Поднизови

| Временско ограничење | Меморијско ограничење | улаз | излаз |
|---|---|---|---|
| 0,4 s | 32 MB | standardni ulaz | standardni izlaz |

Дат је низ од *N* природних бројева.

Допринос подниза у низу се дефинише као производ *max*×*min*×*D* где је *max* највећи број у поднизу, *min* најмањи број у поднизу , *D* је дужина подниза. Написати програм који ће израчунати суму свих могућих доприноса унутар датог низа.

УЛАЗ

У првом реду стандардног улаза налази се природни број *N* (1≤*N*≤500000). У следећих *N* редова налазе се чланови низа, сваки у свом реду. Чланови низа биће природни бројеви из интервала [1,$10^8$]

ИЗЛАЗ

Испишите један број у једном реду, тј. последњих 9 цифара тражене суме доприноса из текста задатка. Водеће нуле тог 9-цифреног броја није потребно исписивати.

ПРИМЕР 1

УЛАЗ

```
2
1
3
```

ИЗЛАЗ

```
16
```

Појашњење: Низ се састоји од два броја 1 и 3. Могући поднизови су (1), (3) и (1,3). Њихови доприноси су редом 1, 9 и 6, и сума доприноса је 16.

ПРИМЕР 2

УЛАЗ

```
4
2
4
1
4
```

ИЗЛАЗ

```
109
```

Појашњење: Могући поднизови су (2), (4), (1), (4), (2, 4), (4, 1), (1, 4), (2,4,1), (4,1,4) i (2,4,1,4). Њихови доприноси су редом 4, 16, 1, 16, 16, 8, 8, 12, 12 и 16,те укупна сума доприноса је једнака 109.

**Решење**

Задатак од нас тражи да за свака два броја ***A*** и ***B*** (1 <= A <= B <= N ) израчунамо допринос подниза [A, B] улазног низа X и да добијене вредности саберемо.

Техником покретне десне границе слева на десно можемо да одржавамо низ решења ***T*** за тренутну вредност границе ***B*** , тако да на позицији ***A*** тог низа ***T*** се чува допринос подниза [***A***, ***B***] .

Замислимо да сваки члан ***T[A]*** низа ***T*** садржи још и редом вредности ***m***, ***M*** и ***L*** односно: минималан број подниза [A,B], максималан број подниза [A,B], дужину подниза [A,B].
Тада би вредност члана T[A] која нам је потребна представљала садржалац описаних вредности m, M и L .
У низу ***T*** је важно ефикасно решити памћење нових вредности доприноса.
Будући да је вредност члана низа T производ његових интерних вредности поља (m , M и L), погледајмо како се оне мењају повећавањем ***B*** за 1 (померањем границе ***B*** удесно).
Вредности **L** сваког члана с индексом мањим или једнаким **B** се повећају за 1. За праћење промена вредности **m**, **M** потребна нам је ефикасна структура података која памти најближи леви претходник, најближи десни следбеник.
Природа операција (описаних у формулацији задатака) је таква да је потребно користити стек и/или сегментно стабло с пропагацијом
Ефикасно решење ће свих O(N) операција на сегментном стаблу обавити у сложености O(log N), те је укупна временска сложеност решења
O(N log N).

**Жирафице**

| Vremensko ograničenje | Memorijsko ograničenje | ulaz | izlaz |
|---|---|---|---|
| 0,4 s | 45 MB | standardni ulaz | standardni izlaz |

Освануо је свечани дан у београдском Зоо врту. Клуб љубитеља жирафа је набавио $N^2$ жирафа и оградио их N×N матрицом којој се у сваком пољу налази један кавез са тачно једном жирафом. Докле год су живе, жирафе расту без престанка. На пример, ако жирафа нарасте 7 центиметара у години дана, онда ће за 6 месеци нарасти 3.5 центиметра. Али, шеф Зоо врта се пита које ће величине бити највећа повезана група жирафа које су све исте висине ако жирафе наставе расти брзином којом тренутно расту. Шеф Зоо врта је измерио тренутну висину сваке жирафе у кавезу и тражио да напишете програм који одговора на дато питање. Две жирафе су суседне ако њихови кавези у матрици имају заједничку страницу. Две жирафе су повезане ако постоји низ суседних жирафа који води од прве до друге жирафе. Група жирафа је повезана ако су сваке две жирафе у групи повезане.

УЛАЗ

У првом реду стандардног улаза налази се природан број N ($1 \leq N \leq 708$). Након тога следи N редова са по N природних бројева. У i-том од тих редова налазе се бројеви hij ($1 \leq hij \leq 10^6$). Broj hij представља почетну висину жирафе у i-том реду и j-тој колони, изражену у центиметрима. Након тога следи још N редова са по N prirodnih brojeva. У i-том од тих редова налазе се бројеви vij ($1 \leq vij \leq 10^6$). Број vij представља раст жирафе у i-том реду и j-тој колони током једне године, изражен у центиметрима.

ИЗЛАЗ

У први и једини ред стандардног излаза испишите тражени број из текста задатка.
ПРИМЕР 1
УЛАЗ

3

10 20 30

30 20 20

50 20 10

30 20 10

10 20 10

10 20 30

ИЗЛАЗ
7
ПРИМЕР 2
УЛАЗ

2

60 20

60 60

20 50

20 50

IZLAZ

3

Појашњење: У једном временском тренутку, највећа повезана група имаће 3 жирафе на позицијама (0,0), (0,1) и (1,0)

**Решење**
Посматрајмо два суседна кавеза. Жирафе у тим кавезим могу највише у једном тренутку бити једнаке висине. Тренутак у ком жирафе у нека два суседна кавеза постану једнаке висине назовимо баланс.

Број баланса је мањи од 4n. За сваки баланс можемо искористити ДФС или БФС за ширење из кавеза у ком су жирафе постале исте висине и бројање жирафа у компоненти повезаности при томе пазећи да за одређени тренутак не прођемо више од једном по неком кавезу.
Мана овог решење је временска сложеност $O(n^4)$ јер ћемо кроз сваки кавез проћи највише 4 пута (једном за сваки баланс у који је укључен тај кавез).

Да би смањили временску сложеност, нећемо користити БФС или ДФС него ћемо користити ДСУ за обраду свих баланса.
Дакле, памтимо родитеља сваког чвора, а корен сваког повезаног скупа памти величину њему припадајућег скупа.
Прво сажмемо у један чвор повезане жирафе која заједно напредују у расту, затим израчунамо балансе за свака два суседна чвора.
Балансе обрађујемо тако да спајамо чворове у унији (структури) који у том тренутку постану једнаке висине. При обради баланса памтимо све промене које смо направили у унији (структури) како можемо реконструисати структуру како је изгледала пре обраде баланса. Након обраде баланса вратимо структуру у почетни изглед.
Временска сложеност $O(n^2 \log n)$.

3. Нинџа корњача Донаетло планира да се ушуња у тајни бункер и украде тајне нацрте. Знамо да се бункер састоји од N соба и M ходника који их повезују. Сваки ходник је проходан у оба смера. Ниједан ходник не повезује собу са самом собом, никоје две собе нису повезане са више од једним ходником.

У неким собама постоје излази из бункера. Чим Донателло узме нацрте, у соби у којој се налази, укључиће се аларм. Након тога, сви излази из

бункера ће бити затворени и Донателло ће бити заробљен.

Али, хакери из Вашег ИТ тима су успели да промени безбедносне поставке бункера, тако да када се аларм активира у соби $i$, неће се сви излази из бункера затворити, већ само K излаза најближих овој соби $i$.

Удаљеност између две собе се дефинише као минимални број ходника које требате прећи да бисте прешли из једне собе у другу.
Ако су неки излази смештени на истој удаљености од собе $i$, онда ће их систем затварати
поштујући растући редослед нумерација њихових соба.

Када се аларм активира, Донателло мора што је пре могуће да напусти бункер. Пошто сви детаљи изгледа бункера и излаза нису унапред познати, неопходно је за сваку од соба израчунати где ће се налазити најближи откључан излаз када се активира аларм у одређеној соби. Од свих откључаних излаза који се налазе на истој удаљености од просторије у којој се активирао алам потребно је одредити излаз који се налази у соби са најмањим бројем.

Улаз

У првом реду стандардног улаза дата су два цела броја N и M раздвојена размаком, која означавају број соба и ходника у бункеру (1\leq N \leq 100 000, 0 \leq M \leq 100 000$).
У наредних $M$ редова дата су по два броја $a_i$ и $b_i$ — бројеви соба, повезаних $i$-тим ходником ($1 \leq a_i , b_i \leq N$ ).
Потом следи цео број $A$ — који представља број соба са излазима из бункера ($1 \leq A \leq N$ ).
У следећем реду налази се A различитих бројева $e_i$ (раздвојених размаком) , који означавају бројеве соба, у којима се налазе излази ($1 \leq e_i \leq N$).

Последњи ред улаза садржи цео број $K$ ($0 \leq K \leq 10$) —број излаза, који се затварају када се активира аларм.

Излаз

Испишите $N$ бројева,тако да $i$-ти број представља нумерацију најближе собе са отвореним излазом након активирања аларма у соби с нумерацијом $i$.
Ако за неки $i$ постоји неколико погодних соба, онда изаберите собу с најмањом нумерацијом. Ако није успела Донателова евакуација из собе $i$, испишите −1.

Пример 1

Улаз

5 5

1 2

2 3

3 4

4 5

5 1

2

1 3

0

Излаз

1 1 3 3 1

Појашњење: У првом примеру,$K$ = 0, означава да само требате пронаћи најближи излаз за сваку собу. Излаз се недвосмислено одређује за све собе осим за 2.
Излазу у собама 1 и 3 налазе се на удаљености од једног пролаза дуж коридора, и међу њима је потребно одабрати собу
са мањим бројем, тј. 1.

Пример 2

Улаз

9 10

1 2

2 3

3 4

4 5

6 7

6 8

6 9

7 8

7 9

8 9

7

1 2 3 4 5 6 7

2

Излаз

3 3 4 5 3 -1 -1 -1 -1

У другом примеру, немогуће је изаћи из соба са бројевима од 6 до 9, јер су из њих
достижна само два излаза. Оба излаза се затварају у случајевима активирања аларма.

**Решење**
Након учитавања улаза, креирамо неусмерени граф чији чворови су собе, а ходници су гране графа. Применом претраге графа у ширину, налазимо достижне излазе.

4. Прва дама такмичарског програмирања ММ увежбава алгоритме текста. ММ креира речник у ком поставља речи ( у лексикографски растућем поретку). Напишите програм који за дату реч утврђује колико речи у речнику је мање (у смислу лексикографског поређења).

У првом реду стандардног улаза дат је природан број N ($1 \leq N \leq 100000$) који представља број команди који Ваш програм обавља над речником. У наредних $N$ редова, налази се нека од команди
INSERT реч
LESS реч
Речи су образоване од малих и великих слова енглеске абецеде, али се сматра да су речи "MagDalina" и "magDaLINa" исте. Можете претпоставити да ниједна реч речника није дужа од 100 карактера, као и да креирани речник неће имати више од 4000000 карактера.

Ваш програм мора (за сваку команду која почиње са LESS) да испише у посебном реду ненегативан цео број који представља број речи лексикографских мањих од речи која следи иза команде LESS. Ако дата реч не постоји у речнику, исписати "NA".

Улаз

12

INSERT potop

INSERT PotoP

INSERT Ognjen

INSERT Rajkov

LESS rajkov

INSERT Daniil

LESS uros

INSERT Milica

INSERT Masa

LESS OgnjeN

INSERT Irina

INSERT MilicA

Излаз

2

NA

3

Решење

У задатку се могло користити префиксно стабло, али је временско ограничење датао тако да и следеће решење лошије ефикасности (али једноставно за имплементацију) освоји све поене

```
#include <vector>
#include <iostream>
#include <algorithm>
#include <set>



using namespace std;

int main()
{

    int n;
    cin >> n;
    set <string> s;
    string k, r;

    for(int i=0; i<n; i++){

        cin >> k >> r;
        transform(r.begin(), r.end(), r.begin(), ::tolower);

        if(k == "INSERT")
            s.insert(r);
        else{
            auto pos = s.find(r);
```

```
            if(pos != s.end())
                cout << distance(s.begin(), pos) << endl;
            else
                cout << "NA" << endl;
        }
    }

    return 0;
}
```

Ефикасније решење:

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

struct Node {

    Node() {
        end = false;
    }

    bool end;
    std::map<char, std::pair<Node*, int>> chars;
};

bool addWord(Node* root, std::string& word, int i) {
    if(i==word.size()) {
        bool res = root->end;
        root->end = true;
        return !res;
    }

    auto it = root->chars.find(word[i]);
    if(it == root->chars.end()) {
```

```
        root->chars[word[i]] = std::make_pair(new Node(), 0);
    }

    bool added = addWord(root->chars[word[i]].first, word, i+1);
    if(added)
        root->chars[word[i]].second++;
    return added;
}

int less(Node* root, std::string& word, int i) {

    if(i == word.size())
        return root->end ? 0 : -1;

    auto it = root->chars.find(word[i]);
    if(it == root->chars.end()) {
        return -1;
    }

    int total = less(it->second.first, word, i+1);
    if(total == -1)
        return -1;
    for(auto it2 = root->chars.begin(); it2 != it; it2++) {
        total += it2->second.second;
    }
    if(root->end)
        total++;
    return total;

}

int main() {

    int n;
```

```
    std::cin >> n;

    std::string command;
    std::string word;

    Node* root = new Node();

    for(int i = 0; i < n; i++) {


        std::cin >> command >> word;
        std::transform(word.begin(), word.end(), word.begin(), ::tolower);

        if(command == "INSERT") {

            addWord(root, word, 0);

        } else if(command == "LESS") {

            int res = less(root, word, 0);
            if(res==-1)
                std::cout << "NA" << std::endl;
            else
                std::cout << res << std::endl;

        }

    }


    return 0;
}
```