Optical Network Service Continuity Optimization Challenge

Time Limit: 90s Space Limit: 512MB

Problem Description

An optical network can be viewed as an undirected connected graph consisting of N nodes and M edges, where multiple edges may exist between two nodes. Each edge in the graph represents a real-world optical fiber, with each fiber containing $K = 40$ optical wavelengths.

Initially, multiple optical services are running in the optical network. An optical service with a source s, a destination d, and a bandwidth W can be seen as a **simple path** (i.e., a path without repeated nodes or edges) from s to d. For each edge on the path, W **consecutively numbered** wavelengths may be used; a service uses an channel, which consists of one or several contiguous wavelengths. Multiple services can use different wavelengths of the same edge, but they cannot use the same wavelength of any edge. Additionally, each service has a corresponding service value, represented by V_i for the i^{th} service.

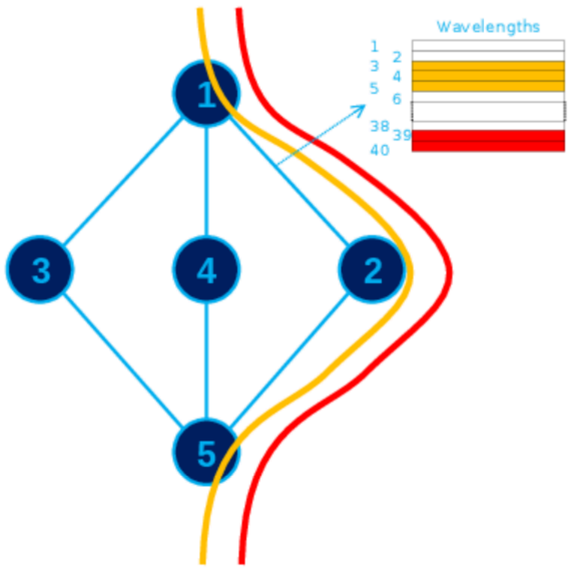

figure 1

As shown in Figure 1, there are currently two optical services running in the optical network; both Service 1 and Service 2 have source at Node 1 and destination at Node 5, with Service 1 having a bandwidth (number of wavelengths used by the service) of 3, using channel {3, 4, 5} (i.e., wavelengths 3, 4 and 5), and Service 2 having a bandwidth of 2, using channel {39, 40} (i.e., wavelengths 39 and 40). Their service paths are (1 → 2 → 5).

Under normal circumstances, for a service passing through edges $e_1, e_2 ... e_n$ in sequence, the set of wavelength indices used should be the same on all edges; specifically, let C_{e_k} be the set of wavelengths used by the service on the k^{th} edge of the path, then $C_{e_{k+1}} = C_{e_k}$. However, nodes have a certain number of channel conversion opportunities P_k. Letting $e_k = (u, v)$ represent the edge connecting nodes u and v, and $e_{k+1} = (v, w)$, if node v still has remaining channel conversion opportunities ($P_v > 0$), then one of the P_v can be used to make $C_{e_{k+1}} \neq C_{e_k}$.Therefore, the source of any path does not involve the use of channel conversion opportunities.

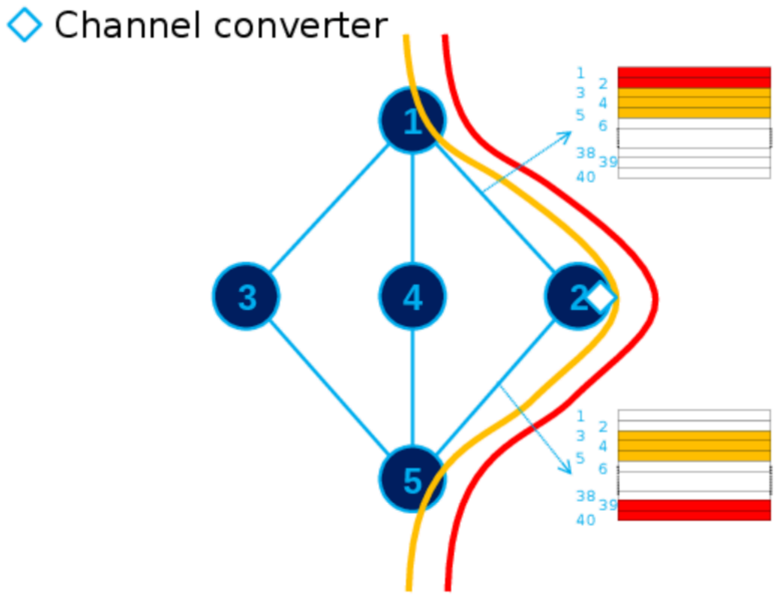

figure 2

As shown in Figure 2, if initially, node 2 has one opportunity to convert channel, it can be observed that Service 2 uses wavelengths {1, 2} on edge (1, 2) and wavelengths {39, 40} on edge (2, 5). Hence, the channel conversion opportunity for node 2 becomes 0, making it impossible for Service 1 to convert channel at node 2 thereafter.

In the real-time operation of the network, various fiber faults (cut) may occur due to natural disasters, hardware failures, or maintenance activities. When a fiber fault occurs, **only** the services whose paths go through the faulty fiber need to be replanned. However, not all services can be successfully replanned.

The replanned service path must satisfy the same source, destination, and bandwidth as the original service attributes, and must meet the aforementioned constraints; Additionally, although a replanned service cannot use the resources of other services, it can use its own old path resources (i.e., channels and channel conversion opportunities). For example, if a fiber fault affects both Service 1 and Service 2, and we denote the old and new path of Service 1 as old1 and new1, and those of Service 2 as old2 and new2, then new1 cannot use the resources of new2, nor the resources of old2, but new1 can use the resources of old1 (and the resources of old1 cannot be used by new2).

If the replanning is successful, the services will release the channels occupied on their original edges and the corresponding nodes' channel conversion opportunities; If new1 uses some of the resources from old1, those resources will not be released after the successful planning. The replanning of services is carried out **simultaneously**, without any sequential order, meaning that the resources of their old paths will only be released after the replanning of this batch of services is completed.

If you fail to replan paths for certain services, they are considered dead, **and the wavelengths occupied on their edges and the corresponding nodes' channel conversion opportunities will not be released. Furthermore, in this test scenario, these dead services cannot be replanned later.**

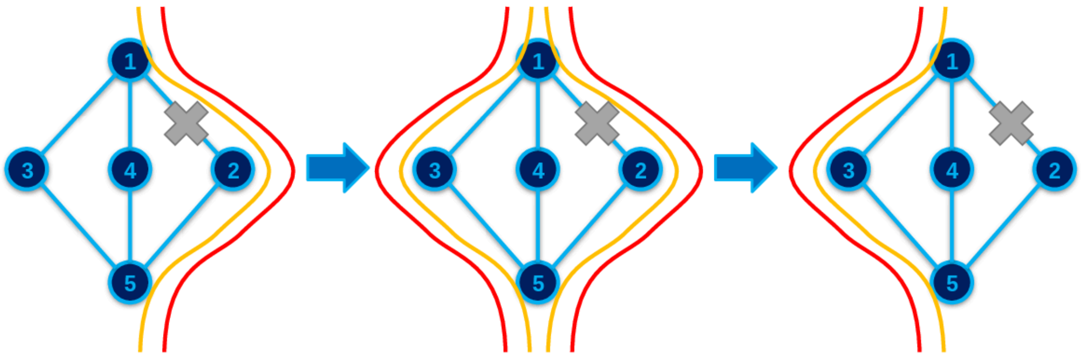

figure 3

As shown in Figure 3, when a disruption occurs on edge (1, 2), Services 1 and 2 will replan their paths, and upon successful replanning, the resources occupied on their original paths (path, wavelengths, channel conversion opportunities) will be released.

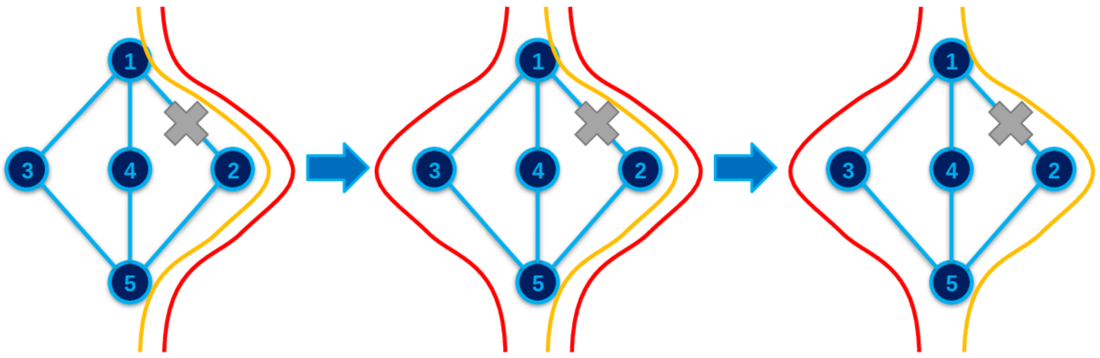

figure 4

As shown in Figure 4, when a disruption occurs on edge (1, 2), only Service 1 successfully replans its path. Although Service 2 dies, it will continue to occupy the resources on its original path indefinitely.

In this scenario, you are required to design an algorithm to ensure that after multiple fiber faults, the total value of surviving services on the network is maximized.

Similarly, we also implemented a corresponding baseline solution. This solution is based on the shortest path algorithm. It sorts the planning path requests of the same batch by weight from largest to smallest, and then finds the global shortest path for each service in succession, using only paths of invariant channels;

your algorithm needs not only to handle the service re-planning after multiple fiber optic failures but also to try to find the bottleneck cases of the baseline solution;

Specifically, after reading the initial environment, your algorithm needs to output T_1 groups of fiber break sequences. The T_1 groups of fiber break sequences output by your algorithm will be combined with our built-in $T_2(1 \leq T_2 \leq 70)$ groups of fiber break sequences as test data;

At the same time, to prevent the T_1 groups of sequences you provide from having too high a similarity, let S_i be the set of sequences for the ith group, and define the Jaccard similarity between two sets as $J_{i,j} = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$ The sequences you provide must ensure that $Max_{i,j} J_{i,j} \leq 0.5, i \in [1, T_1], j \in [1, T_1], i \neq j$;

Problem Interaction

Your program should use the system's standard input stream and standard output stream for input and output.

Initial Environment Input

The first line contains two integers N and M, where N denotes the number of nodes in the graph, and M denotes the number of edges; $(2 \leq N \leq 200, 1 \leq M \leq 1000)$.

The second line contains N integers, where the i^{th} integer P_i represents the maximum number of channel conversion opportunities for node i; $(0 \leq P_i \leq 20)$.

Then, M lines follow, each with two integers u_i, v_i indicating the i^{th} edge (u_i, v_i) in the graph; $(1 \leq u_i, v_i \leq N, u_i \neq v_i)$.

The input graph is guaranteed to be connected, without self-loops, and may have parallel edges.

Next, a line with a single integer J, indicating the number of services initially running on the graph; $(1 \leq J \leq 5000)$.

Then, $2J$ lines follow, with every two lines representing one service.

The first line of each service contains six integers s, d, S, L, R, V indicating the service's source s, destination d, number of edges traversed S, the occupied wavelength range on all edges in the path $[L, R]$, and the service value V; the second line contains S integers indicating the sequence of edge numbers $e_1, e_2 \ldots e_S$ the service traverses, where e_k represents the k^{th} edge on the path; $(1 \leq s, d \leq N, s \neq d, 1 \leq L \leq R \leq K,\ 0 \leq V \leq 100000,$ $1 \leq e_k \leq M)$.

Note that initial services do not use channel converters.

All index in the problem start from 1, and both the service index and edge index increase according to the order of input.

Note that each testing scenario in the interaction part starts from this initial environment.

Interactive Part

Output for bottleneck edge failure scenarios interaction part:

You need to output a single integer T_1 on one line, which represents the number of edge failure test scenarios you provide. Make sure that $0 \leq T_1 \leq 30$.

Then, you need to output $2 \cdot T_1$ lines, with every two lines representing one test scenario. The first number on the first line, c_i, indicates the number of edge failures in that test scenario. On the second line, c_i integers represent the sequence of edge failures by their edge IDs. You must ensure that $0 \leq c_i \leq 60$ and that these c_i integers are valid, unique edge IDs.

Reading test scenarios and planning paths interaction part:

First, a line with a single integer T indicates $T(1 \leq T \leq 100, T = T_1 + T_2)$ independent test scenarios. In processing each scenario, it is guaranteed to **start from the above initial environment** (at this point, no edge has been disrupted yet).

For each scenario, your program needs to receive a series of integers e_{failed}, where e_{failed} indicates the number of the edge that has been disrupted in the network, $(1 \leq e_{failed} \leq M)$. Upon receiving each e_{failed}, your program needs to replan the paths for the services affected by this disruption, as well as plan the channels for each edge on the path. Specifically, for the current e_{failed}, assuming the current set of affected services is $Serv = \{S_1, S_2 ... S_n\}$, your program should output a line with a single integer R, where $0 \leq R \leq n$, indicating the number of services successfully replanned. Then, output $2R$ lines for each successfully replanned service, where the first line contains two integers $Serv_{id}$ and S, indicating the service's index and the number of edges in the new path of the service, respectively. The second line contains $3S$ integers e_k, e_l, e_r, indicating the sequence of edge index the service traverses and the occupied wavelength interval $[e_l, e_r]$ on that edge; this process continues until your program receives an integer -1, which indicates the end of the test scenario. Then, you need to reset your program to the initial environment state and proceed to the next test scenario

(When outputting the planned services, it can be assumed that these services are planned simultaneously, so their order does not matter).

The number of e_failed in each test scenario does not exceed 60.

Note: After each output line, to ensure the interaction program can correctly read your output, you need to flush the output, where:

C and C++ can use fflush(stdout);

Java can use System.out.flush().

Scoring

For each case i, let T_i be the number of test scenarios for this case, let $BeginV_i^j$ be the total value of services at the start of the j^{th} test scenario, let $EndV_i^j$ be the total value of services that survived after the j^{th} test scenario ends; then the score for that test scenario is $Score_i^j = \frac{EndV_i^j \cdot 10000}{BeginV_i^j}$; the score for the case is the sum of scores for all test scenarios $Score_i = \sum_{j=1}^{T_i} Score_i^j$.

Let $Score_i^1$ be the case score of your algorithm, and let $Score_i^2$ be the case score of our baseline algorithm. Then, the score for the problem is the sum of the difference scores for all cases $Score = \sum(Score_i^1 - Score_i^2)$.

The higher the score, the higher the rank; if scores are equal, submissions are ranked by submission time.

Sample

Initial Environment Input

```
5 6

1 1 1 1 1

1 2

2 5

1 4

4 5

1 3

3 5

2

1 5 2 1 20 1

1 2

1 5 2 21 40 1

1 2
```

Interaction Part

(The program output and environment input are just markers for easier understanding; in reality, this string will not be output.)

```
Program output: 1

Program output: 2
Program output: 1 6

Environment input: 1

Environment input: 1

Program output: 2

Program output: 1 2

Program output: 5 1 20 6 1 20

Program output: 2 2

Program output: 5 21 40 6 21 40

Environment input: 6

Program output: 2

Program output: 1 2

Program output: 3 1 20 4 1 20

Program output: 2 2

Program output: 3 21 40 4 21 40

Environment input: -1
```

Error Types

Basic Error Types

1. Code Compilation Error
2. Program Abnormal Exit (Possible reasons: Runtime error, use of exceptional permissions, more output parameters than expected, incorrect format of output parameters, etc...)
3. Time Limit Exceeded (Possible reasons: Failure to use flush command during interaction, program running overtime, fewer output parameters than expected, etc...)
4. Memory Limit Exceeded

Logical Error Types

1. "Unknown Error" (Please contact the competition organizers if this occurs)
2. "Incorrect Number of Services" (The number of services output is incorrect)
3. "Incorrect Service ID" (The service ID output is incorrect)
4. "Duplicate Service ID" (Duplicate service ID in a single output)
5. "Unaffected Service ID" (Output includes service IDs not affected by this fiber cut)
6. "Incorrect Number of Edges" (The number of edges in the path is incorrect)
7. "Incorrect Edge ID" (The edge ID in the path is incorrect)
8. "Duplicate Edge ID" (Duplicate edge ID on the path)
9. "Pass Break Edge" (The path goes through a broken fiber)
10. "Inconsistent Service Width" (Inconsistent service width along the path)

11. "Incorrect wavelength ID" (The wavelength ID used by the service is incorrect)
12. "Cyclic Path" (The path forms a loop)
13. "Wavelength Occupied Kind 1" (The wavelength used by the service is occupied, possibly related to other services' old paths)
14. "Wavelength Occupied Kind 2" (The wavelength used by the service is occupied, possibly related to other services' new paths)
15. "Disconnected Path" (The path is not connected)
16. "Insufficient Wavelength Quantity" (Insufficient number of channel conversion opportunities)
17. "Mismatched Start and End" (The start or end of the new path for the service does not match the old path)
18. "Incorrect Number of TestCase" (The number of test scenarios output or the number of interrupted edges exceeds the range)
19. "Incorrect Edge ID of TestCase" (The interrupted edge ID in the output test sequence is incorrect)
20. "Duplicate Edge ID of TestCase" (Duplicate interrupted edges in the output test sequence)
21. "Large Jaccard Similarity Between TestCase" (The Jaccard similarity between the output test sequences exceeds the threshold)