појединачне вредности рачунамо у времену у O (1), а време рачунања свих вредности у низовима је O ($n^2$).

Приступ 2: префиксне суме
Уведимо помоћне низове:
sumUp[n][k]=\sum_{i=1}^{k} up[n][i]
sumDown[n][k]=\sum_{i=1}^{k} down[n][i]

Тада
up[n][k]=sumDown[n-k][n-k] + sumDown[n-k][k]
down[n][k]=sumUp[n][k-1]

У свим случајевима треба бити пажљив и обезбедити да индекси низова којима се приступа у формулама
не прелазе доњу и горњу границу за број чланова низа.

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 5001;
const int MOD = 1e9+7;

int up[MAXN][MAXN];
int dn[MAXN][MAXN];
int sumu[MAXN][MAXN];
int sumd[MAXN][MAXN];

int main() {
```

```
    int n, p;
    cin >> n >> p;

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j < i; j++) {
            up[i][j] = 0;
            if (i - j > j) {
                up[i][j] = ((sumd[i - j][i - j] - sumd[i - j][j]) % MOD + MOD) % MOD;
            }
            dn[i][j] = sumu[i - j][j - 1];

            sumu[i][j] = (sumu[i][j - 1] + up[i][j]) % MOD;
            sumd[i][j] = (sumd[i][j - 1] + dn[i][j]) % MOD;
        }
        up[i][i] = dn[i][i] = 1;
        sumu[i][i] = (sumu[i][i - 1] + up[i][i]) % MOD;
        sumd[i][i] = (sumd[i][i - 1] + dn[i][i]) % MOD;
        for (int j = i + 1; j <= n; j++) {
            sumu[i][j] = sumu[i][i];
            sumd[i][j] = sumd[i][i];
        }
    }


    int ans = up[n][p];
    if (p != n) {
        ans = (ans + dn[n][p]) % MOD;
    }
    cout << ans << endl;

    return 0;
}
```

## Свемир

| Vremensko ograničenje | Memorijsko ograničenje | ulaz | izlaz |
|---|---|---|---|
| 0,4 s | 64 MB | standardni ulaz | standardni izlaz |

Планирано је да се пошаље у свемир екипа у самовозећој ракети компаније Тесла. Пријављено је nn кандидата обележених бројевима од 1 до n, међу којима је потребно одабрати путнике у ракети.

Власник компаније Тесла жели да пошаље што више кандидата у свемир. Извршено је анкетирање кандидата, током ког је свако могао навести једног кандидата са којим није спреман да иде у свемир. Резултат анкете i-тог кандидата је цео број $a_i$, који је једнак броју кандидата са којим i-ти кандидат није спреман да

иде у свемир, или –1 ако је i-ти кандидат спреман да крене у свемир у било ком саставу.

Власник компаније мора одабрати који ће од кандидата ићи у свемир. Одлучено је да се изаберу путници, тако да ако неки кандидат i уђе у ракету и ако $a_i \neq -1$, онда кандидат $a_i$ не улази у ракету. Потребно је написати програм који, с обзиром на резултате анкете кандидата, утврђује максимални број кандидата који се могу послати у свемир.

Улаз

Са стандардног улаза се уноси у првом реду цео број n ($1 \leq n \leq 300000$). Следећих n редова садржи резултате анкете, тако да i-ти од тих редова садржи резултат анкете i-ог кандидата тј. цео број $a_i$ ($a_i = -1$ или $1 \leq a_i \leq n$, $a_i \neq i$).

Излаз

Испишите на стандардни излаз један цео број - максимални број кандидата који се могу послати у свемир.

Пример 1

Улаз

```
4
4
1
2
3
```

Излаз

```
2
```

Пример 2

Улаз

```
4
2
-1
-1
-1
```

Излаз

```
3
```

Опис решења

За $n \leq 20$ задатак се може решавати претрагом по подскуповима кандидата и за сваки подскуп проверити да ли испуњава захтеве за одлазак у свемир и наћи највећи такав.

Временска сложеност: $O(n2^n)$.

Да бисмо решили проблем за $n \leq 300\,000$ у бар квадратној временској сложености, проблем преформулишемо у проблем из теорије графова.

Конструишемо усмерени граф, чији ће чворови бити кандидати за свемир, тако да грана постоји између i, $a_i$ ако је $a_i \neq -1$\$. Услов да кандидати i и $a_i$ не могу

истовремено да буду у ракети одговара чињеници да скуп кандидата који су послати у свемир мора да чини независни или стабилан скуп у графу.

Ако је полазни граф дрво чији корен је чвор 1, а гране су усмерене ка корену.
можемо користити похлепни алгоритам да бисмо пронашли независни скуп у стаблу.
Похлепни алгоритам: уклонимо оријентацију грана и поступите на следећи начин:
уврстимо све листове стабла у скуп,
бришемо их из стабла и све суседне чворове. Јер, ако лист није изабран у независни скуп и ако његов суседни чвор такође није изабран, онда можемо одабрати лист и тако повећати величину независног скупа.
Ако је изабран њему суседни чвор, онда можемо за скуп изабрати лист уместо суседног чвора (јер величина скупа се неће променити).

Ако полазни граф није дрво, онда можемо користити динамичко програмирање.
Из сваког чвора излази највише једна грана.
Свака компонента повезаности таквог графа је стабло или циклус.
Решимо проблем независно за сваку компоненту повезаности и саберимо одговоре.
За компоненте које представљају стабло, решење је већ описано горе.
Ако желимо доследно да применимо технику динамичког програмирања на стабло, онда уведимо два низа:
is[u], out[u].
Означићемо са is[u] величину максималног независног скупа у подстаблу чвора u тако да скуп садржи чвор u и означимо са out[u] величину максималног независног скупа у подстаблу чвора u који не садржи чвор u.
Онда у том подстаблу стаблу важи:

is[u]= $\sum_{x-sinovi\ od\ u} out[x]$

out[u]= $\sum_{x-sinovi\ od\ u} \max(is[x], out[x])$

Ако чвор u је лист, онда важи is[u] = 1, out[u] = 0.
Сад је одговор за дрво једнак max(is[r],out[r]) за корен стабла r.

Размотримо сада компоненту повезаности графа која садржи циклус.
Уклонимо гране циклуса и решимо проблем динамичким програмирањем за свако добијено стабло тако што ћемо пребројати одговоре.

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

using namespace std;

void solve() {
        int n;
```

```
cin >> n;
vector<vector<int> > graph(n);
vector<pair<int, int>> edges;
for (int i = 0; i < n; ++i) {
	int ind;
	cin >> ind;
	if (ind != -1) {
		--ind;
		int a = i;
		int b = ind;
		if (a > b) {
			swap(a, b);
		}
		edges.push_back({a, b});
	}
}

sort(edges.begin(), edges.end());
edges.erase(unique(edges.begin(), edges.end()), edges.end());

for (pair<int, int> e : edges) {
	graph[e.first].push_back(e.second);
	graph[e.second].push_back(e.first);
}

vector<int> st, cycle;
vector<int> used(n);

function<bool(int, int)> find_cycle = [&](int v, int p) {
	st.push_back(v);
	used[v] = 1;
	for (int to : graph[v]) {
		if (!used[to]) {
			if (find_cycle(to, v)) {
				return true;
			}
		} else if (used[to] == 1 && to != p) {
			while (st.back() != to) {
				cycle.push_back(st.back());
				st.pop_back();
			}
			cycle.push_back(st.back());
			return true;
		}
	}
	used[v] = 2;
```

```
            st.pop_back();
            return false;
        };

        function<pair<int, int>(int, int)> dfs = [&](int v, int p) {
            pair<int, int> ret = {0, 1};
            for (int to : graph[v]) {
                if (used[to] != 3 && to != p) {
                    pair<int, int> tmp = dfs(to, v);
                    ret.first += max(tmp.first, tmp.second);
                    ret.second += tmp.first;
                }
            }
            return ret;
        };

        function<void(int)> mark = [&](int v) {
            used[v] = 4;
            for (int to : graph[v]) {
                if (used[to] != 4) {
                    mark(to);
                }
            }
        };

        int ans = 0;

        for (int i = 0; i < n; ++i) {
            if (!used[i]) {
                cycle.clear();
                st.clear();
                find_cycle(i, i);
                if ((int)cycle.size()) {
                    for (int v : cycle) {
                        used[v] = 3;
                    }

                    vector<pair<int, int>> arr;
                    for (int v : cycle) {
                        arr.push_back(dfs(v, v));
                    }

                    pair<pair<int, int>, pair<int, int>> d = {{arr[0].first, arr[0].first},
{arr[0].first, arr[0].second}};

                    for (int j = 1; j < ((int)arr.size()); ++j) {
```

```
                                        pair<pair<int, int>, pair<int, int>> next = {{0, 0}, {0, 0}};
                                        next.first.first = max(d.first.first, d.first.second) +
arr[j].first;
                                        next.first.second = max(max(arr[j].first, arr[j].second)
+ d.first.first, next.first.first);
                                        next.second.first = max(d.second.first,
d.second.second) + arr[j].first;
                                        next.second.second = max(max(arr[j].first,
arr[j].second) + d.second.first, next.second.first);
                                        swap(next, d);
                                }

                                ans += max(max(d.first.second, d.first.first), d.second.first);
                        } else {
                                auto tmp = dfs(i, i);
                                ans += max(tmp.first, tmp.second);
                        }

                        mark(i);
                }
        }

        cout << ans << "\n";
}


int main() {
        ios::sync_with_stdio(false);
        cin.tie(nullptr);
        solve();
}
```

---

Мали речник

| Vremensko ograničenje | Memorijsko ograničenje | ulaz | izlaz |
|---|---|---|---|
| 0,3 s | 24 MB | standardni ulaz | standardni izlaz |

Милица има списак који се састоји од N речи. Моли Вас да јој помогнете да одреди подскуп састављен од D речи са списка тако да:

1. Не постоје две речи у подскупу такве да је једна реч префикс друге.
2. Дужина најдуже речи је што је могуће мања.

Укупна дужина свих речи у тексту није већа од 999 999. Речи садрже само мала слова енглеске абецеде.

Улаз

У првој линији стандардног улаза дата су два цела броја N и D $1 \leq D \leq N \leq 10^6$). У наредних N линија дат је списак речи (по једна реч у свакој линији).
Излаз

Одштампајте на стандардни излаз дужину најдуже речи са списка. Иначе, штампајте -1.
Пример 1
Улаз
4 2
knezevina
knez
matf
matfizacija
Излаз
4
Пример 2
Улаз
4 3
abcdef
abc
a
kraj
Излаз
-1
Пример 3
Улаз
8 4
knez
imagine
dragons
georges
enescu
androvera
zlatnogrivasti
coldplay
Излаз
7

Опис решења

Задатак се може решавати на више начина.

Најважније је осмислити добар модел за сравњивање ниски у овом задатку како би подскуп заиста чиниле све речи тако да ниједна реч није префикс друге речи. Можемо користити технику хеширања ниски или изградњу стабла.

Можемо и сортирати све речи према дужини и бинарном претрагом по решењу за дато ***D*** убацивати у подскуп речи тако да их узимамо редом из сортиране колекције и да за сваку нову реч важи да у постојећом скупу
све речи су такве да ниједна реч није префикс друге речи.

```
#include <cassert>
#include <algorithm>
#include <iostream>
#include <map>
#include <string>
#include <vector>
using namespace std;


const int BASE = 31;
const int MOD = (int)1e9+7;

inline bool cmp(const string &a, const string &b) {
    return a.size() < b.size();
}

int main() {
    int n, k;
    vector<string> v;
    int totalLen = 0;

    cin >> n >> k;
    for (int i = 0; i < n; ++i) {
        string s;
        cin >> s;
        v.push_back(s);
        totalLen += s.size();
    }
    sort(v.begin(), v.end(), cmp);
    assert(totalLen <= 1000000);

    map<long long, int> hash;
    for (int i = 0; i < n; ++i) {
        long long hashVal = 0;

        for (char ch : v[i]) {
            int d = ch - 'a' + 1;
            hashVal = hashVal * BASE + d;
            if (hashVal >= MOD) {
                hashVal %= MOD;
            }
```

```
            if (hash.find(hashVal) != hash.end()) {
                hash.erase(hash.find(hashVal));
            }
        }
        hash[hashVal] = i;
        if (hash.size() == k) {
            cout << v[i].size() << "\n";
            return 0;
        }
    }
    cout << "-1\n";
    return 0;
}
```