EDAF05 Exam

2 juni 2009, 14.00–19.00

Thore Husfeldt, Computer Science, Lund University

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*. Use English or Swedish. 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.

Algorithmic Problems

Camelot

King Arthur expects N knights for an annual dinner at Camelot, including himself. Unfortunately, some of the knights quarrel with each other. Thanks to his wizard Merlin's network of spies, Arthur knows who quarrels with whom. Arthur wants to seat his guests around a huge, round table so that no two quarrelling knights sit next to each other. Arthur is a beloved king, so he quarrels with nobody.

Input

The input consists of the integer N on a single line, followed by a list of $N-1$ names, one on each line, followed by a list of quarrels in the form of pairs of integers $x\ y$ ($1 \leq x < y < N$). The invited knights are indexed $1, \ldots, N-1$, Arthur himself has index N. Quarrels are mutual and only listed once.

Output

A seating order around the table, starting with Arthur, so that no knight quarrels with his neighbours. If no such order exists, the algorithm outputs the word "impossible".

Example

Example 1

Input

```
150
Gawain
Lancelot
... (146 more names)
Parceval
1 2
2 16
29 145
... (many more lines)
137 149
```

Output

```
Arthur
Parceval
... (147 more names)
Robert
```

Example 2

Input

```
4
Gawain
Lancelot
Parceval
1 2
2 3
```

Output

```
impossible
```

Ferries

Before bridges were common, ferries were used to transport cars across rivers. River ferries, unlike their larger cousins, run on a guide line and are powered by the river's current. Cars drive onto the ferry from one end, the ferry crosses the river, and the cars exit from the other end of the ferry.

There is a ferry across the river that can take M cars across the river in t minutes and return in t minutes. $N > M$ cars arrive at the ferry terminal by a given schedule. (The schedule is known in advance.) What is the earliest time that all the cars can be transported across the river? What is the minimum number of trips that the operator must make to deliver all cars by that time?

Input

Input begins with M, t, N in the first line. N lines follow, each giving the arrival time for a car (in minutes since the beginning of the day). The operator can run the ferry whenever he or she wishes, but can take only the cars that have arrived up to that time.

Output

For each test case, output a single line with two integers: the time, in minutes since the beginning of the day, when the last car is delivered to the other side of the river, and the minimum number of trips made by the ferry to carry the cars within that time. You may assume that $0 < N, t, M < 1440$. The arrival times for each test case are in non-decreasing order.

Examples

Make sure you understand these. In particular, think about why the answer to example 2 is "50 2" and not "60 2".

Example 1

Input

```
2 10 10
0
10
20
30
40
50
60
70
80
90
```

Output

100 5

Example 2

Input

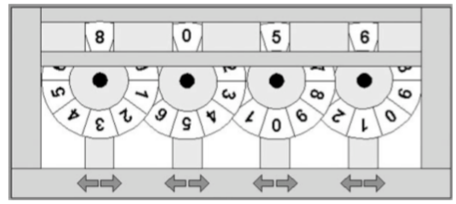
```
2 10 3
10
30
40
```

Output

50 2

Wheels

Consider the machine to the right. Digits ranging from 0 to 9 are printed consecutively (clockwise) on the periphery of each wheel. The topmost digits of the wheels form a four-digit integer. For example, in the following figure the wheels form the integer 8056. Each wheel has two buttons associated with it. Pressing the button marked with a left arrow rotates the wheel one digit in the clockwise direction and pressing the one marked with the right arrow rotates it by one digit in the opposite direction.



We start with an initial configuration of the wheels, with the topmost digits forming the integer $S_1S_2S_3S_4$. You will be given a set of N forbidden configurations $F_{i,1}F_{i,2}F_{i,3}F_{i,4}$ $(1 \leq i \leq N)$ and a target configuration $T_1T_2T_3T_4$. Your job is to write a program to calculate the minimum number of button presses required to transform the initial configuration to the target configuration without passing through a forbidden one.

Input

The first line of each test case contains the initial configuration of the wheels, specified by four digits. Two consecutive digits are separated by a space. The next line contains the target configuration. The third line contains an integer N giving the number of forbidden configurations. Each of the following N lines contains a forbidden configuration.

Output

Print a line containing the minimum number of button presses required. If the target configuration is not reachable print "-1".

Examples

Example 1

Input

```
8 0 5 6
6 5 0 8
5
8 0 5 7
8 0 4 7
5 5 0 8
7 5 0 8
6 4 0 8
```

Output
14

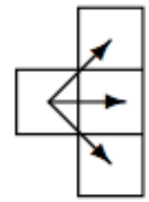
Example 2

Input

```
0 0 0 0
5 3 1 7
8
0 0 0 1
0 0 0 9
0 0 1 0
0 0 9 0
0 1 0 0
0 9 0 0
1 0 0 0
9 0 0 0
```

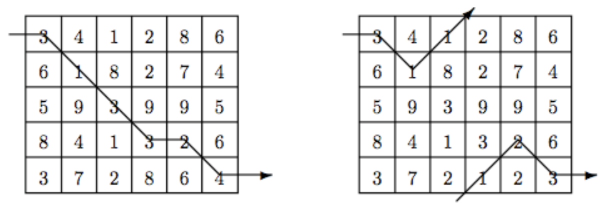
Output
-1

Cylinder



Given an $M \times N$ matrix of integers (including negative numbers and zero), we want to compute a path of minimal weight from left to right across the matrix. A path starts anywhere in column 1 and consists of a sequence of steps terminating in column N. Each step consists of traveling from column i to column $i+1$ in an adjacent (horizontal or diagonal) row, as shown to the right. The first and last rows (rows 1 and M) of a matrix are considered adjacent; i.e., the matrix "wraps" so that it represents a horizontal cylinder.

The weight of a path is the sum of the integers in each of the N cells of the matrix that are visited. The minimum paths through two slightly different 5×6 matrices are



The matrix values differ only in the bottom row. The path for the matrix on the right takes advantage of the adjacency between the first and last rows.

Input

The input consists of a sequence of matrix specifications. Each matrix consists of the row and column dimensions on a line, denoted M and N, respectively. This is followed by MN integers, appearing in row major order; i.e., the first N integers constitute the first row of the matrix, the second N integers constitute the second row, and so on. The integers on a line will be separated from other integers by one or more spaces. Note: integers are not restricted to being positive. No path's weight will exceed integer values representable using 30 bits.

In case you missed it: the integers can be negative, such as in Example 3 below.

Output

The cost of a minimum-weight path.

Examples

Example 1

Input

```
5 6
3 4 1 2 8 6
6 1 8 2 7 4
5 9 3 9 9 5
8 4 1 3 2 6
3 7 2 8 6 4
```

Output

16

Example 2

Input

```
5 6
3 4 1 2 8 6
6 1 8 2 7 4
5 9 3 9 9 5
8 4 1 3 2 6
3 7 2 1 2 3
```

Output

11

Example 3

Input

```
2 2
-1 10
9 -1
```

Output

-2

Gophers

The are N gophers and M gopher holes, each at distinct (x, y) coordinates. A hawk arrives and if a gopher does not reach a hole in s seconds it is vulnerable to being eaten and considered not safe. A hole can save at most one gopher. All the gophers run at the same velocity v. The gopher family needs an escape strategy that minimizes the number of vulnerable gophers.

The input contains several cases. The first line of each case contains four positive integers less than 100: N, M, s, and v. The next N lines give the coordinates of the gophers; the following M lines give the coordinates of the gopher holes. All distances are in metres; all times are in seconds; all velocities are in metres per second.

Output consists of a single line for each case, giving the number of safe gophers.

Input
```
2 2 5 10
1.0 1.0
2.0 2.0
100.0 100.0
20.0 20.0
```

Output
```
1
```

Exam Questions

Analysis of algorithms

1. Let $f(n) = (n + 2\log n + \frac{1}{n})n\log n$. True of false?

(a) *(1 pt.)* $f(n) = O(n^3)$

[A ✔] true [B] false

(b) *(1 pt.)* $f(n) = O(n^2 \log n)$

[A ✔] true [B] false

(c) *(1 pt.)* $f(n) = O(n \log n)$

[A] true [B ✔] false

2. Consider the following piece of code:

1: **for** $i = 1$ **to** n
2: **for** $j = 1$ **to** n
3: **for** $k = j + 1$ **to** $j + 3$
4: **print** k;

(a) *(2 pt.)* What is the running time? (Choose the smallest correct estimate.) Assume **print** takes constant time.

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

(b) *(1 pt.)* Assume I changed line 4 to "**dostuff** (k, n);", where **dostuff**(k, n) takes time $O(k \log n)$. Then the running time for the whole algorithm is asymptotically

[A] faster [B ✔] slower [C] the same

(c) *(2 pt.)* Assume I changed line 3 and 4 to

3: **if** $(i == j)$ **then**
4: **for** $k = 1$ **to** n **print** k;
5: **else** print n;

Then the running time for the whole algorithm is: (Choose the smallest correct estimate.)

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

3. Consider the following piece of code:

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

(a) *(3 pt.)* Which recurrence relation best characterises the running time of this method?

[A ✔] $T(n) = T(n-1) + O(1)$ [B] $T(n) = 2T(n/2) + O(\log n)$
[C] $T(n) = T(n-1) + T(n/2)$ [D] $T(n) = T(n-1) + O(n)$
[E] $T(n) = 2T(n-1) + O(1)$ [F] $T(n) = 2T(n/2) + O(\log n)$

(b) (*1 pt.*) Find the solution to $T(n) = 1 + T(n-1)$, $T(0) = 0$. Give the smallest correct answer.

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

(c) (*1 pt.*) (Harder) Find the solution to $T(n) = 1 + T(\sqrt{n})$, $T(0) = 0$. Give the smallest correct answer.

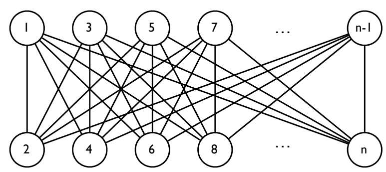
[✓] $O(\log \log n)$ [B] $O(\sqrt{n} \log n)$ [C] $O(n)$ [D] $O(n \log n)$ [E] $O(n^{3/2})$
[F] $O(n^2)$ [G] $O(n^2 \log n)$ [H] $O(n!)$ [I] $O(2^n \log n)$

(d) (*1 pt.*) Find the solution to $T(n) = n^2 + 4T(n/2)$, $T(0) = 0$. Give the smallest correct answer.

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

Graphs

4. Consider the n-vertex graph $G_n = (V_n, E_n)$ that looks like this:



(a) There is an edge between u and v if

[A] $u < v$ [B] $u \leq v$ [✓] $u + v$ is odd
[D] $u = v \pmod 6$ [E] $u^2 < v$ [F] $uv \leq 4$

(b) (*1 pt.*) G_n is bipartite

[✓] True [B] False

(c) (*1 pt.*) G_n is connected

[A] rue [B] False

(d) (*1 pt.*) G_n is directed

[A] True [✓] False

(e) (*1 pt.*) G_n contains a Hamiltonian cycle

[✓] True [B] False

(f) (*1 pt.*) Each vertex in G_n has degree

[A] $\log n$ [B] $\sqrt{n}$ [✓] $n/2$ [D] n [E] $2n$

(g) (*1 pt.*) A maximum independent set in G_n has size

[A] $\log n$ [B] $\sqrt{n}$ [✓] $n/2$ [D] n [E] $2n$

(h) (*1 pt.*) $|E_n| =$

[✓] $n^2/4$ [B] $n^2/2$ [C] n^2 [D] $2n^2$
[E] $\binom{n}{2}/4$ [F] $\binom{n}{2}/2$ [G] $\binom{n}{2}$ [H] $2\binom{n}{2}$

Unweighted graph traversal

5. The following problem can be modelled as a traversal problem in an unweighted (!) graph.

(a) (*3 pt.*) That problem is

[A] Camelot [B] Ferry [✓] Wheels [D] Cylinder [E] Gophers

(b) (*1 pt.*) The number of vertices in the corresponding graph is at most

[A] N [B] M [✓] 10000 [D] NM
[E] 4 [F] $N+M$ [G] $N \log M$ [H] $M \log N$

(c) (*1 pt.*) The number of neighbours of each vertex is at most

[A] N [B] M [C] 10 [✓] 8

(d) (*1 pt.*) The resulting graph is

[✓] undirected [B] directed

(e) (*3 pt.*) In the resulting graph, we can solve the original problem by

[A] starting a DFS from every vertex

[✓] finding the shortest path between two specific vertices

[C] finding the largest connected component

[D] counting the number of odd cycles

[E] finding a topological ordering of the vertices

[F] running Dijkstra from all vertices at the left

(f) (*1 pt.*) The running time is $O(N^2)$. (Be careful about what N is!)

[A] true [✓] false

Greedy

6.

(a) (*3 pt.*) The following problem admits a greedy algorithm:

[A] Camelot [✓] Ferry [C] Wheels [D] Cylinder [E] Gophers

(b) (*2 pt.*) Here's how it works:

[A] Let $K_1, \ldots, K_N$ be the knights sorted by how many quarrels they have,

[B] Let $K_1, \ldots, K_N$ be the knights sorted alphabetically,

[✓] Let $C_1, \ldots, C_N$ be the cars sorted by arrival time,

[D] Let $C_1, \ldots, C_N$ be the cars sorted by the time it has to wait before the next car arrives behind it,

[E] Let $P_1, \ldots, P_N$ be the forbidden positions sorted by their numerical distance to S,

[F] Let $C_1, \ldots, C_N$ be columns sorted from left to right,

[G] Let $G_1, \ldots, G_N$ denote the gophers sorted by distance to nearest hole,

[H] Let $G_1, \ldots, G_N$ denote the gophers sorted by how many other gophers have the same hole as their nearest hole,

[I] Let $G_1, \ldots, G_N$ denote the gophers sorted by the number of other gophers within v^2/s metres,

(c) (*3 pt.*)

[A] in ascending order. [✓] in descending order.

(d) (*1 pt.*)

- [A] Let the ith knight be the first knight among $K_i, \ldots, K_N$ that does not quarrel with the $(i-1)$st knight.
- [B ✔] For $i = 1, \ldots, \lfloor N/M \rfloor$ a full ferry transports the cars $C_{(i-1)M+1}, \ldots, C_{iM}$. A single non-full ferry leaves with $C_{\lfloor N/M \rfloor M+1}, \ldots, C_M$ as soon as they have arrived.
- [C] Start at the smallest entry in C_1. In the ith step, go to the smallest of the three positions available from the $(i-1)$st position.
- [D] For $i = 1, \ldots, n$, let G_i occupy to the nearest hole that is not yet occupied by $G_1, \ldots, G_{i-1}$.

(e) (*1 pt.*) The running time is (give the smallest possible)

[A ✔] $O(N)$. [B] $O(N \log N)$.

Dynamic programming

7.

(a) (*3 pt.*) The following problem admits a straightforward dynamic programming solution:[1]

[A] Camelot [B] Wheels [C ✔] Cylinder [D] Gophers

(b) (*3 pt.*) Following the book's notation, we let $\mathrm{OPT}(i, j)$ denote the value of a partial solution. Then OPT satisfies[2]

[A]

$$\mathrm{OPT}(i, j) = \min\{\mathrm{OPT}(i-1, j), \mathrm{OPT}(i-1, j-1) + f(i, j)\}$$

[B ✔]

$$\mathrm{OPT}(i, j) = f(i, j) + \min_{j-1 \le k \le j+1} \{\, \mathrm{OPT}(i-1, k) \,\}$$

[C]

$$\mathrm{OPT}(i, j) = \min_{j \le k \le N} \{\, \mathrm{OPT}(i-1, k) + f(i, k) \,\}$$

[D]

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

[E]

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

[F]

$$\mathrm{OPT}(i, j) = \min_{j-1 \le k \le j+1} \{\, \mathrm{OPT}(i-1, k) + f(i-1, k) \,\}$$

(c) (*2 pt.*) where $f(i, j)$ is

- [A] 1 if knight i quarrels with knight j and 0 otherwise
- [B] the distance from position i to position j

---

[1] Yes, Ferries is missing from the list of answers. That is deliberate. I know that there is a dynamic programming solution to Ferries as well, but I don't want to hear it.

[2] There may be other cases, in particular, boundary conditions such as maybe for $\mathrm{OPT}(1, 1)$ or $\mathrm{OPT}(N, 0)$. Don't worry about them. This question is just about the most central part of the recurrence relation, otherwise this exercise becomes too large.

- [C] the number of turns needed to turn position i into position j
- [✔] the value in column i, row j
- [E] the value in row i, column j
- [F] the distance between the ith gopher and the jth hole
- [G] the number of seconds needed for the ith gopher to get to the jth hole

(d) (*1 pt.*) The resulting running time is

[A] $O(N)$ [B] $O(N \log N)$ [C] $O(N^2)$ [D] $O(N^2 \log N)$
[E] $O(M)$ [F] $O(N \log M)$ [✔] $O(NM)$ [H] $O(NM^2)$

Network flow

8.

(a) (*3 pt.*) One of the problems in the set is easily solved by a reduction to network flow. That problem is

[A] Camelot [B] Ferry [C] Wheels [D] Cylinder [✔] Gophers

(b) (*2 pt.*) The network consists of a node for[3]

[A] each knight [B] each knight and each table position
[C] each car [D] each car and each "empty slot" on the ferry
[E] each configuration [F] each non-forbidden configuration
[G] each matrix entry [H] each gopher
[I] each hole [✔] each gopher and each hole

(c) (*1 pt.*) The total number of nodes (including the soure and the sink), in terms of the parameters of the original problem, is

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

(d) (*3 pt.*) Describe the reduction on the back of this page, or on a separate piece of paper. Be ridiculously precise about how the nodes are connected and directed, 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. What does a maximum flow mean in terms of the original problem, and what size does it have in terms of the original parameters?

Computational complexity

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

9. *(Decision version.)* The input to the decision version of Gophers is

(a) (*1 pt.*) a list of gopher and hole positions

[✔] true [B] false

(b) (*1 pt.*) an assignment of some gophers to holes

[A] true [✔] false

[3] Apart from these, there may be a *constant* number of extra nodes, such as a source and a sink, or a single node representing the table, or the hawk, or the two banks of the river, etc.

(c) (*1 pt.*) an integer k

- [x] A true
- [ ] B false

(d) (*1 pt.*) The output to the decision version of Gophers is

- [x] A "yes," if at least k gophers are safe
- [ ] B "yes," if there are more holes than gophers
- [ ] C a single integer, which is the number of gophers that can be saved
- [ ] D a list of pairs giving the hole assigned to each gopher
- [ ] E "yes" if the hawk can eat k gophers

10. *Membership in NP.* The decision version of Gophers is easily seen to be in NP, because it admits a certificate.

(a) (*2 pt.*) The certificate includes, (apart from the input already listed in answer above,) the following information

- [ ] A "yes," if k gophers can be rescued
- [ ] B a single integer, which is the largest number of gophers that can be saved
- [ ] C "yes," if the hawk can eat k gophers
- [ ] D an ordering of the k gophers such that the hawk can kill them all before they disappear. (Assume the hawk can kill a gopher in 0 seconds once he reaches it.)
- [x] E a list of pairs giving the hole assigned to each gopher
- [ ] F a list of the occupied holes

(b) (*2 pt.*) The certificate has length (choose the smallest possible)

- [x] A $O(N)$
- [ ] B $O(N+M)$
- [ ] C $O(N^2)$
- [ ] D $O(M^2)$

(c) (*2 pt.*) The certificate can be checked in time (choose the smallest possible)

- [x] 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.[4]

11.

(a) (*3 pt.*) The following problem (called P_1) is NP-hard:

- [x] A Camelot
- [ ] B Ferry
- [ ] C Wheels
- [ ] D Cylinder
- [ ] E Gophers

(b) (*3 pt.*) The easiest way to see that is to take the following NP-hard problem, called P_2,

- [ ] A Graph colouring
- [ ] B 3-dim. matching
- [ ] C Independent set
- [ ] D Set packing
- [ ] E Vertex cover
- [ ] F 3-satisfiability
- [ ] G Travelling salesman
- [x] H Hamiltonian cycle

(c) (*2 pt.*) and prove

- [ ] A $P_1 \leq_P P_2$
- [x] B $P_2 \leq_P P_1$

(d) (*1 pt.*) For this, an arbitrary instance of

- [ ] A Camelot
- [ ] B Ferry
- [ ] C Wheels
- [ ] D Cylinder
- [ ] E Gophers
- [ ] F Graph colouring
- [ ] G 3-dim. matching
- [ ] H Independent set
- [ ] I Set packing
- [ ] J Vertex cover
- [ ] K 3-satisfiability
- [ ] L Travelling salesman
- [x] M Hamiltonian cycle

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

(e) (*1 pt.*) is transformed into an instance of

- ☑ A Camelot
- ☐ B Ferry
- ☐ C Wheels
- ☐ D Cylinder
- ☐ E Gophers
- ☐ F Graph colouring
- ☐ G 3-dim. matching
- ☐ H Independent set
- ☐ I Set packing
- ☐ J Vertex cover
- ☐ K 3-satisfiability
- ☐ L Travelling salesman
- ☐ M Hamiltonian cycle

(f) (*4 pt.*) Describe the reduction on the back of this page, or on a separate piece of paper. Do this both in general and for a small but complete example. In particular, be ridiculously precise about the paramters of the instance you produce (for example number of vertices, edges, sets, colors) in terms of the parameters of the original instance, what the solution of hte transformed instance means in terms of the original instance, etc.