Όνομα Άσκησης: Barn Repair

Πηγή: USACO 1.3

Εκφώνηση

Επιδιορθώνοντας τον αχυρώνα

Ήταν μια σκοτεινή θυελλώδης νύκτα όταν ο αχυρώνας του Farmer John έπαθε μεγάλες ζημιές. Ευτυχώς πολλές από τις αγελάδες έλειπαν και ο αχυρώνας δεν καταστράφηκε ολοσχερώς.

Οι αγελάδες πέρασαν τη νύκτα σε πάγκους (stalls) οι οποίοι ήταν τοποθετημένοι σε μια μεγάλη ευθεία γραμμή. Κάποιοι από τους πάγκους είχαν αγελάδες και κάποιοι όχι. Όλοι οι πάγκοι έχουν το ίδιο μέγεθος.

Ο Farmer John πρέπει να δημιουργήσει γρήγορα προστατευτικά μπροστά από τους πάγκους μιας και οι πόρτες του αχυρώνα έχουν καταστραφεί. Ο προμηθευτής ξυλείας του μπορεί να τον εφοδιάσει με προστατευτικά οποιουδήποτε μήκους αλλά έχει περιορισμό στον αριθμό των προστατευτικών που μπορεί να παραδώσει. Ο Farmer John θέλει να ελαχιστοποιήσει το μήκος των προστατευτικών που θα χρησιμοποιήσει.

Δίνονται (σε μία γραμμή) το M (1 <= M <= 50), που αντιστοιχεί στο πλήθος των προστατευτικών που μπορεί να διαθέσει ο προμηθευτής, το S (1 <= S <= 200), που αντιστοιχεί στο σύνολο των πάγκων, το C (1 <= C <= S) που αντιστοιχεί στον αριθμό των αγελάδων που βρίσκονται στους πάγκου. Στη συνέχεια έχουμε C γραμμές ακεραίων αριθμών που αντιστοιχούν στους πάγκους που είναι κατειλημμένοι από αγελάδα (1 <= αριθμός πάγκου <= S).

Υπολογίστε και τυπώστε τον ελάχιστο αριθμό πάγκων που πρέπει να καλυφτούν ώστε να είναι προστατευμένες όλες οι αγελάδες.

Είσοδος

Γραμμή 1:	Οι αριθμοί M, S και C (χωρισμένοι με το κενό διάστημα)
Γραμμή 2-C+1:	Κάθε γραμμή περιέχει έναν ακέραιο αριθμό που αντιστοιχεί στον αριθμό του πάγκου που είναι κατειλημμένος.

Παράδειγμα Εισόδου (file barn1.in)

```
4 50 18
3
4
6
8
14
15
16
17
21
25
26
27
30
31
40
41
42
43
```

Έξοδος

Ένας ακέραιος αριθμός που αντιστοιχεί στον ελάχιστο αριθμό πάγκων που έχουν καλυφτεί.

Παράδειγμα εξόδου (file barn1.out)

```
25
```

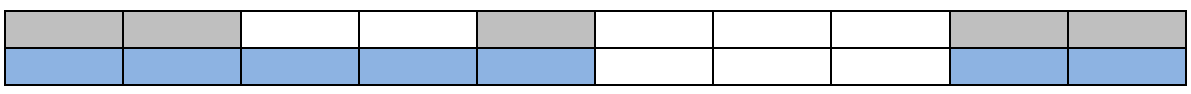
Επεξήγηση: Στο παράδειγμα μας έχουμε 4 προστατευτικά που καλύπτουν τους πάγκους [3-8], [14-21], [25-31] και [40-43].

Επεξήγηση

Σκεφτείτε ότι έχετε στη σειρά S κουτιά (πάγκοι) και κάποια από αυτά είναι γεμάτα και κάποια όχι. Στόχος είναι να βρείτε τρόπο να καλύψετε όλα τα κουτιά που είναι γεμάτα έχοντας στη διάθεση σας M καλύμματα.

Για παράδειγμα έστω ότι S =10 και είναι καλυμμένα τα κουτιά 1,2,5, 9,10. Επίσης έχετε στη διάθεση σας 2 καλύμματα.

Πάγκοι



Καλύμματα

Hints

Βρείτε τα μεγαλύτερα κενά μεταξύ πάγκων και εκμεταλλευτείτε τα. Καλό είναι αυτά να μη τα καλύψετε. Προσέξετε την περίπτωση που τα καλύμματα είναι περισσότερα από τις αγελάδες.

Λύση

Το πρόβλημα αυτό ανήκει στην κατηγορία των προβλημάτων που λύνονται με άπληστο (greedy) τρόπο. Το πρώτο πράγμα που πρέπει να κάνουμε είναι να βρούμε τον πρώτο (low) και τον τελευταίο (high) πάγκο στον οποίο υπάρχει αγελάδα. Αμέσως με αυτό τον τρόπο θα επικεντρωθούμε στους πάγκους από low μέχρι high. Οι πάγκοι που είναι μικρότεροι του low και μεγαλύτεροι του high δεν θα καλυφθούν σίγουρα.

Στη συνέχεια προσπαθούμε να βρούμε τα σημεία που είναι καλύτερα να μην καλυφτούν γιατί θα μας στοίχιζαν πολύ σε μέγεθος. Προφανώς είναι τα σημεία όπου υπάρχει μεγάλο κενό μεταξύ των αγελάδων. Υπολογίζουμε για όλους τους πάγκους τις αποστάσεις μέχρι να συναντήσουν ξανά γεμάτο πάγκο, τα ταξινομούμε σε φθίνουσα σειρά και παίρνουμε τα M-1 τις πρώτες M-1 θέσεις.

Στο παράδειγμα εισόδου έχουμε M=4 άρα θα βρούμε τις 3 μεγαλύτερες αποστάσεις μεταξύ των πάγκων. Παρατηρούμε ότι αυτές είναι από τον 31 μέχρι τον 40, από τον 8 μέχρι τον 14 και από τον 17 μέχρι τον 21 (ή θα μπορούσαμε να πάρουμε την απόσταση 21 μέχρι 25). Αυτές είναι περιοχές που θέλουμε να αφήσουμε ακάλυπτες άρα οι υπόλοιπες θα καλυφτούν.

```
#include<fstream>
 #include<vector>
 #include<algorithm>
 #include<iostream>
 using namespace std;


 int main(){
  int M,S,C,i,j,stall,low,high,sum,temp,start,finish;
 //Παρόμοιο με το dist. Θα περιέχει μόνο τα stalls που θα χρησιμοιοιηθούν
  int boards[50][2]={};
  // Το a έχει μέσα τον αριθμό των stall που έχουν αγελάδα
  //Το oStalls είναι ένας πίνακα με όλα τα διαθέσιμα stalls. Σημειώνει με 1 αυτά που έχουν αγελάδα
  vector< int > a,oStalls;
 //Στην πρώτη στήλη θα αποθηκεύετε η απόσταση ενός stall που έχει αγελάδα
 // μέχρι το επόμενο. Στη δεύτερη θα είναι ο αριθμός του stall
  vector< pair<int, int> > dist;
  ifstream fin("barn1.in");
  ofstream fout("barn1.out");


  fin>>M>>S>>C;
  //Αν ο αριθμός των διαθέσιμων boards είναι μεγαλύτερος από τα stalls που θα χρησιμοποιηθούν τότε
  //μπορούμε να έχουμε ένα board για κάθε stall
  if(M>C){
    sum=C;
```

```
}
else{
for(i=0;i<S;i++){
  dist.push_back(make_pair(0,i));
  oStalls.push_back(0);
}
for(i=0;i<C;i++){
    fin>>stall;
    a.push_back(stall);
    oStalls[stall-1]=1;
            }
sort(a.begin(),a.end());

//Σημειώνω τον αριθμού του πρώτου και του τελευταίου stall που έχουν αγελάδα
low=a[0];
high=a[C-1];

//Βρίσκω τον αριθμό των κενών stall που υπάρχουν ξεκινώντας από ένα που χρησιμοποιήται
for(i=low-1;i<high;i++){
 sum=0;
 j=i;
  if(oStalls[i]==1){
    while((j<high-1)&&(oStalls[j+1]==0)){
      sum++;
      j++;
```

```
      }
      dist[i].first=sum;
    }

  }

 sort(dist.begin(),dist.end());



 boards[M-1][0]=low-1;
 boards[M][0]=high-1;
 for(i=0;i<M-1;i++){
   boards[i][0]=dist[S-1-i].second;
   boards[i][1]=dist[S-1-i].first;
 }
//Bubble sort
for(i=0;i<M;i++)
 for(j=i+1;j<M+1;j++)
  if(boards[i][0]>boards[j][0]){
    temp=boards[i][0];
    boards[i][0]=boards[j][0];
    boards[j][0]=temp;
    temp=boards[i][1];
    boards[i][1]=boards[j][1];
    boards[j][1]=temp;
```

```
  }
sum=0;
j=0;

//Βρίσκω το άθροισμα των stall που καλύφτηκαν
start=boards[0][0];
for(i=1;i<M+1;i++){
    finish=boards[i][0];
    sum=sum+(finish-start+1);
    start=finish+1+boards[i][1];
  }
}
 fout<<sum<<"\n";

  return 0;
}
```