**Problem:**

The winter has come, yet the radiators are not made to work and METU Ceng students are freezing. All department students want is to be able to study in the department laboratories and study-rooms for 7x24. However, they can't do it because of the freezing cold of the department at nights. When they ask the Energy Department of the University to work the radiators, they take the following answer:

"

Dear students,

We lost the 'pipeline usage plan' that provides us to work the radiators economically at the university overall. By using that plan, it is possible to activate the furnace rooms of only the keypoint buildings and work the radiators of the other buildings via the pipelines connected to those keypoints. Nonetheless, without the plan, we have to either activate all the furnace rooms (the furnace room of each building), or deactivate all. Since our budget can not afford to apply the first choice, we apply the second choice. We hope to find the usage plan as soon as possible.

**"**

Upon this reply, the lovely and clever 3rd class Ceng students become volunteer to solve the problem since they do no want to freeze anymore. They ask the map of pipelines in the university and start to work on a new pipeline usage plan.

**PART-I**

You should find a solution such that radiators of each building could be managable and the resulting energy consumption should be in the minimum level.

* + Radiators of a building become managable either via activation of the furnace room of that building or via a pipeline connecting it to another building whose radiators are managed (by its own furnace room or again via a pipeline connected to some other building, so on).
  + Each building has its own furnace room.
  + Amount of the energy consumption is defined differently for each managing method. That is, each pipeline providing the energy transfer between the departments have a different cost (results in a different amount of energy consumption). Also, for each building the cost of activating its own furnace room differs.
  + Note that it is necessary to activate the furnace room of at least one building (this becomes the source), so the others can find the energy through the pipelines reaching them to the source somehow.
  + There can be more than one pipeline connection between two buildings, or between the building itself and its furnace room.
  + There may be more than one cheapest way to warm up the whole university. It is enough for you to find only one.

**Implementation:**

* + Each pipeline is defined as below:

**struct Pipeline {  
    int id;  
    int end1;  
    int end2;  
    int cost\_of\_consumption;  
};**

* + - *id* is a unique integer representing the id of the pipeline.
    - *end1* and *end2* represents the building ids connected via the pipeline. Each building also has a unique id. If *end1=end2*, then it means the pipeline represents the connection between the building itself and its own furnace room.
    - *amount\_of\_consumption* is the cost/amount of the consumption caused by the usage of that pipeline.
  + You will implement the following function for the solution:

**vector<int> plan\_min\_cost\_pipeline\_usage(vector<Pipeline\*> pipelines, int num\_of\_buildings);**

* + - *vector<Pipeline\*> pipelines*holds the pipeline map of the university. Each pipeline is uniquely given in this vector. (The uniqueness is satisfied by the pipelined ids, the other Pipeline members may be the same.)
    - *int num\_of\_buildings* is the total number of buildings located in the university.
    - The function returns the usage plan holding the ids of the pipelines providing to warm up each building with the minimum cost/amoun*t.*
  + The ordering of the ids in the returned vector is not important.

**PART-II**

After you find a solution in the first part, the Energy Department informs you that the current pipeline map is missing. They say that there actually exists one more pipeline which has been forgotten to put into the map. They request you to update your solution by considering this pipeline. It is also possible the pipeline does not affect the current solution, yet you should check it too.

**Implementation:**

* + You will implement the following function for the solution:

**vector<int> optimize\_plan(vector<Pipeline\*> current\_solution, Pipeline\* new\_pipeline);**

* + - *vector<Pipeline\*> current\_solution* is the solution that is expected to be found in the PART-I, yet the vector holds the pipelines itselves, not their ids.. We will call the method by supplying this solution in the first argument.
    - *Pipeline\* new\_pipeline* is the new pipeline which is missing in the previous map.
    - The function returns the updated usage plan holding the ids of the pipelines.
  + The ordering of the ids in the returned vector is not important.
  + **You are required to implement a *O(N)* solution where *N*is the number of buildings. This requirement consists of 20% of THE in grading.**
  + Note that it would be enough to hold an efficient data structure like **adjacency matrix of the graph**. There is no need to use complex structures like heap etc.

**Example-1:**

**metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Example-2:**

metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Example-3:**

metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, diyagram, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Example-4:**

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Constraints:**

* Maximum number of buildings in the university will be 512.
* Maximum number of pipelines is 4096.
* Only for the Testcase-11, this limits change. Maximum  number of buildings for this testcase is 218.

**Evaluation:**

* After your exam, black box evaluation will be carried out. You will get full points if your both function returns the correct result without exceeding time limit.
* There are 10 testcases to test output of the functions. PART-I is 7 points and PART-II is 1 points for each.
* There is one last testcase (Testcase-11) which checks the complexity of your implementation for PART-II. This testcase will be run only if you get full credit for Part-II implementation from all other testcases. If your Part-II implementation is O(N) (where N is the number of buildings), then you get 20 points. If it is more, then you get 0.
* The grading of PART-I and PART-II are independent from each other. Only the Testcase-11 is dependent to get full credit from PART-II for each testcase.

**Specifications:**

* There are **only 2 tasks** to be solved in **12 hours** in this take home exam.
* You will implement your solutions in **the6.cpp** file.
* Do **not**change the first line of **the6.cpp**, which is **#include "the6.h"**
* *<iostream>,  <climits>, <vector>, <string>, <stack>, <queue>, <unordered\_set>*are included in "the6.h" for your convenience.
* Do **not**change the arguments and return **types** of the functions**.**(You should change return **value**, on the other hand.)
* Do **not**include any other library or write include anywhere in your **the6.cpp** file (not even in comments)

**Compilation:**

* You are given **test.cpp** file to **test**your work on **ODTÜClass** or your **locale**. You can and you are encouraged to modify this file to add different test cases.
* If you want to **test**your work and see your outputs you can **compile and run**your work on your locale as:

|  |
| --- |
| >g++ test.cpp the6.cpp -Wall -std=c++11 -o test  > ./test |

* You can test your **the6.cpp** on virtual lab environment. If you click **run**, your function will be compiled and executed with **test.cpp**. If you click **evaluate**, you will get a feedback for your current work and your work will be **temporarily**graded for **limited**number of inputs.
* The grade you see in lab is **not** your final grade, your code will be re-evaluated with **completely** **different**inputs after the exam.

The system has the following limits:

* a maximum execution time of 32 seconds for the Testcases 1-10, and 60 seconds for the Testcase11
* a 4GiB maximum memory limit
* an execution file size of 128MiB
* If you are sure that your solution works in the expected complexity constrains but your evaluation fails due to limits in the lab environment, the constant factors may be the problem.

|  |
| --- |
| vector<int> plan\_min\_cost\_pipeline\_usage(vector<Pipeline\*> pipelines, int num\_of\_buildings);  vector<int> optimize\_plan(vector<Pipeline\*> current\_solution, Pipeline\* new\_pipeline); |