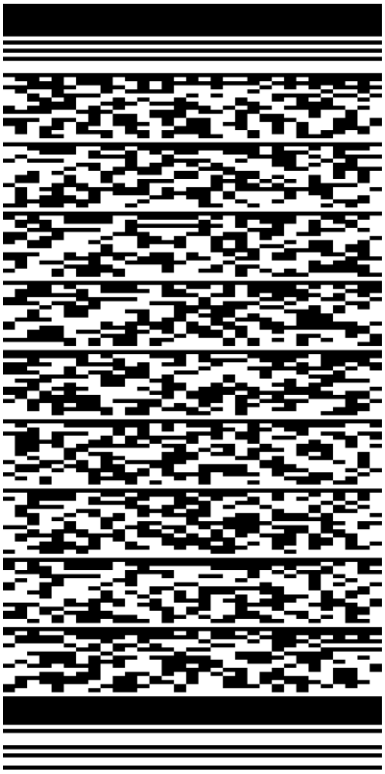
Day 2 Task 4

Xedef



The Xedef Courier Company provides air package delivery among several cities. Some of these cities are Xedef *hubs* where special processing facilities are established. Each of Xedef's aircraft shuttles back and forth between one pair of cities, carrying packages in either direction as required.

To be shipped from one city to another, a package must be transported by a sequence of hops, where each hop carries the package between a pair of cities served by one of the aircraft. Furthermore, the sequence must include at least one of Xedef's hubs.

To facilitate routing, Xedef wishes to encode the length of the shortest sequence of hops from every city to every hub on the shipping label of every package. (The length of the shortest sequence leading from a hub to itself is zero.) Obviously, a compact representation of this information is required.

You are to implement two procedures, **encode**(N,H,P,A,B) and **decode**(N,H). N is the number of cities and H is the number of hubs. Assume that the cities are numbered from 0 iki $N-1$,and that the hubs are the cities with numbers between 0 and $H-1$. Further assume that $N \leq 1000$ and $H \leq 36$. P is the number of pairs of cities connected by aircraft. All (unordered) pairs of cities will be distinct. A and B are arrays of size P, such that the first pair of connected cities is $(A[0], B[0])$, the second pair is $(A[1], B[1])$ and so on.

encode must compute a sequence of bits from which **decode** can determine the number of hops from every city to every hub. **encode** will transmit the sequence of bits to the grading server by a sequence of calls to **encode_bit**(b) where b is either 0 or 1. **decode** will receive the sequence of bits from the grading server by making calls to **decode_bit**. The i-th call to **decode_bit** will return the value of b from the i-th call to **encode_bit**(b). Note that you must ensure that the number of times **decode** calls **decode_bit** will always be at most equal to the number of times **encode** previously called **encode_bit**(b).

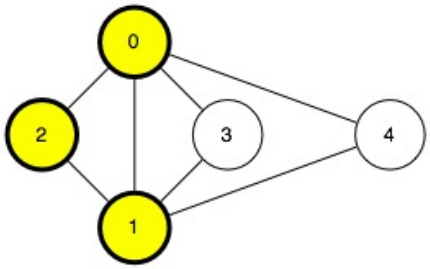
After decoding the numbers of hops, **decode** must call **hops(h,c,d)** for every hub h and every city c (including every hub, that is, also for $c = h$), giving the minimum number d of hops necessary to ship a package between h and c. That is, there must be $N \times H$ calls to **hops(h,c,d)**. The order does not matter. You are guaranteed that it is always possible to ship a package between every hub and every city.

Note: ***encode*** *and* ***decode*** *must communicate only through the specified interface. Shared variables, file access and network access are prohibited. In C or C++, you may declare persistent variables to be* `static` *to retain information for* ***encode*** *or* ***decode****, while preventing them from being shared. In Pascal, you may declare persistent variables in the* `implementation` *part of your solution files.*

Day 2 Task 4

Xedef

Example.



As an example, consider the diagram on the right. It shows five cities ($N = 5$) connected by seven aircraft ($P = 7$). Cities 0, 1 and 2 are hubs ($H = 3$). One hop is needed to ship a package between hub 0 and city 3, whereas 2 hops are needed to ship a package between hub 2 and city 3. The data for this example are in `grader.in.1`.

The entries in the following table are all d-values that **decode** must deliver by calling **hops**(h,c,d):

d		City c				
		0	**1**	**2**	**3**	**4**
Hub h	**0**	0	1	1	1	1
	1	1	0	1	1	1
	2	1	1	0	2	2

1 Subtask [25 points]. **encode** must make no more than 16 000 000 calls to **encode_bit**(b).

2 Subtask [25 points]. **encode** must make no more than 360 000 calls to **encode_bit**(b).

3 Subtask [25 points]. **encode** must make no more than 80 000 calls to **encode_bit**(b).

4 Subtask [25 points]. **encode** must make no more than 70 000 calls to **encode_bit**(b).

Implementation Details.

- Use the *RunC programming and test environment*
- Implementation folder: `/home/ioi2010-contestant/saveit/` (prototype: *saveit.zip*)
- To be implemented by contestant:
  - `encoder.c` or `encoder.cpp` or `encoder.pas`
  - `decoder.c` or `decoder.cpp` or `decoder.pas`
- Contestant interface:
  - `encoder.h` or `encoder.pas`
  - `decoder.h` or `decoder.pas`
- Grader interface: `grader.h` or `graderlib.pas`
- Sample grader: `grader.c` or `grader.cpp` or `grader.pas` *and* `graderlib.pas`
- Sample grader input: `grader.in.1`, `grader.in.2` etc
  Note: The first line of each file contains N P H. The next P lines contain the pairs of

Day 2 Task 4

Xedef

cities $A[0]B[0]$, $A[1]B[1]$, etc. The next $H \times N$ lines contain the numbers of hops from each hub to each city (including itself and all other hubs); that is, the number of hops from a hub i to a city j is in the $i \cdot N + j + 1$-st of these lines.

- Expected output for sample grader input:
  - If the implementation is correct for subtask 1, the output will contain `OK 1`
  - If the implementation is correct for subtask 2, the output will contain `OK 2`
  - If the implementation is correct for subtask 3, the output will contain `OK 3`
  - If the implementation is correct for subtask 4, the output will contain `OK 4`
- Compile and run (command line): `runc grader.c` or `runc grader.cpp` or `runc grader.pas`
- Compile and run (gedit plugin): *Control-R*, while editing any implementation file.
- Submit (command line): `submit grader.c` or `submit grader.cpp` or `submit grader.pas`
- Submit (gedit plugin): *Control-J*, while editing any implementation or grader file.