V2

Home | Up

I my self have studied several religions in overview and basically each religion is offering its own way to gain salvation from what we know all people will face one day, the death... Some people believe that by doing more good things, they can cover their bad things. But I start to consider, will that be enough? How good should I be?... Can I be sure that my good things will be able to save me when I face death?... Correct me if I am wrong, but according to what I know so far, other religions offer a way that may lead its believer to heaven after death, but only Christianity says for sure that putting faith on the son of God named Jesus Christ earns you a place in heaven, the place where God (the one whom we assume creates the whole universe) lives. In the Christian Bible, there is an interesting verse "For it is by grace you have been saved, through faith and this not from yourselves, it is the gift of God, not by works , so that no one can boast."... Is it really that simple? just believe in Jesus and that's it? Why is that so? To be continued in volume 3. See previous story in volume 1.

Last updated on: 15 October 2007 08:12:38 PM

Comment on this volume: This volume contains many old ACM ICPC world finals problem, with some regional problems at the end. Having a lot of world final contest problems surely make problems in these volume very hard to solve... Moreover, many of them still can't be judge yet, currently only 70 out of 100 problems can be judged.

No	Problem Name	*	Algorithm
200-207: ACM ICPC World Finals - 1989			
200	Rare Order	4.5	Ad Hoc
201	Squares	5.0	Ad Hoc
202	Repeating Decimals	5.5	Math
203	Running Lights Visibility Calculator	*	Haven't try yet
204	Robot Crash	*	Haven't try yet
205	Getting There	*	Haven't try yet
206	Meals on Wheels Routing System	*	Haven't try yet
207	PGA Tour Prize Money	*	Haven't try yet
208-214: ACM ICPC World Finals - 1991			
208	Firetruck	5.0	Backtracking
209	Triangular Vertices	*	Geometry stuffs... Haven't try yet
210	Concurrency Simulator	*	Haven't try yet
211	The Domino Effect	*	Haven't try yet
212	Use of Hospital Facilities	*	Haven't try yet
213	Message Decoding	*	Haven't try yet
214	Code Generation	*	Haven't try yet
215-221: ACM ICPC World Finals - 1992			
215	Spreadsheet Calculator	*	Haven't try yet
216	Getting in Line	4.5	Backtracking
217	Radio Direction Finder	*	Cannot be judged yet!!!
218	Moth Eradication	*	Math (Geometry)-Convex Hull problem...
219	Department of Redundancy Department	*	Cannot be judged yet!!!
220	Othello	6.0	Ad Hoc
221	Urban Elevations	*	Haven't try yet
222-229: ACM ICPC World Finals - 1993			
222	Budget Travel	*	Haven't try yet
223	Classifying Lots in a Subdivision	*	Haven't try yet
224	Kissin' Cousins	*	Cannot be judged yet!!!
225	Golygons	*	Haven't try yet
226	MIDI Preprocessing	*	Haven't try yet
227	Puzzle	5.5	Ad Hoc
228	Resource Allocation	*	Cannot be judged yet!!!
229	Scanner	*	Haven't try yet
230-237: ACM ICPC World Finals - 1994			
230	Borrowers	*	Haven't try yet
231	Testing the CATCHER	5.5	DP (LIS)
232	Crossword Answers	6.0	Ad Hoc
233	Package Pricing	*	Haven't try yet

234	Switching Channels	*	Haven't try yet
235	Typesetting	*	Haven't try yet
236	VTAS - Vessel Traffic Advisory Service	*	Haven't try yet
237	Monitoring Wheelchair Patients	*	Haven't try yet
238-245: ACM ICPC World Finals - 1995			
238	Jill's Bike	*	Haven't try yet
239	Tempus et mobilius. Time and motion	*	Haven't try yet
240	Variable Radix Huffman Encoding	*	Haven't try yet
241	Sail Race	*	Haven't try yet
242	Stamps and Envelope Size	*	Haven't try yet
243	Theseus and the Minotaur (II)	*	Haven't try yet
244	Train Time	*	Haven't try yet
245	Uncompress	*	Haven't try yet
246-252: ACM ICPC World Finals - 1996			
246	10-20-30	*	Haven't try yet
247	Calling Circles	*	Haven't try yet
248	Cutting Corners	*	Haven't try yet
249	Bang the Drum Slowly	*	Haven't try yet
250	Pattern Matching Prelims	*	Haven't try yet
251	Nondeterministic Trellis Automata	*	Haven't try yet
252	Trucking	*	Cannot be judged yet!!!
253-261: Source Unknown			
253	Cube Painting	4.5	Ad Hoc
254	Towers of Hanoi	*	Haven't try yet, look at Arif's notes
255	Correct Move	4.5	Ad Hoc
256	Quirksome Squares	2.5	Math
257	Palinwords	*	Haven't try yet, look at Arif's notes
258	Mirror Maze	*	Haven't try yet
259	Software Allocation	4.5	Graph Traversal
260	Il Gioco dell'X	4.0	Graph (FloodFill)
261	The Window Property	*	Haven't try yet
262-269: East Central Regionals - 1993			
262	Transferable Voting	*	Haven't try yet
263	Number Chains	4.0	Simulation
264	Count on Cantor	4.0	Math
265	Dining Diplomat	*	Haven't try yet
266	Stamping Out Stamps	*	Haven't try yet
267	Of(f) Course!	*	Haven't try yet
268	Double Trouble	*	Haven't try yet
269	Counting Patterns	*	Haven't try yet
270-277: East Central Regionals - 1994 (2nd link)			
270	Lining Up	8.0	Mine is WA, please look at Lego's notes
271	Simply Syntax	6.5	Ad Hoc
272	TEX Quotes	0.5	Ad Hoc
273	Jack Straws	*	Haven't try yet
274	Cat and Mouse	*	Haven't try yet
275	Expanding Fractions	5.0	Math
276	Egyptian Multiplication	6.5	Math
277	Cabinets	*	Haven't try yet
278-284: Southwestern European Regionals - 1993			
278	Chess	4.5	Chess
279	Spin	*	Haven't try yet
280	Vertex	4.0	Graph
281	Rubik's Cube	*	Cannot be judged yet!!!
282	Rename	*	Haven't try yet
283	Compress	*	Haven't try yet
284	Logic	*	Haven't try yet
285-291: Asia Regionals (Shanghai) - 1994			
285	Crosswords	*	Haven't try yet
286	Dead Or Not -- That Is The Question	*	Haven't try yet

287	Text Comparison	*	Cannot be judged yet!!!
288	Arithmetic Operations With Large Integers	*	Haven't try yet
289	A Very Nasty Text Formatter	*	Haven't try yet
290	Palindroms <-> smordnilaP	*	Haven't try yet
291	The House Of Santa Claus	3.5	Backtracking
292-299: Northwestern European Regionals - 1994			
292	Presentation Error	*	Cannot be judged yet!!!
293	Bits	*	Haven't try yet
294	Divisors	6.0	Math
295	Fatman	*	Cannot be judged yet!!!
296	Safebreaker	*	Shouldn't be difficult, Haven't try yet
297	Quadtrees	5.0	Recursion
298	Race Tracks	*	Haven't try yet
299	Train Swapping	1.0	Sorting

Total submit-able problems in this volume: 100
Solved problems: 21
Problems in Wrong Answer list from this volume: 6
Unattempted problems: 63
Total hints in this volume: 27

---

200 - Rare Order (by: Shaka)

To determine the collating sequence:
1. Create a 2 dimension array (10000 rows * 21 chars/row is enough) and a queue.
2. Read all input and store it in your 2 dimension array.
   Btw do you realize that this problem is not a multiple input problem. Only one test case in this problem...
3. Read the first character from your array from the first input until the last and store it in the queue when you encounter a new character which has not been in your queue.
4. Continue reading the 2nd character until the 20th character (the input limit) and just do the same as the first (enqueue it if it is not found in your queue). Don't forget that NOT all inputs consist of 20 characters.
5. After you finished reading all of your stored input, just print the queue.

This algorithm works because the list will imply a complete ordering among those letters that are used. It will obviously sorted by first characters in the list, then by second characters, and so on. :)

201 - Squares

There is no other way to solve this problem other than simulate it. Maximum size of N is 9 anyway. Record all the edges given and then simply use brute force to try all combinations of squares with size 1, size 2, ..., up to size n. Print the number of occurrences appropriately. :)

202 - Repeating Decimals

Try to do fraction division manually, and determine when the cycle repeats. This problem is EXACTLY SIMILAR to problem 275 (Expanding Fractions), only change output format. You can solve 2 problems using one source code (with very minor changes) :-)

208 - Firetruck

This is just a backtracking problem. Given a city map, you must determine all valid routes from fire station (number 1) to a desired street corner. You must do a special check to test whether your truck trapped in cycles and must do so efficiently.

216 - Getting in Line (with help from: Reuber's webpage)

When you see the description, you'll find out that max computer per test case is 8. A total DFS brute force search + a bit pruning is sufficient to solve this problem. First, generate a table of distance from each every computer to other computer (use Phytagoras formula), and then enumerate all possible permutation of links for these computers (max 7 links for 8 computers), save the best permutation so far, prune branches that already exceed the current best. At the end, just print the result with additional 16 feet per link.

227 - Puzzle

This problem involves complex array handling.

Since the input is very complicated, you need to parse it. Put every single character in the first five rows to a 5x5-sized array; remember the coordinate of the empty spot. Then you read all the sequence of moves until you encounter '0'. There are 4 commands:

A: shift the character above the empty position down,

then change the coordinate of the empty spot.
R: shift the character on the right of the empty position left,
then change the coordinate of the empty spot.
B: shift the character below the empty position up,
then change the coordinate of the empty spot.
L: shift the character on the left of the empty position right,
then change the coordinate of the empty spot.

Repeat all the process until you encounter 'Z'.

Common Mistake:
1. The sequence of moves may span more than one line, read the input carefully.
2. If something goes wrong (the coordinate go out from the boundary 1 to 5) then the Puzzle has no final configuration, don't do any more processing.
3. Don't forget to display the puzzle with a space between each character.

231 - Testing the CATCHER

Apply Dynamic Programming to Longest Decreasing Subsequence problem. Click here to see my Dynamic Programming section if you don't familiar with this "Longest Increasing/Decreasing Subsequence" problem.

232 - Crossword Answers (with help from: Yudha Irsandy)

Quite complex array manipulation problem.

239 - Tempus et mobilius. Time and motion (by: Ing Ing)

This is a simple math problem.
1. First, you have to simulate the ball movement, how long does it takes to return to it's initial position (t1 ... tn). (Compute the time required for all n balls)
2. Finally you have to compute the total time, where the total time is the Least Common Multiple (LCM) of all t1 ... tn. You can use associative characteristic of LCM: LCM(a, b, c) = LCM(a, LCM(b, c))

254 - Towers of Hanoi (by: Md. Arifuzzaman)

Binary conversion of m in n bit make this solution easier and interesting than we think.

input: n and m (m input should be in string or char array)
1. convert m into binary number
2. append zero at beginning if necessary to make it n bit binary number
(
example:
5 3
binary number: 00011
)
3. d[0]=d[1]=d[2]=0
4. beg=0, aux=1, dest=2
5. for bit 0 to n-1
if bit=0
d[beg]=d[beg]+1
swap(aux,dest)
else if bit=1
d[dest]=d[dest]+1
swap(aux,beg)
6.if even number of disk
print d[0] d[1] d[2]
else if odd number of disk
print d[0] d[2] d[1]

256 - Quirksome Squares (by: Felix Halim)

This problem is very easy, since the possible input only either 2,4,6, or 8, simply do a brute force calculation for all those 4 possible input.

Here is the answer (yeah, you can send this and get accepted...):

Input:

2
4
6
8

Output:

00
01
81
0000

```
0001
2025
3025
9801
000000
000001
088209
494209
998001
00000000
00000001
04941729
07441984
24502500
25502500
52881984
60481729
99980001
```

257 - Palinwords (by: Md. Arifuzzaman)

Very easy problem

```
1. input in word[257] //search for first one
2. for i=2 to len-1
3.  if word[j-1]=word[j+1]
4.  c=word[j], d=word[j-1]
5.  size=3
  found=1
  break
6. else if word[j]=word[j+1] and word[j-1]=word[j+2]
7.  c=word[j], d=word[j-1]
8.  size=4
  found=1
  break
//if first one found search second one
9. for j=i+1 to len-1
10.  if word[j-1]=word[j+1]
11.   if size=4 or (c!=word[j] || d!=word[j-1])
12.  print YES
  break;
13 else if word[j]=word[j+1] and word[j-1]=word[j+2]
14.  if size=3 or (c!=word[j] || d!=word[j-1])
   print YES
   break;
15. if j=len
16.  print NO
```

259 - Software Allocation

This problem naturally fit as a Constraint Satisfaction Problem (CSP).

Constraint: There are 10 computers, each of them can run 0 to 1 applications (one of the 26 applications or don't run anything).

Your task is to find an assignment such that it satisfy the constraint and the number of applications brought by the user...

To solve this, first reduce the domain of each computer from 27 assignments to the smallest size using domain reduction. For example if application 'A' can only be run in computer 0, then remove all 'A' in computer 1 to 9... Then do backtracking to enumerate all possible assignments in this reduced domain, then check whether this assignment satisfy the user requirement. Done :)

260 - Il Gioco dell'X

When usually in graph traversal, you go to either 4 direction (up,right,down,left) or 8 directions (including diagonals), this time you are given special graph, which cell's have 6 neighbours. You need to check whether from leftmost, there is a way to reach rightmost (which means white wins) or the other way, topmost to bottommost (black wins). You can do simple backtracking, but the easiest way to solve this problem is to do flood fill. flood fill all 'w' starting from leftmost with colour A, and flood fill all 'b' starting from topmost with colour B. Finally check whether there exist a colour A in rightmost column (white wins) or colour B in bottommost row (black wins).

263 - Number Chains

With a tool to sort characters in descending and ascending (qsort), a tool to convert this characters into integer (atoi), and a list (just a short list will do) to memorize the past few numbers generated... You can simply simulate this problem efficiently. No tricks, no traps, just simulate it...

264 - Count on Cantor

A brute force simulation may be possible, but you can solve this problem in a more efficient manner if you can derive the formula. Study the pattern and derive it.

270 - Lining Up (by: Lego Haryanto)

Problem summary: You are given N points (N <= 700), we need to print the maximum number of points that can be passed through by one straight line.

This algorithm is $O(n^2 \lg n)$, AC 1.9xx s in UVa OJ. There may be better algorithm than this.

Solution:
The key idea: angular sorting.

Let the first point from n points be chosen as pivot. With other words, we assume our optimal line will pass through this pivot. Then, sort (n-1) remaining points according to angle with respect to pivot. Then we have order of the points that we can use as reference to count how many collinear points (pseudo code below).

Then, we repeat the abovementioned procedure with the second point as the pivot, and so on until we have tried all points as pivot.

Pseudo code:

```
for (pivot = 0; pivot < n; pivot++) {
  sort other points by angle with respect to pivot. // O (n lg n)

  // O (n) algorithm to determine how many collinear points with this pivot
  lo = 1; hi = 2;
  while (hi < n-1) {
    if (collinear(pivot, lo, hi))
      hi++;
    else {
      maxCount = max(maxCount, hi-lo+1);
      lo = hi++;
    }
  }

  maxCount = max(maxCount, hi-lo+1);
}
```

Thus the complexity is $O(n^2 \lg n)$

Optimization:
This algorithm can be optimized by first sorting the points according to y-coordinates. Then, for ties, sort according to x-coordinates.

Then, proceed as before, but the inner loop don't need to handle (n-1) points but just the "remaining points" starting from the point (pivot+1) in sorted order. That is, the more pivots that have been examined, the number of "remaining points" to be sorted decreases.

We do not need to care about the ignored points as we have taken care of them when we set them as pivot previously.

271 - Simply Syntax (by: Niaz Morshed Chowdhury)

This problem its not that much easy as it looks. For this problem I am giving here an Algorithm, afterwards I will explain it.

```
n = 0 // a variable containing total sentence, initialized as ZERO.
bank[1000] // here the given line will be stored
len = length of [bank]

for i = len - 1 down to 0 {
  if bank[i] == any character between p through z
    n = n+1
  else if bank[i] == any character from C,D,E,I
    if n > = 2
      n = n - 1
    else
      n = 0
      break
  else if bank[i] == character N
    if n < 1
      n = 0
      break
    else
      n = n // no change in 'n'
}
```

After completing the FOR loop.....

```
if n == 1
  Print YES
else
  Print NO
```

Now I am describing how does this algorithm work.

1. Any character from p to z is a correct sentence. So, when we get any of them we just increase the total sentence (n).

2. Two correct sentences and any one of C,D,I,E make a correct sentence. But this time at first our sentence number decrease two.

n = n -2

But with C,D,I,E it makes a new sentence. So,

n = n + 1

So, finally we get,

n = n - 2 + 1

n = n - 1

3. One correct sentence and N make a new sentence. Here also at first total sentence decrease one but then it increase again one. As a result there is no change in 'n'.
4. If we get less than two sentence before C,D,E & I, we just break the loop with assigning 0 at n. Finally it will work as flag. You may notice that we do not break the loop if we get more than two correct sentence. This is because to make with [C...I] we need two sentence and the extras are might be for some other parts. If not then we can track it later. But we can not consider less then two.
5. Same explanation goes for N also.
6. But...finally we must get one and only one correct sentence to tell that its is correct. If we don't get n = 1, then we can surely say that its not a correct sentence.

272 - TEX Quotes

"This is bad quote".
`` This is elegant quote ' '
Simply replace all " to their corresponding opening/closing quote, use flag to determine this.

273 - Jack Straws (by: Sohel Hafiz)

First run a n^2 loop to mark the straws that are directly connected to some other straws. Use adjacency matrix to store this information. Use line intersecting algorithm (see CLRS computational geometry chapter) to find the intersection. Then simply apply Floyd Warshall to see whether a straw is connteced to some other straw (i.e. there is a path from a source straw to a destination straw). Since there is at most 12 straws, Floyd Warshall will pass the time limit

275 - Expanding Fractions

EXACTLY SIMILAR to problem 202 (Repeating Decimals), only change output format. You can solve 2 problems using one source code (with very minor changes) :-)

278 - Chess (with help from: Felix Halim)

From various observation, I found out the following rules:

1. Maximum rooks in an m*n chessboard so they are not in position to take any other rook is minimum (m,n).

   Proof: To make a rook does not attack other rook, each rook must be placed in a different row and different column with other rook. The easiest way to do this is to place these rooks diagonally.

r1			
	r2		
		r3	
			r4

Figure 1. One of the optimal rooks placement.

2. Maximum queens in an m*n chessboard (m>=4 and n>=4) so they are not in position to take any other queen is minimum (m,n).

   Proof: To make a queen does not attack other queen, each queen must be placed in a different row, different column, and different diagonal with other queen. You can do backtracking to do this. However, in problem 278, you are only have to find the maximum number of queens, not their position.

		q1	
q2			
			q3
	q4		

Figure 2. One of the optimal queens placement.

3. Maximum knights in an m*n chessboard so they are not in position to take any other knights is maximum (black tiles,white tiles).

   Proof: A knight on a black tile cannot attack any piece on any other black tiles, and a knight on white tile cannot attack any piece located on any other white tiles. By simply placing all knights in either all white tiles or all black tiles, you can make sure these rooks will not attack each other. To maximize the number of knights, you should choose maximum (black,white) as your answer.

k1		k2	
	k3		k4

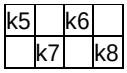

Figure 3. One of the optimal knights placement.

4. Maximum Kings in an m*n chessboard so they are not in position to take any other Kings is (m+1) div 2 * (n+1) div 2

   Proof: A King can reach all directions with length 1. By placing each King like the figure below (separated with length > 1), you can make sure these Kings will not be able to attack each other.

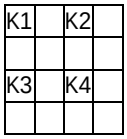

Figure 4. One of the optimal Kings placement.

280 - Vertex

A sample problem to train your Depth First Search skill. The input format is actually an adjacency list, however you can use adjacency matrix if you like. However, please note that N is from 1 to 100 !!!, don't declare 1000 or you'll get Time Limit Exceeded/Crash, and don't declare < 100 or you'll get Wrong Answer...

291 - The House Of Santa Claus

Backtracking will solve this problem.
Tip: Since there are only 44 solutions for this problem, pre-calculate them is a good idea.

294 - Divisors (with help from: Ed Karrels webpage)

An inefficient program will always give you Time Limit Exceeded. Refer to many mathematic websites and verify this formula:

if the number is a^i * b^j * c^k * ...,
then it has (i+1)(j+1)(k+1)... divisors.

297 - Quadtrees

Create an image buffer (i.e. a 2-dimensional Boolean array) of size 32 x 32. Initialize everything to false (white), then according to Quadtrees rule given in problem description, fill the image buffer with true (black) for the first tree and second tree. (Note: the second tree will overwrite any black cell used by first tree, this is what we want). Finally, count how many black cells in the image buffer, and output this value.

299 - Train Swapping

Long problem explanation..., but this problem is simple, just count the Bubble Sort swaps in $O(n^2)$. However, you can solve this problem by counting inversion index using Merge Sort O(n lg n).
