Solutions

	Statistics
NWERC 2011	Problem E
Solutions to the problems	Problem B
Solutions to the problems	Problem C
	Problem A
The Jury Jacobs University Bremen	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F



## Statistics

Solutions

Е

problem	correct/submissions	fastest	
E - Please, go first	62/106	25	Statistics
B - Bird Tree	44/83	34	Problem I
C - Move to Front	33/171	20	Problem I
A - Binomial Coefficients	18/135	22	Problem
H - Tichu	13/40	115	Problem
I - Tracking RFIDs	7/31	88	Problem I
G - Smoking Gun	3/63	134	Problem
D - Piece it Together	3/39	260	Problem I
J - Train delay	1/9	279	Problem I
F - Pool construction	0/2	N/A	Problem .
	/	/	Problem I

The end







event

sponsor

Sample case 2: Ab9AAb2bC2	Solutions
The last person in the line will stay the last person	
All his friends line up in front of him. ALCAALLCOO	Statistics
All his menus line up in none of him. Ab9AAbbC22	Problem E
	Problem B
	Problem C
	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end

event sponsor Calutions



	Sample	case 2	: Ab9AAb2bC <mark>2</mark>	
--	--------	--------	----------------------------	--

- The last person in the line will stay the last person
- All his friends line up in front of him: Ab9AAbbC22
- Then the last person that isn't his friend: Ab9AAbbC22

#### Solutions

#### Statistics

### Problem E

- Problem B
- Problem C
- Problem A
- Problem H
- Problem G
- Problem I
- Problem D
- Problem J
- Problem F

The end



	Sample	case	2:	Ab9AAb2bC <mark>2</mark>
--	--------	------	----	--------------------------

- The last person in the line will stay the last person
- All his friends line up in front of him: Ab9AAbbC22
- Then the last person that isn't his friend: Ab9AAbbC22
- And his friends (in this case none)

#### Solutions

#### Statistics

### Problem E

- Problem B
- Problem C
- Problem A
- Problem H
- Problem G
- Problem I
- Problem D
- Problem J
- Problem F

The end



Sample case 2: Ab9AAb2bC2	Solutions
The last person in the line will stay the last person	
All his friends line up in front of him: Ab9AAbbC22	Statistics
Then the last person that isn't his friend: Ab9AAbbC22	Problem E Problem B
And his friends (in this case none)	Problem C
Then the next: AbQAbbC22	Problem A
Fillen the next. Ab9AAbbC22	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end



Sample case 2: Ab9AAb2bC2	Solutions
The last person in the line will stay the last person	
All his friends line up in front of him: Ab9AAbbC22	Statistics Problem F
Then the last person that isn't his friend: Ab9AAbbC22	Problem B
And his friends (in this case none)	Problem C
Then the next: Ab9AAbbC22	Problem A
And his friends: A9AAbbbC22	Problem H
	Problem I
	Problem D
	Problem J
	Problem F

The end



Sample case 2: Ab9AAb2bC2	Solutions
The last person in the line will stay the last person	
All his friends line up in front of him: Ab9AAbbC22	Statistics Problem F
Then the last person that isn't his friend: Ab9AAbbC22	Problem B
And his friends (in this case none)	Problem C
Then the next: Ab9AAbbC22	Problem A Problem H
And his friends: A9AAbbbC22	Problem G
And so on. Final order: 9AAAbbbC22	Problem I
	Problem D
	Problem J

Problem F

The end



Sample case 2: Ab9AAb2bC2	Solutions
The last person in the line will stay the last person	
All his friends line up in front of him: Ab9AAbbC22	Statistics Problem F
Then the last person that isn't his friend: Ab9AAbbC22	Problem B
And his friends (in this case none)	Problem C
► Then the next: Ab9AAbbC22	Problem A
And his friends: A9AAbbbC22	Problem H
<ul> <li>And so on. Final order: 9AAAbbbC22</li> </ul>	Problem I
Now you know the final ordering count the saved time	Problem D
s now you know the multipleting, count the saved time	Problem J
	Problem F

Contractional Collegiate Collegiate Collegiate Collegiate

Sample case 2: Ab9AAb2bC2	Solutions
The last person in the line will stay the last person	
All his friends line up in front of him: Ab9AAbbC22	Statistics Problem F
Then the last person that isn't his friend: Ab9AAbbC22	Problem B
And his friends (in this case none)	Problem C
► Then the next: Ab9AAbbC22	Problem A
And his friends: A9AAbbbC22	Problem H Problem G
And so on. Final order: 9AAAbbbC22	Problem I
Now you know the final ordering, count the saved time	Problem D
Time saved by X is number of positions that the last X	Problem J
moved forward times the number of Xs	Problem F The end



## E - Please, go first (source code)

#include <iostream></iostream>	
#include <vector></vector>	Solutions
#include <string></string>	
#include <cctype></cctype>	
using namespace sta;	
int main () {	Statistics
int runs;	Problem E
<pre>cin &gt;&gt; runs;</pre>	
while (runs) {	Problem B
int n;	Problem C
string s;	Durildana A
$cin \gg n \gg s;$	Problem A
vector <int> cnt(128,0);</int>	Problem H
<pre>for (int i=0; i<n; cnt[s[i]]++;<="" i++)="" pre=""></n;></pre>	Problem G
<pre>int res = 0, num_used = 0;</pre>	Problem I
<pre>vector<bool> used(128,false);</bool></pre>	
	Problem D
for (int i=n-1; i>=0; i) {	
if (!used[s[i]]) {	Problem J
<pre>res += (num_used-(n-1-1))*cnt[s[1]]; num_used += cnt[s[1]];</pre>	Problem F
used[s[i]] = true;	The end
}	
}	
cout << 5 * res << endl;	
}	
return 0;	
Com International Collegiate      I国     International Collegiate     I国     International Collegiate     I国     International Collegiate     I国     International Collegiate     I国     International Collegiate     III     International Collegiate     IIIII     International Collegiate     IIII     International Collegiate     III     International Collegiate     III     International Collegiate     III     International Collegiate     III     International Collegiate     IIIII     International Collegiate     International Collegiate     IIII     International Collegiate     International Collegiate     International Collegia	

	Root:	1/1,	left:	1/(T)	+1),	right:1	+ 1/T
--	-------	------	-------	-------	------	---------	-------

• If a/b < 1, go left, else go right

Statistics

Solutions

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event

sponsor

- Root: 1/1, left: 1/(T + 1), right: 1 + 1/T
- If a/b < 1, go left, else go right
- ▶ If left, replace  $a/b \rightarrow (b/a) 1 = (b a)/a$

Solutions

Statistics

Problem E

### Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- Root: 1/1, left: 1/(T + 1), right: 1 + 1/T
- If a/b < 1, go left, else go right
- ▶ If left, replace  $a/b \rightarrow (b/a) 1 = (b a)/a$
- If right, replace  $a/b \rightarrow 1/(a/b-1) = b/(a-b)$

Solutions

Statistics

Problem E

### Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- ► Root: 1/1, left: 1/(*T* + 1), right:1 + 1/*T*
- If a/b < 1, go left, else go right
- ▶ If left, replace  $a/b \rightarrow (b/a) 1 = (b a)/a$
- If right, replace  $a/b \rightarrow 1/(a/b-1) = b/(a-b)$
- Proceed with first step

Solutions

Statistics

Problem E

### Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



Solutions • Root: 1/1, left: 1/(T+1), right: 1+1/T• If a/b < 1, go left, else go right ▶ If left, replace  $a/b \rightarrow (b/a) - 1 = (b - a)/a$ • If right, replace  $a/b \rightarrow 1/(a/b-1) = b/(a-b)$ Problem B Proceed with first step Stop once you encounter a/b = 1/1



## B - Bird tree (source code)

Ę

<pre>#include <iostream> #include <string></string></iostream></pre>					Solutions
using namespace std;					
int main () {				S	tatistics
int runs; cin>>runs;				Ρ	roblem E
while (runs) {				Р	roblem B
dank a bi				P	roblem C
char c;				Р	roblem A
cin >> a >> c >> b;				Р	roblem H
while (a>1    b>1) { if (a <b) th="" {<=""><td></td><td></td><td></td><td>P</td><td>roblem G</td></b)>				P	roblem G
cout << "L";				P	roblem I
b -= a; }				Р	roblem D
<pre>else {     cout &lt;&lt; "R";</pre>				P	roblem J
a -= b; }				Ρ	roblem F
<pre>swap(a,b); }</pre>				Т	he end
<pre>cout &lt;&lt; endl; }</pre>					
<pre>return 0; }</pre>	acm International Collegiate Programming Contest	IBM.	event sponsor		

► Note, since m, r = 100 000, you cannot update the stack in O(m) time
Solutions

Problem C

event

sponsor



- Note, since m, r = 100000, you cannot update the stack in O(m) time
- Therefore you need some smart data structure to store information

### Solutions

Statistics

Problem E

Problem B

### Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- ► Note, since m, r = 100 000, you cannot update the stack in O(m) time
- Therefore you need some smart data structure to store information
- Binary indexed tree/Fenwick tree does the job in *O*(log(m)) time



#### Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end

▶ At every step, take movie x and put it back on -1,-2,-3,...



Note that this is also O(n<sup>2</sup>):
for (int j = 0; j < r; j++) {
 movie = sc.nextInt();
 index = movies.indexOf(movie);
 System.out.print("" + (m - index - 1) + " ");
 movies.removeElementAt(index);
 movies.add(movie);
}
</pre>

Problem D

Problem J

Problem F

The end



Find *n*, *k* such that 
$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$$

Statistics

Problem E

Problem B

Problem C

### Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- A Binomial coefficients (1)
  - Find n, k such that  $\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$
  - ► Only look for solutions with k ≤ n/2 and count twice if necessary

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



Find 
$$n, k$$
 such that  $\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$ 

► Only look for solutions with k ≤ n/2 and count twice if necessary

• Loop over k from 0 to 
$$\binom{2k}{k} > x$$

Solutions

Statistics

Problem E

Problem B

Problem C

### Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event

sponsor

Find 
$$n, k$$
 such that  $\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$ 

- ► Only look for solutions with k ≤ n/2 and count twice if necessary
- Loop over k from 0 to  $\binom{2k}{k} > x$
- Binary search for n

Solutions

Statistics

Problem E

Problem B

Problem C

### Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



Find 
$$n, k$$
 such that  $\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$ 

- ► Only look for solutions with k ≤ n/2 and count twice if necessary
- Loop over k from 0 to  $\binom{2k}{k} > x$
- Binary search for n
  - Be really careful with overflows!

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



Find *n*, *k* such that 
$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$$

Statistics

Problem E

Problem B

Problem C

### Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



Find 
$$n, k$$
 such that  $\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$ 

► Again, only look for solutions with k ≤ n/2 and count twice if necessary Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event

sponsor

Find 
$$n, k$$
 such that  $\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$ 

► Again, only look for solutions with k ≤ n/2 and count twice if necessary

For 
$$k = 1$$
, solution is  $\binom{x}{1}$ 

Statistics

Solutions

Problem E

Problem B

Problem C

### Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event

sponsor

- A Binomial coefficients (2)
  - Find n, k such that  $\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$
  - ► Again, only look for solutions with k ≤ n/2 and count twice if necessary

For 
$$k = 1$$
, solution is  $\binom{x}{1}$  Problem B

For 
$$k = 2$$
, solve  $x = \binom{n}{2} = \frac{1}{2}n(n-1)$ 

Problem A

Solutions

- Problem H
- Problem G
- Problem I
- Problem D
- Problem J
- Problem F

The end



Find 
$$n, k$$
 such that  $\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$ 

• Again, only look for solutions with  $k \le n/2$  and count twice if necessary

For 
$$k = 1$$
, solution is  $\binom{x}{1}$  Problem B

For 
$$k = 2$$
, solve  $x = \binom{n}{2} = \frac{1}{2}n(n-1)$ 

For 
$$k \ge 3$$
, loop over *n* until  $\binom{n}{k} > x$ 

Solutions



Find 
$$n, k$$
 such that  $\binom{n}{k} = \frac{n!}{k!(n-k)!} = x$ 

• Again, only look for solutions with  $k \le n/2$  and count twice if necessary

For 
$$k = 1$$
, solution is  $\binom{x}{1}$  Problem B

For 
$$k = 2$$
, solve  $x = \binom{n}{2} = \frac{1}{2}n(n-1)$ 

For 
$$k \ge 3$$
, loop over *n* until  $\binom{n}{k} > x$ 

Again, be really careful with overflows!

Solutions

### Problem A



## H - Tichu (1)

### Greedy solution

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



## H - Tichu (1)

<ul> <li>Greedy solution</li> </ul>	Solutions
<ul> <li>Brute force over two straights</li> </ul>	
J. J	Statistics
	Problem E
	Problem B
	Problem C
	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end



# H - Tichu (1)

	Greedy solution	Solutions
•	Brute force over two straights Greedily take quads, full houses, trips, pair and singletons from the remaining cards	Statistics Problem E Problem B Problem C
		Problem A Problem H
		Problem G Problem I Problem D
		Problem J Problem F The end


# H - Tichu (1)

Greedy solution	Solutions
Brute force over two straights	
Greedily take guads, full houses, trips, pair and singletons	Statistics
	Problem E
from the remaining cards	Problem B

• One tricky case: 2 full houses is better than 1 quads + 2trips

Problem H



event

# H - Tichu (2)

Bitmask dynamic programming solution

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event

# H - Tichu (2)

Solutions Bitmask dynamic programming solution For each subset  $(2^{13} = 8192)$  determine whether it is a valid combination Problem H



# H - Tichu (2)

- Bitmask dynamic programming solution
- ► For each subset (2<sup>13</sup> = 8192) determine whether it is a valid combination
- DP step: best[x] = best[x&!y] + 1 with x, y bitmasks, x&y = y and y a valid combination

#### Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



► Calculate t<sub>ij</sub>, the minimal time difference between i shooting and j shooting: t<sub>i</sub> ≤ t<sub>j</sub> + t<sub>ij</sub>

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event

- ► Calculate t<sub>ij</sub>, the minimal time difference between i shooting and j shooting: t<sub>i</sub> ≤ t<sub>i</sub> + t<sub>ij</sub>
- "k heard i shoot before j" leads to  $t_{ij} = d_{kj} d_{ki}$

Statistics

Solutions

Problem E

Problem B

Problem C

Problem A

Problem H

#### Problem G

Problem I

Problem D

Problem J

Problem F

The end



- ► Calculate t<sub>ij</sub>, the minimal time difference between i shooting and j shooting: t<sub>i</sub> ≤ t<sub>i</sub> + t<sub>ij</sub>
- "k heard i shoot before j" leads to  $t_{ij} = d_{kj} d_{ki}$
- Use Floyd-Warshall to draw inferences:  $t_{ij} \leq t_{ik} + t_{kj}$

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- ► Calculate t<sub>ij</sub>, the minimal time difference between i shooting and j shooting: t<sub>i</sub> ≤ t<sub>i</sub> + t<sub>ij</sub>
- "k heard i shoot before j" leads to  $t_{ij} = d_{kj} d_{ki}$
- Use Floyd-Warshall to draw inferences:  $t_{ij} \leq t_{ik} + t_{kj}$
- This is all information that needs to be obtained

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

#### Problem G

Problem I

Problem D

Problem J

Problem F

The end



- ► Calculate t<sub>ij</sub>, the minimal time difference between i shooting and j shooting: t<sub>i</sub> ≤ t<sub>j</sub> + t<sub>ij</sub>
- "k heard i shoot before j" leads to  $t_{ij} = d_{kj} d_{ki}$
- Use Floyd-Warshall to draw inferences:  $t_{ij} \leq t_{ik} + t_{kj}$
- This is all information that needs to be obtained
- If  $t_{ii} < 0$ , it is impossible

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

#### $\mathsf{Problem}\ \mathsf{G}$

Problem I

Problem D

Problem J

Problem F

The end



- ► Calculate t<sub>ij</sub>, the minimal time difference between i shooting and j shooting: t<sub>i</sub> ≤ t<sub>j</sub> + t<sub>ij</sub>
- "k heard i shoot before j" leads to  $t_{ij} = d_{kj} d_{ki}$
- Use Floyd-Warshall to draw inferences:  $t_{ij} \leq t_{ik} + t_{kj}$
- This is all information that needs to be obtained
- If  $t_{ii} < 0$ , it is impossible
- Otherwise, find *i* such that  $t_{ij} < 0$  for all *j*

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

 $\mathsf{Problem}\ \mathsf{G}$ 

Problem I

Problem D

Problem J

Problem F



- ► Calculate t<sub>ij</sub>, the minimal time difference between i shooting and j shooting: t<sub>i</sub> ≤ t<sub>j</sub> + t<sub>ij</sub>
- "k heard i shoot before j" leads to  $t_{ij} = d_{kj} d_{ki}$
- Use Floyd-Warshall to draw inferences:  $t_{ij} \leq t_{ik} + t_{kj}$
- This is all information that needs to be obtained
- If  $t_{ii} < 0$ , it is impossible
- Otherwise, find *i* such that  $t_{ij} < 0$  for all *j*
- If multiple, it is unknown

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

 $\mathsf{Problem}\ \mathsf{G}$ 

Problem I

Problem D

Problem J

Problem F



- ► Calculate t<sub>ij</sub>, the minimal time difference between i shooting and j shooting: t<sub>i</sub> ≤ t<sub>j</sub> + t<sub>ij</sub>
- "k heard i shoot before j" leads to  $t_{ij} = d_{kj} d_{ki}$
- Use Floyd-Warshall to draw inferences:  $t_{ij} \leq t_{ik} + t_{kj}$
- This is all information that needs to be obtained
- If  $t_{ii} < 0$ , it is impossible
- Otherwise, find *i* such that  $t_{ij} < 0$  for all *j*
- If multiple, it is unknown
- Otherwise, this gives the unique solution

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



 To determine whether a sensor can see product, calculate the distance between them Solutions

Statistics
Problem E
Problem B
Problem C
Problem A
Problem H
Problem G
Problem I
<b>Problem I</b> Problem D
<b>Problem I</b> Problem D Problem J
Problem I Problem D Problem J Problem F
Problem I Problem D Problem J Problem F The end
Problem I Problem D Problem J Problem F The end

event



- I Tracking RFIDs
  - To determine whether a sensor can see product, calculate the distance between them
  - Subtract the number of intersecting walls and compare this with r

#### Solutions

- Statistics
- Problem E
- Problem B
- Problem C
- Problem A
- Problem H
- Problem G
- Problem I
- Problem D
- Problem J
- Problem F
- The end



- I Tracking RFIDs
  - To determine whether a sensor can see product, calculate the distance between them
  - Subtract the number of intersecting walls and compare this with r
  - Problem: you cannot do this for all pairs of sensors and products

#### Solutions

- Statistics
- Problem E
- Problem B
- Problem C
- Problem A
- Problem H
- Problem G
- Problem I
- Problem D
- Problem J
- Problem F
- The end



- To determine whether a sensor can see product, calculate the distance between them
- Subtract the number of intersecting walls and compare this with r
- Problem: you cannot do this for all pairs of sensors and products
- Note: since sensors are separated by at least r, only a few sensors can possibly be in range of a product

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- To determine whether a sensor can see product, calculate the distance between them
- Subtract the number of intersecting walls and compare this with r
- Problem: you cannot do this for all pairs of sensors and products
- Note: since sensors are separated by at least r, only a few sensors can possibly be in range of a product
- One possible solution: store all sensors in a search tree (e.g. C++'s set or Java's TreeSet)

#### Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F



- To determine whether a sensor can see product, calculate the distance between them
- Subtract the number of intersecting walls and compare this with r
- Problem: you cannot do this for all pairs of sensors and products
- Note: since sensors are separated by at least r, only a few sensors can possibly be in range of a product
- One possible solution: store all sensors in a search tree (e.g. C++'s set or Java's TreeSet)
- For each product, look if a sensor at (x + δx, y + δy) exists for −r ≤ δx, δy ≤ r



#### Solutions

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

- To determine whether a sensor can see product, calculate the distance between them
- Subtract the number of intersecting walls and compare this with r
- Problem: you cannot do this for all pairs of sensors and products
- Note: since sensors are separated by at least r, only a few sensors can possibly be in range of a product
- One possible solution: store all sensors in a search tree (e.g. C++'s set or Java's TreeSet)
- For each product, look if a sensor at  $(x + \delta x, y + \delta y)$ exists for  $-r \le \delta x, \delta y \le r$
- More difficult solutions using binning, quad trees are also possible



#### Solutions

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

- I Tracking RFIDs (test case)
  - A maximum of 6 sensors can be in range of a product:







event sponsor Solutions

Solution: not matching/max.flow

Statistics

Solutions

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event

<ul> <li>Solution: not matching/max.flow</li> <li>Solution: not backtrack</li> </ul>	Solutions
	Statistics
	Problem E
	Problem B
	Problem C
	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end

event



<ul> <li>Solution: not matching/max.flow</li> <li>Solution: not backtrack</li> </ul>	Solutions
S Solution: 2SAT	statistics
P Solution. 25A1	Problem E
P	Problem B
P	Problem C
P	Problem A
Р	Problem H
P	Problem G
Р	Problem I
Р	Problem D
Р	Problem J
P	Problem F
Т	The end

event



<ul> <li>Solution: not matching/max.flow</li> <li>Solution: not backtrack</li> </ul>	Solutions
Solution: 2SAT	Statistics
	Problem E
First, check white $= 2 \times black$	Problem B
	Problem C
	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end



event

Solution: not matching/max.flow	Solutions
Solution: not backtrack	
Solution: 2SAT	Statistics
First, check white = $2 \times black$	Problem B
Boolean variables: x is part of the same puzzle piece as y	Problem C
(x, y  adjacent)	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end



event

<ul> <li>Solution: not matching/max.flow</li> <li>Solution: not backtrack</li> </ul>	Solutions
<ul> <li>Solution: Not backflack</li> <li>Solution: 2SAT</li> <li>First, shash white = 2 × hlash</li> </ul>	Statistics Problem E
<ul> <li>First, check white = 2 × black</li> <li>Boolean variables: x is part of the same puzzle piece as y</li> <li>(x, y, adjacent)</li> </ul>	Problem B Problem C Problem A
<ul> <li>Black square should be connected to its left xor right white neighbor: (A B)&amp;(1A 1B)</li> </ul>	Problem H Problem G
$\operatorname{Heighbor}(A D) \otimes (A D)$	Problem I Problem D
	Problem J Problem F The end



Solution: not matching/max.flow	Solutions
Solution: not backtrack	<b>C</b> 1.11.11
Solution: 2SAT	Statistics
First, check white = $2 \times black$	Problem E Problem B
Boolean variables: x is part of the same puzzle piece as y	Problem C
(x, y adjacent)	Problem A
Black square should be connected to its left yor right white	Problem H
peighbor: $(A B)\&( A  B)$	Problem G
$(A D) \otimes (A D)$	Problem I
Identically, it should be connected to its upper or lower	Problem D
neighbor	Problem J
	Problem F
	The end



event

Solution: not matching/max.flow	Solutions
Solution: not backtrack	
Solution: 2SAT	Statistics
First sheet tite One block	Problem E
First, check white $= 2 \times black$	Problem B
Boolean variables: x is part of the same puzzle piece as y	Problem C
(x, y  adjacent)	Problem A
Plack cause chould be connected to its left yer right white	Problem H
Diack square should be connected to its left for right white	Problem G
neighbor: $(A B)\&(!A !B)$	Problem I
Identically, it should be connected to its upper or lower	Problem D
neighbor	Problem J
White square should be connected to at most one black	Problem F
square: $( A  B) \& ( A  C) \& ( B  C)$ etc.	The end

event



<ul> <li>Solution: not matching/max.flow</li> <li>Solution: not backtrack</li> </ul>	Solutions
Colution. Not backtrack	Statistics
Solution: 25A1	Problem E
First, check white $= 2 \times black$	Problem B
Boolean variables: x is part of the same puzzle piece as y	Problem C
(x, y  adjacent)	Problem A
Black square should be connected to its left yor right white	Problem H
point square should be connected to its left xor light write noighbor: $(A B)k( A  B)$	Problem G
$\operatorname{Heighbor}(A D) \otimes (A B)$	Problem I
Identically, it should be connected to its upper or lower	Problem D
neighbor	Problem J
White square should be connected to at most one black	Problem F
square: $( A  B) \& ( A  C) \& ( B  C)$ etc.	The end
Now you can use a standard 2SAT solution	



event

- D Piece it together (2)
  - Solutions But this problem has way more structure! Problem D



event

<ul> <li>But this problem has way more structure!</li> <li>You can divide the problem in four subproblems</li> </ul>	Solutions
	Statistics
	Problem E
	Problem B
	Problem C
	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end

event



<ul> <li>But this problem has way more structure!</li> <li>You can divide the problem in four subproblems</li> </ul>	Solutions Statistics
▶ Take white squares at $x = a \mod 2$ and $y = b \mod 2$ ( $a, b = 0, 1$ ) and adjacent black squares	Problem E Problem B
	Problem C
	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end



event

But this problem has way more structure!	Solutions
You can divide the problem in four subproblems	Statistics
• Take white squares at $x = a \mod 2$ and $y = b \mod 2$	Problem E
(a, b = 0, 1) and adjacent black squares	Problem B
Each connected component of these subproblems should	d Problem C
have <i>white</i> = <i>black</i>	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end



	But this problem has way more structure!	Solutions
	You can divide the problem in four subproblems	
•	Take white squares at $x = a \mod 2$ and $y = b \mod 2$ (a, b = 0, 1) and adjacent black squares	Statistics Problem E Problem B
•	Each connected component of these subproblems should have <i>white</i> = <i>black</i>	Problem C Problem A
•	If so, it's possible to solve the puzzle, otherwise, it's not	Problem H Problem G
		Problem I Problem D
		Problem J
		Problem F
		The end



#### D - Piece it together (source code)

```
Solutions
#include <iostream>
                                                     int main () {
#include <vector>
                                                        int runs:
#include <string>
                                                       cin >> runs:
using namespace std:
                                                        while (runs--) {
                                                          cin \gg Y \gg X:
int Y,X,W,B,py,px;
                                                          s = vector<string>(Y);
vector<string> s:
                                                          for (int y=0; y<Y; y++)
vector<vector<bool> > u:
                                                            cin >> s[v]:
void go (int y, int x) {
                                                          bool ok=true;
  if (v<0||v>=Y||x<0||x>=X) return;
                                                          for (px=0; px<2; px++)
                                                            for (py=0; py<2; py++) {
  if (u[v][x]) return:
                                                              u = vector<vector<bool> >
  u[v][x]=true;
                                                                  (Y.vector<bool>(X. false)):
  if (s[v][x]=='.') return:
                                                              for (int y=0; y<Y; y++)
  if (s[y][x]=='W' && (y+py)%2+(x+px)%2!=0)
                                                                for (int x=0; x<X; x++) {
                                                                                                       Problem D
    return:
                                                                  W=B=O:
  if (s[v][x]=:B' \&\& (v+pv)\&2+(x+px)\&2!=1)
                                                                  go(y,x);
    return:
                                                                  if (W!=B) ok=false:
                                                                }
  if (s[v][x]=='W') W++;
                                                            }
  if (s[v][x]=='B') B++:
                                                          cout << (ok ? "YES" : "NO") << endl:
  go(y-1,x);
                                                        3
  go(v+1.x):
  go(y,x-1);
                                                        return 0:
  go(y,x+1);
                                                  International Collegiate
                                                                             ovent
                                            Cm Programming Contest
                                                                             sponsor
```

J - Train delays (1)

#### Calculate best expected time best[x, t] for each station x and time t = 0...59

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event
- J Train delays (1)
  - Calculate best expected time best[x, t] for each station x and time t = 0...59
  - Easiest: use Bellman-Ford algorithm for shortest paths

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- J Train delays (1)
  - Calculate best expected time best[x, t] for each station x and time t = 0...59
  - Easiest: use Bellman-Ford algorithm for shortest paths
  - Initially, best[end, t] = 0 and  $best[other, t] = \infty$

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- J Train delays (1)
  - Calculate best expected time best[x, t] for each station x and time t = 0...59
  - Easiest: use Bellman-Ford algorithm for shortest paths
  - Initially, best[end, t] = 0 and  $best[other, t] = \infty$
  - Loop over all trains, calculate potential new expected time best[from, depart]

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- J Train delays (1)
  - Calculate best expected time best[x, t] for each station x and time t = 0...59
  - Easiest: use Bellman-Ford algorithm for shortest paths
  - Initially, best[end, t] = 0 and  $best[other, t] = \infty$
  - Loop over all trains, calculate potential new expected time best[from, depart]
  - If it's better, update best[from, depart] and best[from, t] for all t

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- J Train delays (1)
  - Calculate best expected time best[x, t] for each station x and time t = 0...59
  - Easiest: use Bellman-Ford algorithm for shortest paths
  - Initially, best[end, t] = 0 and  $best[other, t] = \infty$
  - Loop over all trains, calculate potential new expected time best[from, depart]
  - If it's better, update best[from, depart] and best[from, t] for all t
  - Repeat until nothing changes anymore

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- J Train delays (1)
  - Calculate best expected time best[x, t] for each station x and time t = 0...59
  - Easiest: use Bellman-Ford algorithm for shortest paths
  - Initially, best[end, t] = 0 and  $best[other, t] = \infty$
  - Loop over all trains, calculate potential new expected time best[from, depart]
  - If it's better, update best[from, depart] and best[from, t] for all t
  - Repeat until nothing changes anymore
  - Use epsilon when comparing doubles!

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- J Train delays (2)
  - Calculate best expected time best[x, t] for each station x and time t = 0...59

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event

- J Train delays (2)
  - Calculate best expected time best[x, t] for each station x and time t = 0...59
  - Harder: use Dijkstra's algorithm for shortest paths

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



- J Train delays (2)
  - Calculate best expected time best[x, t] for each station x and time t = 0...59
  - Harder: use Dijkstra's algorithm for shortest paths
  - Issue: sometimes you have to update the same state multiple times

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



J - Train delays (2)

Calculate be	est expected	time	best[x, t]	for	each	station	х
and time t =	= 0 59						

Harder: use Dijkstra's algorithm for shortest paths

#### Issue: sometimes you have to update the same state multiple times

At most 60 times though

#### Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



Solution: maximum flow

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



	Solution:	maximum	flow
--	-----------	---------	------

First, fill all boundary squares

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end



event

<ul> <li>Solution: maximum flow</li> <li>First, fill all boundary squares</li> </ul>	Solutions
<ul> <li>First, ini an boundary squares</li> <li>Constant the following figures</li> </ul>	Statistics
Construct the following flow graph:	Problem E
	Problem B
	Problem C
	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end



Solution: maximum flow	Solutions
First, fill all boundary squares	Statistics
Construct the following flow graph:	Problem E
Vertices: source, sink, every square	Problem B
	Problem C
	Problem A
	Problem H
	Problem G
	Problem I
	Problem D
	Problem J
	Problem F
	The end



Solution: maximum flow	Solutions
<ul> <li>Solution: maximum flow</li> <li>First, fill all boundary squares</li> <li>Construct the following flow graph: <ul> <li>Vertices: source, sink, every square</li> <li>Edge from source to boundary square with capacity ∞</li> </ul> </li> </ul>	Solutions Statistics Problem E Problem B Problem C Problem A Problem H Problem G
	Problem D Problem J Problem F
	The end



Solution: maximum flow	Solutions
<ul> <li>Solution: maximum flow</li> <li>First, fill all boundary squares</li> <li>Construct the following flow graph:         <ul> <li>Vertices: source, sink, every square</li> <li>Edge from source to boundary square with capacity ∞</li> <li>Edge from source to non-boundary grass square with capacity D (dig)</li> </ul> </li> </ul>	Statistics Problem E Problem B Problem C Problem A Problem H Problem G Problem I Problem D Problem J
	Problem F

The end



Solution, maximum flow	Solutions
First, fill all boundary squares	
Construct the following flow graph:	Statistics
Norticos: course sink even square	Problem E
Vertices, source, slink, every square	Problem B
Edge from source to boundary square with capacity $\infty$	Problem C
<ul> <li>Edge from source to non-boundary grass square with capacity D (dig)</li> </ul>	Problem A
Edge from non-boundary hole square to sink with capacity	Problem H
E (fill)	Problem G
7 (III)	Problem I
	Problem D
	Problem J
	Problem F
	The end



event

Solution: maximum flow	Solutions
First, fill all boundary squares	
Construct the following flow graph:	Statistics
<ul> <li>Vertices: source sink every square</li> </ul>	Problem E
Edge from course to boundary square with conseins on	Problem B
Edge from source to boundary square with capacity $\infty$	Problem C
<ul> <li>Edge from source to non-boundary grass square with capacity D (dig)</li> </ul>	Problem A
Edge from non-boundary hole square to sink with capacity	Problem H
F (fill)	Problem G
<ul> <li>Edges between connected squares with capacity B</li> </ul>	Problem I
(boundary)	Problem D
	Problem J
	Problem F

event sponsor

The end



Solution: maximum flow	Solutions
<ul> <li>First, fill all boundary squares</li> </ul>	
<ul> <li>Construct the following flow graph:</li> <li>Vertices: source, sink, every square</li> </ul>	Statistics Problem E
<ul> <li>► Edge from source to boundary square with capacity ∞</li> <li>► Edge from source to non-boundary grass square with</li> </ul>	Problem B Problem C Problem A
<ul> <li>Edge from non-boundary hole square to sink with capacity</li> <li>F (fill)</li> </ul>	Problem H Problem G
<ul> <li>Edges between connected squares with capacity B</li> <li>(boundary)</li> </ul>	Problem I Problem D
You can show that the cost of a cut of this graph equals the cost of splitting it into grass and holes along this cut	Problem J Problem F The end



Solution: maximum flow	Solutions
First, fill all boundary squares	
Construct the following flow graph:	Statistics
Norticos: cource sink even square	Problem E
Vertices. source, sink, every square	Problem B
$\blacktriangleright$ Edge from source to boundary square with capacity $\infty$	Problem C
<ul> <li>Edge from source to non-boundary grass square with</li> </ul>	Problem A
capacity D (dig)	T TODICITI / Y
Edge from non-boundary hole square to sink with capacity	Problem H
F (fill)	Problem G
<ul> <li>Edges between connected squares with capacity B</li> </ul>	Problem I
(boundary)	Problem D
	Problem J
You can show that the cost of a cut of this graph equals	Problem F
the cost of splitting it into grass and holes along this cut	The end
So find the minimum cut, i.e., the maximum flow	

event



The end

Solutions

Statistics

Problem E

Problem B

Problem C

Problem A

Problem H

Problem G

Problem I

Problem D

Problem J

Problem F

The end

# The end



IBM.