Practicum Assignment 2

The problem

The Netherlands largely consists of *polders*, which are low-lying parts of the land, often reclaimed from a body of water. We have dykes as a physical barrier to ensure that the water does not enter the polder. Moreover, we have pumping stations to pump the water out of the polders during rainy periods and to pump it into the polder during droughts. There are several thousand pumping stations throughout the Netherlands. The water direction can only be changed manually at the pumping station.

Figure 1: Pumping station in the Ooijpolder near Nijmegen

It is currently December 24^{th}, just before Christmas, and the weather forecast looks very bad. There will be heavy rain all throughout Christmas! All people working at the pumping stations have made plans to spend Christmas with their families and cannot change the water direction in the coming days. Moreover, all pumping stations are currently not pumping the water into or out of the polder.

It would be horrible if all the polders flooded during Christmas and since the pumping station workers are not available, you are the only one who can save the country! You have to go to all the pumping stations and ensure that the water gets pumped out of the polder.

Luckily, Google Maps has all the information on the Dutch roads, the estimated travel time between road intersections and the locations of the pumping stations. Formally, you are given a set of road intersections V and a

set of pumping stations $W \subseteq V$. The set E consists of (bidirectional) roads which are represented by triples (v_i, v_j, d) with $v_i, v_j \in V$ and d a natural number representing the number of minutes required to travel from road intersection v_i to v_j. Moreover, you are given a natural number t representing the time in minutes before your own Christmas party starts. It takes exactly 10 minutes to change the direction of the water at a pumping station. Each pumping station can pump $200m^3$ water out of the polder per minute. The water direction can only be changed once at every pumping station. Initially, $0m^3$/minute gets pumped out of the polder. It is possible that there is not enough time to change the water direction at every pumping station.

We say that a *pumping station route* is a sequence of road intersections. It is not necessary that the pumping station route ends at the starting point. To save both the country and your Christmas party, you need to determine a pumping station route that maximizes the total amount of water that can get pumped out of the polders in t minutes.

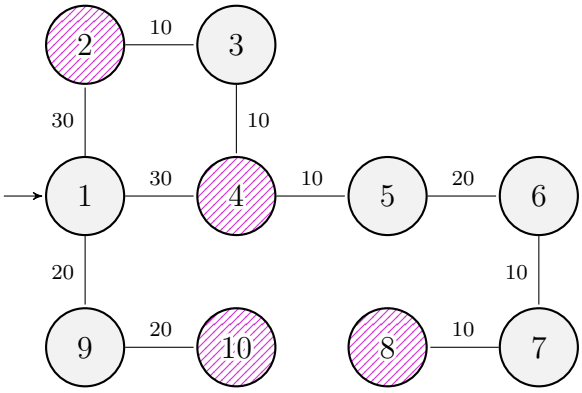

Figure 2: Example input graph.

Example. Consider the graph displayed in Figure 2 (or the equivalent input sample 1 on page 5). There are 10 road intersections and 4 pumping stations. The pumping stations are located at the pink and lined-through vertices. Assume you only have 300 minutes to spare before you have to get to your Christmas party. The optimal pumping station route would be:

$$1, 2, 3, 4, 5, 6, 7, 8, 7, 6, 5, 4, 1, 9, 10$$

This optimal route pumps $140000m^3$ water out of the polder at the end of the 300 minutes with the timeline displayed in Table 1.

Time period	Action	m^3/min in period	Total m^3	Visited stations
300 - 270	Go to 2	0	0	-
270 - 260	Change direction	0	0	-
260 - 250	Go to 3	$200 \cdot 10$	2000	2
250 - 240	Go to 4	$200 \cdot 10$	4000	2
240 - 230	Change direction	$200 \cdot 10$	6000	2, 4
230 - 220	Go to 5	$400 \cdot 10$	10000	2, 4
220 - 200	Go to 6	$400 \cdot 20$	18000	2, 4
200 - 190	Go to 7	$400 \cdot 10$	22000	2, 4
190 - 180	Go to 8	$400 \cdot 10$	26000	2, 4
180 - 170	Change direction	$400 \cdot 10$	30000	2, 4
170 - 160	Go to 7	$600 \cdot 10$	36000	2, 4, 8
160 - 150	Go to 6	$600 \cdot 10$	42000	2, 4, 8
150 - 130	Go to 5	$600 \cdot 20$	54000	2, 4, 8
130 - 120	Go to 4	$600 \cdot 10$	60000	2, 4, 8
120 - 90	Go to 0	$600 \cdot 30$	78000	2, 4, 8
90 - 70	Go to 9	$600 \cdot 20$	90000	2, 4, 8
70 - 50	Go to 10	$600 \cdot 20$	102000	2, 4, 8
50 - 40	Change direction	$600 \cdot 10$	108000	2, 4, 8
40 - 0	-	$800 \cdot 40$	140000	2, 4, 8, 10

Table 1: Timeline for the example

Instructions

You may work in groups of two. For this assignment, you have to hand in two things:

- Source code. We will run the code for grading.
- A report. In the report, you have to explain the algorithm and analyse it.

Source code You are allowed to submit a solution in either C, C++, Java or Python 3. If you prefer another language, please contact teaching assistant Cas Visser asap via Discord or `cas.visser@ru.nl` (no guarantees!) You must write all code yourself, copying code from fellow students or the internet is considered plagiarism. You can only use libraries that come with a minimal installation of the language. We will soon set up a test website on which you can upload and test your code.

Report Besides handing in code, we would like to receive a report in which you explain your algorithm and analyse its correctness and runtime com-

plexity. The report is important and determines 40% of the final grade! We expect a clearly written high-level explanation of how your algorithm works, and a convincing analysis of correctness and asymptotic complexity.

Submission The deadline for sending in your solution is on January 2. You must submit your solutions via Brightspace as a group assignment for the group "Practical 2" (for which all team members need to enroll, even if you submit alone!). Only one team member has to submit a solution; the names and student numbers of both team members must be mentioned in the report. Please make sure your source code compiles and runs on domjudge, is directly accessible (i.e. not in a zip file) and contains only a single file (i.e. not a complete Java project). We will run the code for grading.

Grading Grades will be determined as follows. You may earn up to 100 points for your solution:

- 20 points for the explanation of your algorithm.
- 10 points for the correctness analysis.
- 10 points for the complexity analysis.
- 50 points for the test results. We will be running several tests and you will get points for every correct answer within the time limit. (The exact time limit will be determined after the submission deadline, but it will be in the order of 1sec.) If your code does not compile or does not read and write via `stdin` and `stdout`, you will get zero points on the test cases. So please test your code well!
- 10 points for the quality of the code.

If you have any questions, do not hesitate to contact teaching assistant Cas Visser via Discord (there is a special channel practical-assignments) or via email `cas.visser@ru.nl`. You may also contact lecturer Frits Vaandrager via `F.Vaandrager@cs.ru.nl`.

Input

- The first line has four numbers:
  - The number of road intersections, $1 \leq v \leq 5000$
  - The number of pumping stations, $1 \leq w \leq \min(v, 20)$
  - The number of roads, $1 \leq e \leq 10000$
  - The time limit, $1 \leq t \leq 10000$
- Then follow w unique lines with a natural number of at most v
- Next, there are e lines each containing three natural numbers. Each line contains two road intersections represented as natural numbers of at most

v and one natural number of at most 1000 representing the estimated travel time.

Disclaimer: all the above bounds may still change. We have not finished benchmarking the different solutions yet, but so far it looks like the maximum problem size will have to be a bit smaller.

Output The maximum amount of water in m^3 moved out of the polder at the end of the t minutes as an integer.

Sample inputs and outputs Here are some examples:

Sample 1	
Input	Output
10 4 10 300 2 4 8 10 1 2 30 1 4 30 1 9 20 2 3 10 3 4 10 4 5 10 5 6 20 6 7 10 7 8 10 9 10 20	140000

Sample 2	
Input	Output
4 4 5 80 1 2 3 4 1 2 10 1 3 10 1 4 20 2 3 30 3 4 10	30000