|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| http://ace.delos.com/usaco/cowhead2.gif | |  | | --- | | Contest: JAN11 **GOLD** Division | |  | |  | |  | |

**ANALYSIS MODE  
Submit solutions for your own enjoyment.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**GOLD PROBLEMS**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Three problems numbered 1 through 3**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Problem 1: Bottleneck [John Pardon, 2010]**

**Farmer John is gathering the cows. His farm contains a network of**

**N (1 <= N <= 100,000) fields conveniently numbered 1..N and connected**

**by N-1 unidirectional paths that eventually lead to field 1. The**

**fields and paths form a tree.**

**Each field i > 1 has a single one-way, exiting path to field P\_i,**

**and currently contains C\_i cows (1 <= C\_i <= 1,000,000,000). In**

**each time unit, no more than M\_i (0 <= M\_i <= 1,000,000,000) cows**

**can travel from field i to field P\_i (1 <= P\_i <= N) (i.e., only**

**M\_i cows can traverse the path).**

**Farmer John wants all the cows to congregate in field 1 (which has**

**no limit on the number of cows it may have). Rules are as follows:**

**\* Time is considered in discrete units.**

**\* Any given cow might traverse multiple paths in the same time**

**unit. However, no more than M\_i total cows can leave field i**

**(i.e., traverse its exit path) in the same time unit.**

**\* Cows never move \*away\* from field #1.**

**In other words, every time step, each cow has the choice either to**

**a) stay in its current field**

**b) move through one or more fields toward field #1, as long**

**as the bottleneck constraints for each path are not violated**

**Farmer John wants to know how many cows can arrive in field 1 by**

**certain times. In particular, he has a list of K (1 <= K <= 10,000)**

**times T\_i (1 <= T\_i <= 1,000,000,000), and he wants to know, for**

**each T\_i in the list, the maximum number of cows that can arrive**

**at field 1 by T\_i if scheduled to optimize this quantity.**

**Consider an example where the tree is a straight line, and the T\_i**

**list contains only T\_1=5, and cows are distibuted as shown:**

**Locn: 1---2---3---4 <-- Pasture ID numbers**

**C\_i: 0 1 12 12 <-- Current number of cows**

**M\_i: 5 8 3 <-- Limits on path traversal; field**

**1 has no limit since it has no exit**

**The solution is as follows; the goal is to move cows to field 1:**

**Tree: 1---2---3---4**

**t=0 0 1 12 12 <-- Initial state**

**t=1 5 4 7 9 <-- field 1 has cows from field 2 and 3**

**t=2 10 7 2 6**

**t=3 15 7 0 3**

**t=4 20 5 0 0**

**t=5 25 0 0 0**

**Thus, the answer is 25: all 25 cows can arrive at field 1 by time**

**t=5.**

**PROBLEM NAME: bottleneck**

**INPUT FORMAT:**

**\* Line 1: Two space-separated integers: N and K**

**\* Lines 2..N: Line i (not i+1) describes field i with three**

**space-separated integers: P\_i, C\_i, and M\_i**

**\* Lines N+1..N+K: Line N+i contains a single integer: T\_i**

**SAMPLE INPUT (file bottleneck.in):**

**4 1**

**1 1 5**

**2 12 7**

**3 12 3**

**5**

**OUTPUT FORMAT:**

**\* Lines 1..K: Line i contains a single integer that is the maximum**

**number of cows that can arrive at field 1 by time T\_i.**

**SAMPLE OUTPUT (file bottleneck.out):**

**25**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Problem 2: The Continental Cowngress [Louis Wasserman, 2010]**

**Displeased with Farmer John's leadership, the cows have seceded**

**from the farm and have formed the first Continental Cowngress.**

**Built on the principle of "every cow gets something they want,"**

**they've decided on the following voting system:**

**The M (1 <= M <= 4000) cows in attendance will vote on N (1 <=**

**N <= 1,000) legislative bills. Each cow casts a "yes" or "no"**

**vote (denoted as 'Y' or 'N' in the input file) on exactly two**

**(distinct) bills B\_i and C\_i (1 <= B\_i <= N; 1 <= C\_i <= N). The**

**votes are called VB\_i (VB\_i in {'Y', 'N'}) and VC\_i (VC\_i in**

**{'Y', 'N'}) respectively.**

**Finally, the bills are to be passed or not in such a way that**

**every cow gets her way on at least one of her votes. For example,**

**if Bessie votes "yes" on Bill 1, and "no" on Bill 2, then in any**

**valid solution, it must be the case that either Bill 1 gets**

**passed or Bill 2 gets rejected (or both).**

**Given the votes of each of the cows, it's your job to figure out**

**which bills will be passed and which bills will be rejected in order**

**to conform to the rules above. If there is no solution, print**

**"IMPOSSIBLE". If there is at least one solution, then for each bill,**

**display:**

**Y if in every solution this bill passes**

**N if in every solution this bill fails**

**? if there are solutions where this bill passes and solutions**

**where it does not pass**

**Consider the following set of votes (two for each cow):**

**- - - - - BILL - - - - -**

**1 2 3**

**Cow 1 YES NO**

**Cow 2 NO NO**

**Cow 3 YES YES**

**Cow 4 YES YES**

**From this, two solutions satisfy every cow:**

**\* Bill 1 passes (this then satisfies cows 1, 3, and 4)**

**\* Bill 2 fails (this then satisfies cow 2)**

**\* Bill 3 could pass or fail (and this is the reason there are**

**two solutions)**

**In fact, these are the only two solutions, so the answer is the**

**three character string below:**

**YN?**

**PROBLEM NAME: cowngress**

**INPUT FORMAT:**

**\* Line 1: Two space-separated integers: N and M**

**\* Lines 2..M+1: Line i+1 describes cow i's votes with four**

**space-separated fields -- an integer, a vote, another integer,**

**and another vote: B\_i, VB\_i, C\_i, VC\_i**

**SAMPLE INPUT (file cowngress.in):**

**3 4**

**1 Y 2 N**

**1 N 2 N**

**1 Y 3 Y**

**1 Y 2 Y**

**OUTPUT FORMAT:**

**\* Line 1: A string with N characters, where the ith character is**

**either a "Y" if the ith bill must pass, an "N" if the ith bill**

**must fail, or a "?" if it cannot be determined whether the**

**bill passes from these votes.**

**If there is no solution which satisfies every cow, then output the**

**single line "IMPOSSIBLE".**

**SAMPLE OUTPUT (file cowngress.out):**

**YN?**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Problem 3: Roads and Planes [Michael Cohen, 2010]**

**Farmer John is conducting research for a new milk contract in a new**

**territory. He intends to distribute milk to T (1 <= T <= 25,000)**

**towns conveniently numbered 1..T which are connected by up to R (1**

**<= R <= 50,000) roads conveniently numbered 1..R and/or P (1 <= P**

**<= 50,000) airplane flights conveniently numbered 1..P.**

**Either road i or plane i connects town A\_i (1 <= A\_i <= T) to town**

**B\_i (1 <= B\_i <= T) with traversal cost C\_i. For roads, 0 <= C\_i**

**<= 10,000; however, due to the strange finances of the airlines,**

**the cost for planes can be quite negative (-10,000 <= C\_i <= 10,000).**

**Roads are bidirectional and can be traversed from A\_i to B\_i or B\_i**

**to A\_i for the same cost; the cost of a road must be non-negative.**

**Planes, however, can only be used in the direction from A\_i to B\_i**

**specified in the input. In fact, if there is a plane from A\_i to**

**B\_i it is guaranteed that there is no way to return from B\_i to A\_i**

**with any sequence of roads and planes due to recent antiterror**

**regulation.**

**Farmer John is known around the world as the source of the world's**

**finest dairy cows. He has in fact received orders for his cows from**

**every single town. He therefore wants to find the cheapest price**

**for delivery to each town from his distribution center in town S**

**(1 <= S <= T) or to know that it is not possible if this is the**

**case.**

**MEMORY LIMIT: 64MB**

**PROBLEM NAME: roadplane**

**INPUT FORMAT:**

**\* Line 1: Four space separated integers: T, R, P, and S**

**\* Lines 2..R+1: Three space separated integers describing a road: A\_i,**

**B\_i and C\_i**

**\* Lines R+2..R+P+1: Three space separated integers describing a plane:**

**A\_i, B\_i and C\_i**

**SAMPLE INPUT (file roadplane.in):**

**6 3 3 4**

**1 2 5**

**3 4 5**

**5 6 10**

**3 5 -100**

**4 6 -100**

**1 3 -10**

**INPUT DETAILS:**

**6 towns. There are roads between town 1 and town 2, town 3 and town 4, and**

**town 5 and town 6 with costs 5, 5 and 10; there are planes from town 3 to**

**town 5, from town 4 to town 6, and from town 1 to town 3 with costs -100, -**

**100 and -10. FJ is based in town 4.**

**OUTPUT FORMAT:**

**\* Lines 1..T: The minimum cost to get from town S to town i, or 'NO**

**PATH' if this is not possible**

**SAMPLE OUTPUT (file roadplane.out):**

**NO PATH**

**NO PATH**

**5**

**0**

**-95**

**-100**

**OUTPUT DETAILS:**

**FJ's cows begin at town 4, and can get to town 3 on the road. They**

**can get to towns 5 and 6 using planes from towns 3 and 4. However,**

**there is no way to get to towns 1 and 2, since they cannot go**

**backwards on the plane from 1 to 3.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***