USA COMPUTING OLYMPIAD

Barn Repair

Russ Cox

If we can purchase M boards, then we can leave unblocked M-1 runs of stalls without cows in them, in addition to any stalls on the leftmost side that don't have cows and any stalls on the rightmost side that don't have cows.

We input the list of cows in stalls, storing into an array whether or not there is a cow in a particular stall. Then we walk the array counting sizes of runs of cowless stalls. We sort the list of sizes and pick the M-1 largest ones as the stalls that will remain uncovered.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#define MAXSTALL 200
int hascow[MAXSTALL];

int
intcmp(const void *va, const void *vb)
{
        return *(int*)vb - *(int*)va;
}

void
main(void)
{
    FILE *fin, *fout;
    int n, m, nstall, ncow, i, j, c, lo, hi, nrun;
    int run[MAXSTALL];

    fin = fopen("barn1.in", "r");
    fout = fopen("barn1.out", "w");

    assert(fin != NULL && fout != NULL);

    fscanf(fin, "%d %d %d", &m, &nstall, &ncow);
    for(i=0; i<ncow; i++) {
        fscanf(fin, "%d", &c);
        hascow[c-1] = 1;
    }

    n = 0;      /* answer: no. of uncovered stalls */

    /* count empty stalls on left */
    for(i=0; i<nstall && !hascow[i]; i++)
        n++;
    lo = i;

    /* count empty stalls on right */
    for(i=nstall-1; i>=0 && !hascow[i]; i--)
        n++;
    hi = i+1;

    /* count runs of empty stalls */
    nrun = 0;
    i = lo;
```

```
    while(i < hi) {
        while(hascow[i] && i<hi)
            i++;

        for(j=i; j<hi && !hascow[j]; j++)
            ;

        run[nrun++] = j-i;
        i = j;
    }

    /* sort list of runs */
    qsort(run, nrun, sizeof(run[0]), intcmp);

    /* uncover best m-1 runs */
    for(i=0; i<nrun && i<m-1; i++)
        n += run[i];

    fprintf(fout, "%d\n", nstall-n);
    exit(0);
}
```

Alexandru Tudorica's solution might be simpler:

```
var f:text;
    a,b:array[1..1000] of longint;
    i,m,s,c,k:longint;

procedure qsort(l,r:longint);
    var i,j,x,y:longint;
begin
     i:=l; j:=r; x:=a[(l+r) div 2];
     repeat
           while a[i]<x do i:=i+1;
           while x<a[j] do j:=j-1;
           if i<=j then begin
                y:=a[i]; a[i]:=a[j]; a[j]:=y;
                i:=i+1;
                j:=j-1;
           end;
     until i>j;
     if l<j then qsort(l,j);
     if i<r then qsort(i,r);
end;

procedure qsortb(l,r:longint);
    var i,j,x,y:longint;
begin
     i:=l; j:=r; x:=b[(l+r) div 2];
     repeat
           while b[i]<x do i:=i+1;
           while x<b[j] do j:=j-1;
           if i<=j then
           begin
                y:=b[i]; b[i]:=b[j]; b[j]:=y;
                i:=i+1;
                j:=j-1;
           end;
     until i>j;
     if l<j then qsortb(l,j);
     if i<r then qsortb(i,r);
end;


begin
     assign(f,'barn1.in');
     reset(f);
     readln(f,m,k,c);
     for i:=1 to c do readln(f,a[i]);
     qsort(1,c);
```

```
    for i:=1 to c-1 do b[i]:=a[i+1]-a[i]-1;
    qsortb(1,c-1);
    for i:=c-1 downto (c-m+1) do s:=s+b[i];
    close(f);
    assign(f,'barn1.out');
    rewrite(f);
    writeln(f,a[c]-a[1]-s+1);
    close(f);
end.
```

USACO Gateway | Comment or Question