Ինֆորմատիկայի հանրապետական օլիմպիադայի առաջին (դպրոցական) փուլի խնդիրները և նրանց վերլուծությունները

Խնդիրների հերոսների անունները պատահական չեն: Դեկտեմբերի 1-ին Լևոնը Մուրադյանը, Խաժակ Գալստյանը և Համլետ Միքաելյանը ԵՊՀ թիմի կազմում նվաճեցին ծրագրավորման ICPC միջազգային օլիմպիադայի եզրափակիչի ուղեգիր: Դա մեծ նվաճում է, քանի որ եզրափակիչում լինելու են աշխարհի շուրջ 130 համալսարանների թիմեր: Վերջին խնդիրը նվիրված է Քվանտ վարժարանի թիմին, Jigglypuff-ը այդ թիմի անունն է, որի կազում ընդգրկված են Սամվել Անդրեասյանը, Ալեքսանդր Աբելյանը և Արայի Խալաթյանը: Թիմը Ռուսաստանի դպրոցականների ծրագրավորման բաց օլիմպիադայում (ВКОШП) նվաճեց երկրորդ կարգի դիպլոմ, որը Հայաստանի դպրոցական թիմերի համար այդ օլիմպիադայում լավագույն ցուցանիշն է:
Առաջին փուլի խնդիրները կազմել է ԵՊՀ դասախոս Արմեն Անդրեասյանը: Խնդիրների վերլուծությունները պատրաստել են Արմեն Անդրեասյանը և ԵՊՀ ուսանող Վահագն Ալթունյանը:
Խորհուրդ ենք տալիս վերլուծությունները կարդալուց հետո աշխատել այդ խնդիրների վրա՝ գրել ծրագրերը և թեստավորել դրանք am.spoj.com կայքում:

Լևոնը և շոկոլադե սալիկը

Շարադրանք

Ոչ միայն Լևոնը, շատերն են նախընտրում մրցույթի ընթացքում շոկոլադ ուտել: Լևոնը սովորաբար շոկոլադե ուղղանկյունաձև սալիկը զգուշությամբ բաժանում է չորս ուղղանկյունաձև մասերի ուղիղ գծով կտրելով շոկոլադի մի կողմին, ապա մյուս կողմին զուգահեռ գծերով: Հաճախ ստացվում են անհավասար կտորներ: Պատահում է, որ նա այնքան է տարվում խնդիրներով, որ չի ուտում այդ կտորներից վերջինը: Հարկավոր է գտնել այդ կտորի հնարավոր մեծագույն մակերեսը:

Մուտքային տվյալներ

Տրված է շոկոլադե սալիկի w լայնությունը և h բարձրությունը ($2 \le w, h \le 100$): Ապա տրված են a և b թվերը ($1 \le a < w$, $1 \le b < h$), որտեղ a-ն սալիկի բարձրությանը զուգահեռ տարված ուղղի հեռավորությունն է ձախ եզրից, իսկ b-ն լայնությանը զուգահեռ տարված ուղղի հեռավորությունն է ներքևի եզրից: Մուտքում տրված բոլոր թվերը բնական են:

Ելքային տվյալներ

Արտածել մեկ թիվ՝ նշված ուղիղներով կտրելու դեպքում չորս կտորներից մեծագույնի մակերեսը:

Մուտք	*Ելք*
5 10 2 3	21

Լուծումը

Սա մրցույթի ամենահեշտ խնդիրն է: Ակնհայտ է, որ խնդիրը բերվում է 4 թվերից մեծագույնը բերելու խնդրին: Շոկոլադի չորս կտորների մակերեսները դժվար չէ ստանալ՝ a*b, (w-a)*b, a*(h-b) և (w-a)*(h-b): Պահենք այս արժեքները s1, s2, s3, s4 փոփոխականներում: Որպեսզի չսխալվենք մեծագույնը փնտրելիս, այդ արժեքները հերթով համեմատենք ընթացիկ մեծագույնի հետ: C++-ով կոդը այսպիսի տեսք կունենա.

```
#include <iostream>
using namespace std;

int main()
{
   int a, b, w, h, s1, s2, s3, s4, res;
   cin>>w>>h>>a>>b;
   s1 = a*b;
   s2 = (w-a)*b;
   s3 = a*(h-b);
   s4 = (w-a)*(h-b);
   res = s1;
   if(res < s2) res = s2;
   if(res < s3) res = s3;
   if(res < s4) res = s4;
   cout<<res<<endl;
   return 0;
}
```

Կարելի է օգտագործել algorithm գրադարանի max ֆունկցիան: Այդ դեպքում կոդը շատ ավելի կարճ կստացվի:

```
   res = max(a*b, (w-a)*b);
   res = max(res, a*(h-b));
   res = max(res, (w-a)*(h-b));
```

Կամ էլ կարելի է այսպես.

```
   res = max(a, w-a) * max(b, h-b);
```

Համլետը և բարակաբլիթները

Շարադրանք

Մոտենում է նոր տարին և մայրիկները պետք է բարակաբլիթներ (բլինչիկ) պատրաստեն: Համլետը իր մայրիկին օգնելու համար բարակաբլիթ թխող ռոբոտներ է պատրաստել: Ռոբոտի երկու ձեռքերից յուրաքանչյուրը ժամանակի յուրաքանչյուր պահին կարող է կատարել հետևյալ գործողություններից մեկը՝ խմորը շերեփով լցնել թավայի մեջ, շրջել բարակաբլիթը, հանել բարակաբլիթը: Ռոբոտները մի փոքր թերություն ունեն՝ առաջին գործողության ժամանակ շերեփում կարող է տարբեր քանակությամբ խմոր լինել և բարակաբլիթների թխելու ժամանակները տարբեր են ստացվում (դա այլ հանգամանքներից էլ է կախված):

Առաջին փորձերը պսակվեցին հաջողությամբ, և հիմա Համլետը բարակաբլիթներ համտեսելու արագ սննդի կետ է բացել: Հենց հաճախորդներին սպասարկողը մոտենում է խոհանոցի պատուհանին և ասում է պայմանական կոդը՝ ԲԲ, հերթական՝ ազատ ձեռք ունեցող, խոհարար ռոբոտը նույն վայրկյանին պետք է կատարի առաջին գործողությունը: Հենց որևէ բարակբլիթի մի կողմը թխվում է, նույն վայրկյանին այն պետք է շուռ տալ: Երկրորդ կողմը թխվելուն պես պետք է հանել:

Ենթադրենք հայտնի են յուրաքանչյուր բարակաբլիթի առաջին գործողությունը կատարելու պահերը և մի կողմը թխվելու ժամանակը: Առնվազն քանի՞ խոհարար-ռոբոտ է պետք, որպեսզի ոչ մի բարակաբլիթ չփչանա և բոլոր պատվերները կատարվեն:

Մուտքային տվյալներ

Առաջին տողում տրված է բարակաբլիթների պատվերների n քանակը ($1 \le n \le 1000$): Հաջորդ n տողերից յուրաքանչյուրում տրված են երկուական թվեր, առաջին թիվը ցույց է տալիս խմորը թավայի մեջ լցնելու T_i ($1 \le T_i \le 1000$) պահը, երկրորդ թիվը՝ տվյալ բարակաբլիթի մի կողմը թխելու D_i տևողությունը($1 \le D_i \le 1000$): Ենթադրում ենք, որ երկու կողմերն էլ թխվում են նույն ժամանակում:

Ելքային տվյալներ

Պետք է արտածել ճիշտ մեկ բնական թիվ՝ ռոբոտների անհրաժեշտ նվազագույն քանակը:

Մուտք	***Ելք***
3 1 5 1 5 1 5	2
3	1

1 3 5 1 2 2	

Լուծումը

Հարկավոր է պարզել այն պահերը, երբ ռոբոտների ձեռքերի անհրաժեշտություն կա, հաշվել յուրաքանչյուր պահին անհրաժեշտ ձեռքերի քանակը և գտնել դրանցից մեծագույնը: Ինչպե՞ս դա անել: Դրա համար վերցնենք մի օժանդակ a զանգված, որտեղ յուրաքանչյուր բարակաբլիթի համար անհրաժեշտ երեք գործողությունների պահերին զանգվածի համապատասխան տարրերի արժեքները մեկով մեծացնենք: Դա կարելի է անել ներածման ընթացքում՝

```
cin>>n;
for(i=0; i<n; i++){
   cin>>t>>d;
   a[t]++;
   a[t + d]++;
   a[t + 2 * d]++;
}
```

Արդյունքում a զանգվածի i-րդ տարրը ցույց կտա, թե ժամանակի i-րդ պահին քանի գործողություն է պետք կատարել: a զանգվածը սկզբում պետք է զրոյացնել: Դրա համար այն գլոբալ հայտարարենք: Որքա՞ն պետք է լինի a զանգածի չափը: t-ի և d-ի մեծագույն արժեքները 1000 են: Հետևաբար զանգվածի չափը պետք է լինի առնվազն 3001:
Մնում է գտնել այդ զանգվածի մեծագույն տարրի արժեքը: Հետո դա պետք է բաժանել 2-ի, քանի որ յուրաքանչյուր ռոբոտ երկու ձեռք ունի, բայց եթե մեծագույնը կենտ թիվ է, պետք է պատասխանը մեկով մեծացնել:

Շախմատի թագուհին

Շարադրանք

Հայտնի է, որ թագուհին ամենահզոր խաղաքարն է շախմատում: Նա կարող է շարժվել 8 ուղղություններով: Պետք է գրել ծրագիր, որը հաշվում է, թե հակառակորդի քանի խաղաքարի են հարվածում սպիտակ և սև թագուհիները: Համարել, որ կա առավելագույնը մեկ սև և առավելագույնը մեկ սպիտակ թագուհի: Հիշեցնենք, որ թագուհին շարժվում է հորիզոնական, ուղղահայաց ուղղություններով և անկյունագծերով:

Մուտքային տվյալներ

Մուտքում տրված է շախմատի դիրքը՝ սիմվոլների 8 տող, յուրաքանչյուրում 8 սիմվոլ: Սև խաղաքարերը նշված են b տառով, սպիտակ խաղաքարերը՝ w տառով, սպիտակների թագուհին նշված է Q տառով, սևերի թագուհին՝ q տառով:

Ելքային տվյալներ

Ելքում պետք է արտածել մեկ տող, որը պարունակում է երկու թիվ: Առաջին թիվը պետք է ցույց տա, թե քանի սև խաղաքարի է հարվածում սպիտակ թագուհին, երկրորդ թիվը պետք է ցույց տա, թե քանի սպիտակ խաղաքարի է հարվածում սև թագուհին: Եթե սև կամ սպիտակ թագուհին բացակայում է, ապա համապատասխան արժեքը պետք է լինի -1:

Մուտք	*Ելք*
......ww .... wb..q... ....b.... .... .Q.bw...	3 2

Լուծումը

Այս խնդիրը իրականացնելու (ռեալիզացիայի) խնդիր է, այսինքն պետք է երկու չափանի զանգվածի հետ աշխատելու հմտություններ ունենալ այն լուծելու համար և պարզապես ծրագրավորել այն, ինչ խնդրում պահանջվում է: Օրինակ, եթե երկչափանի a զանգվածում գտել ենք q տառը և ուզում ենք ստուգել սևերի թագուհուց ձախ ներքև տանող անկյունագիծը, պետք է այսպիսի ցիկլ կազմակերպենք

```
for(i=x+1, j =y-1; i<8 && j>=0; i++, j--)
  if(a[i][j] != '.')
   break;
```

Այս ցիկլի ավարտից հետո պետք է ստուգել, եթե ցիկլն ընդհատվել է այն պատճառով, որ a[i][j]-ի արժեքը կետ չէ, ուրեմն պետք է ստուգել, եթե a[i][j]-ի արժեքը w տառն է, նշանակում է սևերի թագուհին հարվածում է a[i][j] վանդակում գտնվող սպիտակ խաղաքարին:

Այսպես պետք է ութ ցիկլ կազմակերպել ութ ուղղություններով թագուհու հարվածելը ստուգելու համար: Կոդը երկրորդ թագուհու համար չկրկնելու նպատակով կարելի է ֆունկցիա սահմանել, որտեղ պարամետրերից մեկը ցույց տա, թե ինչ գույնի թագուհու համար են ստուգումներն արվում: Պետք է ուշադիր լինել և բաց չթողնել այն դեպքը, երբ երկու թագուհիներն իրար են հարվածում:

Խաժակը և K-կտորները

Շարադրանք

Երբ Խաժակը դեռ փոքր էր և նոր էր հաճախում օլիմպիական խմբակ, նրան առաջարկեցին հետևյալ խնդիրը. Տրված է թվերի հաջորդականություն: Հարկավոր է հաջորդականության

յուրաքանչյուր k երկարության կտորում հաշվել պարզ թվերի քանակը և արտածել դրանցից մեծագույնը: Խաժակը բավականին արագ գտավ արդյունավետ լուծում և գրեց ծրագիրը: Հիմա հերթը ձերն է:

Մուտքային տվյալներ

Առաջին տողում տրված է հաջորդականության n ($1 \le n \le 50000$) երկարությունը և k թիվը ($1 \le k \le n$): Երկրորդ տողում տրված են իրարից մեկ բացատով անջատված n դրական ամբողջ թվեր, որոնք չեն գերազանցում 10^8-ը:

Ելքային տվյալներ

Ելքում պետք է արտածել մեկ թիվ, k երկարության կտորներից առավելագույն թվով պարզ թվեր պարունակող կտորում պարզ թվերի քանակը:

Մուտք	*Ելք*
6 3 2 5 10 3 6 7	2

Լուծումը

n երկարության հաջորդականությունում k երկարության անընդհատ կտորների քանակը (n-k+1) է: Եթե դրանցից յուրաքանչյուրում պարզ թվերի քանակը իմանանք ապա պատասխանը կլինի այդ թվերից մեծագույնը: Եթե փորձենք ամեն կտորի համար առանձին հաշվել այդ քանակը ապա ժամանակի սահմանափակումը կգերազանցվի:

Դիցուք ունենք a հաջորդականությունը: a[i...j] -ով կնշանակենք (a[i], a[i+1], …, a[j]) կտորը

Ենթադրենք գիտենք պարզ թվերի քանակը a[i. . .i+k-1] կտորի վրա, նշանակենք այդ քանակը c_1: Հաշվենք պարզ թվերի քանակը a[i+1...i+k] կտորի վրա, նշանակենք c_3: Նկատենք որ երկու կտորները ունեն ընդհանուր մաս՝ a[i+1...i+k-1], այստեղ պարզ թվերի քանակը նշանակենք c_2:

Եթե a[i] -ն պարզ թիվ է, ապա $c_1=c_2+1$, հակառակ դեպքում $c_1=c_2$:

Եթե a[i+k]-ն պարզ թիվ է, ապա $c_3=c_2+1$, հակառակ դեպքում $c_3=c_2$:

Կամ եթե սահմանենք prime(n) = {1 եթե n-ը պարզ է, 0 հակառակ դեպքում} ֆունկցիան ապա կարող ենք գրել՝ $c_1=c_2+$prime(a[i]) և $c_3=c_2+$prime(a[i+k]): Միավորելով այս երկուսը կստանանք՝

$$c_3 = c_1 - \text{prime}(a[i]) + \text{prime}(a[i+k])$$

Այսինքն մի k երկարության կտորի պատասխանից մյուսը ստանալը կարող ենք անել 2 քայլով և հարկավոր չէ նորից հաշվել: Այս հնարքը հայտնի է ***սահող պատուհան*** (***sliding window)*** անունով:

prime ֆունկցիան կարելի է իրականացնել հետևյալ կերպ

```
int prime(int n) {
    for(int d = 2; d * d <= n; d++)
        if (n % d == 0)
            return 0;
    return (n == 1 ? 0 : 1); // 1-ը պարզ թիվ չէ
}
```

Jigglypuff

Շարադրանք

Jigglypuff-ը թաքնվել է s տողում: Հարկավոր է գտնել s տողի ամենափոքր նախածանցը, որը պարունակում է "jigglypuff" բառի բոլոր տառերը: Մուտքային տողում կարող են լինել միայն անգլերեն այբուբենի տառեր՝ մեծատառեր կամ փոքրատառեր: Ելքում նախածանցը, որը պարունակում է "jigglypuff" բառի տառերը՝ մեծատառ կամ փոքրատառ, պետք է արտածել այնպես, ինչպես տրված է մուտքային ֆայլում:

Նախածանցը ենթատող է որը ստացվում է տողի վերջից 0 կամ ավելի տառեր ջնջելով: Օրինակ՝ "olymp" տողի նախածանցերն են - "o" "ol" "oly" "olym" "olymp".

Մուտքային տվյալներ

Առաջին տողում տրված է s տողի n երկարությունը, որը չի գերազանցում 123456>-ը: Երկրորդ տողում տրված է մինչև n երկարության միայն անգլերեն այբուբենի մեծատառերից և փոքրատառերից կազմված s տողը:

Ելքային տվյալներ

Արտածել տրված տողի ամենափոքր նախածանցի երկարությունը, ապա նախածանցը, որը պարունակում է "jigglypuff" բառի բոլոր տառերը: Երաշխավորվում է, որ պատասխան միշտ գոյություն ունի:

Մուտք	***Ելք***
26 GGjigbestofALLfyfupsVKOSHP	19 GGjigbestofALLfyfup

Լուծումը

Նախածանցը պետք է պարունակի "jigglypuff"-ի բոլոր տառերը, կամ այլ կերպ ասած պարունակի նվազագույնը 1 հատ "j" սիմվոլ, 1 հատ "i" սիմվոլ, 2 հատ "g" սիմվոլ, 1 հատ "l" սիմվոլ, 1 հատ "y" սիմվոլ, 1 հատ "p" սիմվոլ, 1 հատ "u" և 2 հատ "f" սիմվոլ: Եթե հերթով դիտարկենք 1, 2, 3 … n երկարությամբ նախածանցները, նրանցում նշված տառերի քանակները և պայմանին բավարարելու դեպքում տպենք նախածանցը ու ավարտենք ծրագիրը, ապա լուծումը ճիշտ կլինի, քանի որ մենք գտանք ամենափոքր երկարության այդպիսի նախածանցը:

Եթե ամեն նախածանցի համար մենք զրոյից հաշվենք այդ քանակները, ապա ալգորիթմի կատարած քայլերի քանակը կլինի $1 + 2 + ... + n = n * (n + 1)/2$, որը ահռելի թիվ է n=123456 դեպքում, ուստի պարզ լուծումը չի բավարարի ժամանակի սահմանափակմանը:

Եթե ունենք s[0...k-1] նախածանցի համար այդ քանակները ապա s[0...k] նախածանցի համար այդ քանակները հաշվելու համար բավարար է միայն ստուգել s[k]-ն և դրա արժեքից կախված թարմացնել մեր ունեցած քանակները:

Հարմար է սիմվոլների քանակը պահել ամբողջ թվերի զանգվածի միջոցով

int cnt[26];

Պայմանավորվենք, որ cnt[i] -ն ցույց կտա այբուբենի i-րդ տառի քանակը: Ենթադրենք ինչ որ պահի cnt-ն ցույց է տալիս s[0...k-1] նախածանցում տառերի քանակները: s[0...k] -ի համար քանակները գտնելու համար կգրենք

cnt[tolower(s[k]) - ‘a’]++;

Օգտվեցինք C++ ի ստանդարտ tolower ֆունկցիայից:

Ամեն նախածանցի համար այդ քանակները ստանալուց հետո ընդամենը պետք է ստուգել համապատասխան տառերի քանակները և նվազագույն պայմաններին բավարարելու դեպքում տպել այդ նախածանցը: