Name: ____________________

May 12, 2009

This is a fake "old" exam for EDAF05. I'm pretty sure the format is right, but maybe I still want to change some things. No guarantees. I haven't decided yet on how many points invidivual exercises should be worth. This is version 1, maybe I'll replace it by a better one later. Also, a sheet with answers will come really soon (I hope).

Instructions

What to bring. You can bring any written aid you want. This includes the course book and a dictionary. In fact, these two things are the only aids that make sense, so I recommend you bring them and only them. But if you want to bring other books, notes, print-out of code, old exams, or today's newspaper you can do so. (It won't help.)

You can't bring electronic aids (such as a laptop) or communication devices (such as a mobile phone). If you really want, you can bring an old-fashioned pocket calculator (not one that solves recurrence relations), but I can't see how that would be of any use to you.

Filling out the exam Most questions are multiple choice. Mark the box or boxes with a cross or a check-mark. If you change your mind, completely black out the box and write your answer's letter(s) in the left margin. In case it's unclear what you mean, I will choose the least favourable interpretation.

In those questions where you have to write or draw something, I will be extremely unco-operative in interpreting your handwriting. So *write clearly*. If there is a way to misunderstand what you mean, I will use it.

Scoring Each multiple choice question has exactly *one* correct answer. To get the maximum score for that question, you must check that answer, and only that. However, to reflect partial knowledge, you *may* check several boxes, in case you're not quite sure (this lowers your score, of course – the more boxes you check, the fewer points you score). If you check *no* boxes or *all* boxes, your score for that question is 0. If the correct answer is not among the boxes you checked, you score is negative, so it's better to *not* answer a question where you're on very thin ice. The worst thing you can do is to check all boxes except the correct one, which gives you a large negative score.

Want an example? Assume a question worth maximum 2 points has $k = 4$ possible answers (one of them correct).

- If you select only the correct answer, you receive 2 points.
- If you select 2 answers, one of which is correct, you receive 1 point.
- If you select 3 answers, one of which is correct, you receive 0.41 points.
- if you select no answer or all answers, you receive 0 point.
- If you select only one answer, and it is wrong, you receive -0.67 points.
- If you select 2 answers, that are both wrong, you receive -1 point.
- If you select 3 answers, that are all wrong, you receive -1.25 points.

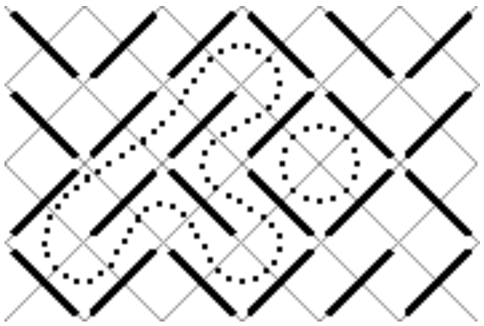
As a special case, for a yes/no question, you receive 1, 0, or -1 points, depending on whether you answer is correct, empty, or wrong.

If you want to know the precise formula, if the question has k choices, and you checked a boxes, your score is $\log(k/a)$, provided you checked the correct answer, and $-a\log(k/a)/(k-a)$ if you only checked wrong answers. Moreover, I have weighted the questions by relevance (not necessarily difficulty), and indicated the maximum points with each question.

You really care why this scoring system makes sense? Then read [Gudmund Skovbjerg Frandsen, Michael I. Schwartzbach: A singular choice for multiple choice. SIGCSE Bulletin 38(4): 34–38 (2006)]. For example, random guessing will give you exactly 0 points, at least in expectation.

Slash maze

By filling a rectangle with slashes (/) and backslashes (\), you can generate nice little mazes. An example is



As you can see, paths in the maze cannot branch, so the whole maze contains only (1) cyclic paths and (2) paths entering somewhere and leaving somewhere else. We are only interested in the cycles. There are exactly two of them in our example. Your task is to write a program that counts the cycles and finds the length of the longest one. The length is defined as the number of small squares the cycle consists of (the ones bordered by gray lines in the picture). In this example, the long cycle has length 16 and the short one length 4.

Input

Each description begins with one line containing two integers M and N ($1 \leq M, N \leq 75$), representing the width and the height of the maze. The next N lines describe the maze itself and contain M characters each; all of these characters will be either "/" or "\".

Output

Output the line "k Cycles; longest has length l.", where k is the number of cycles in the maze and l the length of the longest of the cycles. If the maze is acyclic, output "There are no cycles.".

Example 1

Input:
```
6 4
\//\\/
\///\/
//\\/\
\/\///
```

Output: `2 Cycles; longest has length 16.`

Example 2

Input:
```
3 3
///
\//
\\\
```

Output: `2 Cycles; longest has length 16.`

The great dinner

Each world in the Galactic Federation of Planets has sent a team of representatives that is expected to attend the annual grand banquet arranged by the galactic senate. To maximize the amount of interaction between members of different worlds, no two representatives of the same world will be allowed to sit at the same table.

Given the number of representatives from each world (including senators, ambassadors, clerks, and staff) and the seating capacity of each table, determine whether it is possible for the banquet guests to be seated as described. If such an arrangement is possible, output one such seating assignment. If there are multiple possible arrangements, any one is acceptable.

Input

The first line of each test case contains two integers, $1 \leq M \leq 70$ and $1 \leq N \leq 50$, denoting the number of worlds and tables, respectively. The second line of each test case contains M integers, where the ith integer $m_i \leq 100$ indicates the number of teams from world i. The third line contains N integers, where the jth integer n_j , $2 \leq n_j \leq 100$, indicates the seating capacity of table j.

Output

For each test case, print a line containing either 1 or 0, denoting whether there exists a valid seating arrangement of the representatives. In case of a successful arrangement, print M additional lines where the ith line contains a table number (from 1 to N) for each of the representatives from team i.

Example 1

Input:
```
4 5
4 5 3 5
3 5 2 6 4
```

Output:
```
1 2 4 5
1 2 3 4 5
2 4 5
1 2 3 4 5
```

Example 2

Input:
```
4 5
4 5 3 5
3 5 2 6 3
```

Output:
```
0
```

Hooligans

Bob manages a hotel that is popular among football fans. To prevent the fans from different teams from having a fight in the hotel, each team's supporters are on different floors. Also, each team's supporters always get the same floor, year after year (otherwise they would be confused and go to a different floor by mistake, probably starting a fight.) There are many hundred football teams in total, so even though there is only one match per week, and even though Bob's hotel has many floors, he finds it difficult to schedule. Not all teams meet.

Input

The input starts with N, the number of teams, K, the number of floors in Bob's hotel, M, the number of matches. Then follows a list of team names, and and a list of matches

Output

A floor number for each team, or the word "impossible".

Example 1

Input:
```
4 2 3
Durham
Tottenham
Nottingham
Leeds
Durham - Tottenham
Durham - Leeds
Tottenham - Leeds
```

Output:
```
impossible
```

Example 2

Input:
```
4 2 2
Durham
Tottenham
Nottingham
Leeds
Durham - Tottenham
Durham - Leeds
```

Output:
```
Durham: 1st floor
Tottenham: 2nd floor
Leeds: 2nd floor
```

Turtle stacking

A priest of the great turtle god Yertle has enlisted your advice as to the order in which turtles should be stacked without cracking. Each of the 5607 turtles ordered by the priest has a different weight and strength. Your task is to build the largest stack of turtles possible in honour of Yertle.

Input

Standard input consists of several lines, each containing a pair of integers separated by one or more space characters, specifying the weight and strength of a turtle. The weight of the turtle is in grams. The strength, also in grams, is the turtle's overall carrying capacity, including its own weight. That is, a turtle weighing 300g with a strength of 1000g can carry 700g of turtles on its back. There are *N* turtles, at most 5607.

Output

Your output is a single integer indicating the maximum number of turtles that can be stacked without exceeding the strength of any one.

Input:
```
300  1000
1000  1200
200  600
100  101
```

Output:
```
3
```

Analysis of algorithms

1. True of false?

(a) $n(n+1)/2 = O(n^3)$

$\boxed{\text{A}}$ true $\boxed{\text{B}}$ false

(b) $n(n+1)/2 = O(n^2)$

$\boxed{\text{A}}$ true $\boxed{\text{B}}$ false

(c) $n(n+1)/2 = O(n \log n)$

$\boxed{\text{A}}$ true $\boxed{\text{B}}$ false

2.

(a) Suppose you have an algorithm with running time exactly $10n^2$. How much slower does the algorithm get when you double the input size?

$\boxed{\text{A}}$ twice as slow $\boxed{\text{B}}$ 4 times slower $\boxed{\text{C}}$ 10 times slower
$\boxed{\text{D}}$ 20 times slower $\boxed{\text{E}}$ 40 times slower

3. Consider the following piece of code:

```
1: for $i = 1$ to $n$
2:     for $j = 1$ to $n$
3:         $k = k + i + j$
```

(a) What is the running time? (Choose the smallest correct estimate.)

$\boxed{\text{A}}$ $O(\log n)$ $\boxed{\text{B}}$ $O(\sqrt{n} \log n)$ $\boxed{\text{C}}$ $O(n)$ $\boxed{\text{D}}$ $O(n \log n)$ $\boxed{\text{E}}$ $O(n^{3/2})$
$\boxed{\text{F}}$ $O(n^2)$ $\boxed{\text{G}}$ $O(n^3)$ $\boxed{\text{H}}$ $O(n!)$ $\boxed{\text{I}}$ $O(2^n \log n)$

(b) Assume I changed line 2 to "**for** $j = i$ **to** n". Then the running time is

$\boxed{\text{A}}$ asympototically faster

$\boxed{\text{B}}$ asymptotically slower

$\boxed{\text{C}}$ asymptotically the same

4. Consider the following piece of code:

```
1: int $f$(int $n$) {
2:     if $n > 9$ then return $f(n/3) + f(n/3 - 1) + f(n/3 - 2)$
3:     else return 3 }
```

(a) Which recurrence relation best characterises the running time of this method?

$\boxed{\text{A}}$ $T(n) = T(n) + T(n-1) + T(n-2) + O(1)$

$\boxed{\text{B}}$ $T(n) = 2T(n/2) + O(\log n)$

$\boxed{\text{C}}$ $T(n) = 3T(n) + T(n/3)$

$\boxed{\text{D}}$ $T(n) = 3T(n/3) + O(1)$

$\boxed{\text{E}}$ $T(n) = 2T(n/3) + O(1)$

$\boxed{\text{F}}$ $T(n) = 3T(n) + T(n/3)$

(b) Find the solution to $T(n) = n + T(n-1)$, $T(0) = 0$.

$\boxed{A}$ $O(\log\log n)$ $\boxed{B}$ $O(\sqrt{n}\log n)$ $\boxed{C}$ $O(n)$ $\boxed{D}$ $O(n\log n)$ $\boxed{E}$ $O(n^{3/2})$
$\boxed{F}$ $O(n^2)$ $\boxed{G}$ $O(n^3)$ $\boxed{H}$ $O(n!)$ $\boxed{I}$ $O(2^n\log n)$

(c) (Harder) Find the solution to $T(n) = n + T(\sqrt{n})$, $T(0) = 0$.

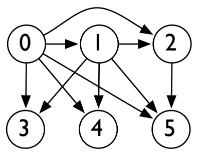
$\boxed{A}$ $O(\log\log n)$ $\boxed{B}$ $O(\sqrt{n}\log n)$ $\boxed{C}$ $O(n)$ $\boxed{D}$ $O(n\log n)$ $\boxed{E}$ $O(n^{3/2})$
$\boxed{F}$ $O(n^2)$ $\boxed{G}$ $O(n^3)$ $\boxed{H}$ $O(n!)$ $\boxed{I}$ $O(2^n\log n)$

(d) Find the solution to $T(n) = n + 2T(n/3)$, $T(0) = 0$.

$\boxed{A}$ $O(\log\log n)$ $\boxed{B}$ $O(\sqrt{n}\log n)$ $\boxed{C}$ $O(n)$ $\boxed{D}$ $O(n\log n)$ $\boxed{E}$ $O(n^{3/2})$
$\boxed{F}$ $O(n^2)$ $\boxed{G}$ $O(n^3)$ $\boxed{H}$ $O(n!)$ $\boxed{I}$ $O(2^n\log n)$

Graphs

5. Consider the graph $G = (V, E)$ with $V = \{0,1,2,3,4,5\}$ that looks like this:



(a) The edge set E is the number of pairs (u, v) such that

$\boxed{A}$ $u < v$ $\boxed{B}$ $u \le v$ $\boxed{C}$ u is even and v is odd
$\boxed{D}$ $u = v \pmod 6$ $\boxed{E}$ $u^2 < v$ $\boxed{F}$ $uv \le 4$

(b) What is $|V|$?

$\boxed{A}$ 5 $\boxed{B}$ 6 $\boxed{C}$ 10 $\boxed{D}$ 16

(c) What is $|E|$?

$\boxed{A}$ 5 $\boxed{B}$ 6 $\boxed{C}$ 10 $\boxed{D}$ 16

Graph traversal

6. The following problem can be modelled as a graph traversal problem:

(a) That problem is

$\boxed{A}$ Slash maze $\boxed{B}$ Hooligans $\boxed{C}$ Turtle stacking $\boxed{D}$ The great dinner

(b) For example, the graph corresponding to the smaller example input to that problem looks like this: (draw a graph on the back of this page).

(c) In general, if the original instance has size n then the resulting graph has

$\boxed{A}$ $|V| = O(n)$ $\boxed{B}$ $|V| = O(n\log n)$ $\boxed{C}$ $|V| = O(n^2)$ $\boxed{D}$ $|V| = O(2^n)$

(d) and

$\boxed{A}$ $|E| = O(n)$ $\boxed{B}$ $|E| = O(n\log n)$ $\boxed{C}$ $|E| = O(n^2)$ $\boxed{D}$ $|E| = O(2^n)$

(e) In the resulting graph, we can solve the original problem by

$\boxed{A}$ starting a BFS from every vertex

$\boxed{B}$ counting the number of edges

$\boxed{C}$ finding the largest connected component

D finding a shortest path

(f) The running time is, asymptotically linear in

A the number of edges

B the number of cyles

C the size of the largest connected component

D the length of the shortest path or cycle

Dynamic programming

One of the problems in the set is a clean example of dynamic programming.

7.

(a) The dynamic programming problem is

A Slash maze B Hooligans C Turtle stacking D The grand dinner

(b) The recurrence relation is:

A

$$\mathrm{OPT}(i,k) = \begin{cases} \min\{\mathrm{OPT}(i-1,k), \mathrm{OPT}(i-1,k-1)+f(i)\}\,, & \text{if } \mathrm{OPT}(i-1,k-1)+f(i) \leq g(i)\,, \\ \mathrm{OPT}(i-1,k)\,, & \text{otherwise}\,. \end{cases}$$

B

$$\mathrm{OPT}(i) = \max\{\mathrm{OPT}(f(i)) + g(i), \mathrm{OPT}(i-1)\}$$

C

$$\mathrm{OPT}(i,k) = \max(\mathrm{OPT}(i-1, g(i)), f(i) + \mathrm{OPT}(i-1, f(i)-g(i)))$$

(c) where $f(i)$ is the

A weight of the ith heaviest turtle

B strength of the ith strongest turtle

C number of matches of the ith team

D number of teams on the ith floor

E size of the ith largest table

F size of the ith largest contingent

G length of the ith longest cycle

(d) and $g(i)$ is the

A weight of the ith strongest turtle

B strength of the ith heaviest turtle

C number of matches of the ith team

D number of teams on the ith floor

E size of the ith table

F size of the ith contingent

G length of the ith cycle

(e) The resulting running time is

A $O(N)$ B $O(N \log N)$ C $O(N^2)$ D $O(2^N)$

Network flow

One of the problems in the set is easily solved by a reduction to network flow.

8.

(a) That problem is

A Slash maze B Hooligans C Turtle stacking D The grand dinner

(b) The network contains a node for

A each team and each table B each team C each table
D each turtle E each turtle size F each turtle weight
G each floor H each team and each floor I each forward slash
J each maze position

(c) The total number of nodes (including the soure and the sink), in terms of the parameters of the original problem, is

A $M+N+2$ B $M+N$ C 2 D M
E N F $O(1)$ G MN H $MN\log N$

(d) On the back of this page, describe how the nodes are connected, and what the capacities are. Do this in general (use words like "every node corresponding to a giraffe is connected to every node corresponding to a letter by an undirected arc of capacity the length of the neck"), and also draw a small, but complete example for one of the example instances.

Computational complexity

These questions are about *Hooligans*, and include questions about NP. Don't take this as an indication that Hooligans may or may not be NP-hard.

9. *(Decision version.)* The decision version of Hooligans requires as *input* as a minimum

(a) a list of matches of the form

A true B false

(b) an assignment of teams to floor numbers

A true B false

(c) the number k of floors in Bob's hotel

A true B false

(d) The decision version expects as output

A "yes," if the given teams can be arranged on k floors, and "no" otherwise

B a single integer, which is the smallest number of floors needed to arrange the given number of teams

C a list of pairs giving the floor number assigned to each team, or "impossible"

D a single integer, which is a number of floors sufficient to arrange the given number of teams

10. *Membership in NP.* The decision version of Hooligans is in NP.

(a) The certificate includes, (apart from the input already listed in answer above,) the following information

A 'yes," if the given teams can be arranged in the given number of floors, and "no" otherwise

B a single integer, which is the smallest number of floors needed to arrange the given number of teams

C a single integer, which is a number of floors sufficient to arrange the given number of teams

D a list of pairs giving the floor number assigned to each time, or "impossible"

(b) The certificate has length

A $O(N)$ B $O(N+M)$ C $O(N^2)$ D $O(M^2)$

(c) The certificate can be checked in time (choose the smallest possible)

A $O(N)$ B $O(N+M)$ C $O(N^2)$ D $O(M^2)$

NP-hardness

One of the problems in the set is NP-hard.[1]

11.

(a) The following problem (called P_1) is NP-hard:

A Slash maze B Hooligans C Turtle stacking D The grand dinner

(b) The easiest way to see that is to take the following NP-hard problem, called P_2,

A Graph colouring B Set cover C Vertex cover D 3-Sat

(c) and prove

A $P_1 \leq_P P_2$ B $P_2 \leq_P P_1$

(d) For this, an arbitrary instance of

A Slash maze B Hooligans C Turtle stacking D The grand dinner
E Graph colouring F Set cover G Vertex cover H 3-Sat

(e) is transformed into an instance of

A Slash maze B Hooligans C Turtle stacking D The grand dinner
E Graph colouring F Set cover G Vertex cover H 3-Sat

(f) Describe, on the back of this page, both in general and for a small but complete example, the transformed instance. In particular, be clear about the paramters of the instance you produce (for example number of vertices, edges, sets, colors, what the path is, etc.))

[1]If P = NP then *all* these problems are NP-hard. Thus, in order to avoid unproductive (but hypothetically correct) answers this section assumes that P $\neq$ NP.