

15-451/651 Algorithm Design & Analysis, Fall 2026

Programming Assignment #2

Programming Assignment #2, Due: Friday, October 9th at 11:59 pm

This is a programming assignment. You need to write code to solve the problems below, reading input from `stdin` and writing the answer to `stdout`. This problem must be completed solo, including the theoretical and conceptual ideas as well as the implementation. All the code you submit must be your own. All submissions will be screened for plagiarism. You may choose to solve the problems in **C++, C, Python, Rust, Java, or OCaml**.

Multiple Problems: This homework, unlike previous homeworks, has two problems. Your overall grade will be $\min(100\%, \frac{2}{5}(2x + y))$, where x and y are your grades (as real numbers from 0 to 1) on problems 1 and 2 respectively. Note that this means a perfect score on both problems is not required to receive an overall perfect score, but is strongly encouraged.

Submission: Your solution to each problem should be submitted as a single code file using Gradescope. Submit only the code file with an appropriate file extension (`.c` for C, `.cpp` for C++, `.py` for Python, `.rs` for Rust, `.java` for Java, and `.ml` for OCaml). Do **not** put your file inside a tar or zip archive. Submissions for the first problem should be submitted under Programming Problem 2 (A), and for the second one under Programming Problem 2 (B).

All submissions must be legitimate attempts to solve the problem. It will be considered an academic integrity violation to make submissions for any other purpose. For example, but not limited to: mass submitting submissions to DOS (denial-of-service attack) the autograder, submitting code that attempts to exfiltrate information from the autograder or tamper with the autograder or the machine it runs on, etc.

Assignment: the Yellow-Cricket Treefrog

Deep in the humid jungle of Costa Rica there lives a yellow tree-frog, no larger than a thumb. This delightful animal hops from leaf to leaf, snacking on small insects and dew-drops to survive. Today, the frog is planning to visit a nearby bush and hop around – but, being such a small critter, he’s worried that the distances he’ll have to hop might be too much for his legs to handle!

To help him out, you’ll receive a description of the bush he’s planning to hop around in, which has been reduced to a set of leaves and some pairwise hoppable distances between them. The resulting graph is guaranteed to be connected. You’ll then receive a list of possible pairs of leaves, where the frog’s plan is to make his way from one to the other via some sequence of hops. The frog wants to *minimize the length of the longest hop* he’ll have to make along his chosen path; for each pair, your task is to tell him the length of that longest hop, assuming he’ll hop his path optimally to minimize that quantity.

The yellow tree-frog is well accustomed to trees (it’s in the name!) – the guanacaste, the *cecropia* trumpet-tree, the walking palm – and remembers learning about a certain type of tree in 15-210, the *minimum-spanning* kind. He’s also well acquainted with the foliage of *segment* and *union-find* trees. Maybe some of these rare rainforest species could help you in your task!

Input You’ll receive a line containing two space-separated integers, V and E , the number of leaves and provided hop-lengths respectively. You’ll then receive E lines, with line i containing three space-separated integers a_i, b_i, c_i . a_i and b_i will be leaf numbers between 0 and $V - 1$, while c_i will be the hop length between those two.

After this, you’ll receive a line containing the integer Q , and then Q lines, with line j containing two space-separated integers u_j and v_j , which are leaf numbers between 0 and $V - 1$.

You are guaranteed that self-loops and parallel edges will be absent from this graph. What’s more, there are certain bounds on the size of the input:

$$\begin{aligned} 2 &\leq V \leq 10^5 \\ V - 1 &\leq E \leq 2 \cdot 10^5 \\ 1 &\leq Q \leq 10^5 \\ 1 &\leq c_i \leq 10^9 \end{aligned}$$

Output Please output Q lines, with line j containing the minimal value of the heaviest edge along a path connecting u_j and v_j , as defined above.

Program limits The base time limit is 2.5 seconds per run. Java gets 50% extra time, and Python gets 500% extra time. The memory limit is 512MB for all languages.

Complexity requirement Your solution may spend $O(E \log E)$ time preprocessing, at most, and then must answer each query in polylogarithmic time. We reserve the right to manually deduct points from submissions that use an inefficient or brute-force algorithm. The target is $O(\log V)$ per query, although $O((\log V)^2)$ is also acceptable.

Scoring The problem is worth 20 points. You will lose 3 points per failed test. There are 25 total secret test cases (not including the 2 sample tests below which are only for debugging and are not scored). Your program will receive zero points if it does not pass the sample test cases.

Sample input/output:

Input	Output
4 5 0 1 3 1 2 5 0 2 4 2 3 2 1 3 7 3 0 3 1 3 0 1	4 4 3

Input	Output
2 2 0 1 10 1 0 6 1 1 0	6

Apples

At the end of the autumn harvest, a farm has N crates of apples, numbered $1, 2, \dots, N$. Crate i has weight w_i and can be sold for value v_i . A truck can carry at most W units of weight. Choose a subset of the crates to load onto the truck without exceeding its capacity. Find the maximum total value of the crates that can be transported.

Input The first line of the input contains N and W , separated by a space. The following N lines each contain two space-separated integers w_i and v_i , representing the weight and value of crate i .

You may assume that $1 \leq N \leq 100$, $1 \leq w_i \leq W$, and $1 \leq v_i \leq 10^9$. Every input satisfies at least one of the following conditions:

- $1 \leq W \leq 10^5$, or
- $1 \leq W \leq 10^9$ and $v_i \leq 10^3$ for every $1 \leq i \leq N$.

Output Output an integer representing the maximum total value of a subset of crates whose combined weight does not exceed W .

Program limits The base time limit is 2 seconds per run. Java gets 50% extra time, and Python gets 100% extra time. The memory limit is 512MB for all languages.

Complexity requirement When $W \leq 10^5$, your solution should run in $O(NW)$ time. When $W > 10^5$, your solution should run in $O\left(N \sum_{i=1}^N v_i\right)$ time. Your program should select the appropriate approach based on the value of W . We reserve the right to manually deduct points from submissions that use an inefficient or brute-force algorithm.

Scoring The problem is worth 19 points. You will lose 3 points per failed test. There are 19 secret test cases, not including the three sample tests below, which are provided only for debugging and are not scored. Your program will receive zero points if it does not pass all sample test cases.

Sample input/output:

Input	Output
3 8 3 30 4 50 5 60	90

Input	Output
4 1000000000 600000000 1000 500000000 900 400000000 800 300000000 700	1800

Input	Output
4 5 1 2 2 4 3 4 4 5	8