Attack on Titans - Evacuation Arc

The world is under siege. Enormous humanoid creatures known as **Titans** have begun invading human cities, leaving destruction in their wake. These Titans are relentless, mindless predators that devour humans without mercy. The last remnants of humanity live within **a network of fortified cities** connected by roads.

Commander Erwin Smith, the fearless and strategic leader of the Survey Corps, is renowned for his unwavering resolve and willingness to sacrifice everything for humanity's survival. He commands elite soldiers like Captain Levi, who can single-handedly take down Titans.

Today, a dire message arrives—Titans are spawning from all directions, charging toward the fortified cities. Normally, the walls would hold, but among the horde are the **Armored Titan** and the **Colossal Titan**, capable of breaching the city's defenses and unleashing chaos. With countless Titans ready to pour in and devour civilians, immediate action is required.

Fighting back is not an option—taking down Titans requires coordinated teamwork, and spreading the soldiers too thin across multiple cities would be suicidal. Instead, you, **Commander Erwin**, must devise an urgent evacuation plan, directing civilians to **designated safe shelters** where Captain Levi and the Survey Corps can defend them. Every second counts, and the fate of humanity rests on your decisions.

However, the evacuation is anything but simple. Some cities are densely populated, and moving everyone at once risks **severe bottlenecks** on key roads. Each road has a **limited capacity**, meaning only a certain number of people can travel at a time. All the evacuees walk at a constant pace of **5 km/h.**

For instance, consider a road that **holds 100 people** and stretches **2 km long**, with evacuees moving at **5 km/h**. If **250 people** need to cross, they must move in batches. The first **100 people** will take **2/5 hours (24 minutes)** to traverse, followed by the next batch, and so on. In total, it would take **(2/5) × ceil(250/100) = 1 hour 12 minutes (72 minutes)** for everyone to cross.

Additionally, the **evacuation must happen in stages—all civilians from one city must reach the next safe checkpoint before moving forward**. The Survey Corps has **limited soldiers** to guard the escape routes. If civilians scatter, defenses weaken, leaving many vulnerable. Securing one city at a time ensures safe escort, preventing unprotected groups from being caught mid-journey.

Some citizens, especially the **elderly**, cannot walk more than a certain **distance**. If a shelter is beyond this distance, they must be **left behind** in a city within their distance limit, hoping the Titans **don't attack** it.

You can drop individuals (whether elderly or prime-age) in any city or shelter. However, only

those dropped at a shelter will be counted as saved. Each shelter has **limited capacity** and if the number of people dropped at a shelter exceeds its capacity, the excess individuals will not survive. Additionally, a penalty will be applied due to overcrowding. For instance, if a shelter has a capacity of 10 but receives 14 people, the extra 4 will not make it. Moreover, the same number (4) from within the shelter will also be lost due to chaos—perhaps a stampede breaking out as people scramble for limited resources. However, the **maximum penalty** will not exceed the shelter's capacity—so in this case, at most 10 people (including the excess and penalty) would be lost.

Contention for Road Usage

1. **Sequential usage**: If evacuees from City X are already traveling along Path P from A to B, then any evacuees from City Y arriving at A or B must wait until the last person from City X reaches B before they can start moving on path P. Note that evacuees from City Y can start using the path as soon as the last person from City X reaches B. If the last person from City X reaches B at time T, then evacuees from City Y can begin using the path from time T onward.
2. **Priority Based on Population**: If evacuees from multiple cities arrive at either end of the road simultaneously, priority is given to the city with the larger population. The city with more evacuees will be allowed to use the road first, while others must wait their turn. Note that evacuees refer to people who are not-yet allotted shelter i.e they plan to move to the next city. If multiple cities arrive at the same time with an equal number of evacuees, priority is given to the city with the lower numerical identifier. For example, if City 2 and City 5 reach the road simultaneously with the same number of evacuees, City 2 will be given priority.

Your strategy must output the **evacuation path**, along with the **number of people dropped at shelters/cities along the way**. The path continues until the **last evacuee from a city** reaches their final destination—whether it's a shelter or a city.

Input

- `num_cities` : Number of cities in the graph.
- `num_roads` : Number of roads in the graph.
- `roads` : A list of roads where each road is represented as `[u, v, length, capacity]`, indicating a road between city `u` and city `v` with a given `length` and `capacity`.
- `num_shelters`: Number of shelter cities
- `shelters` : A list of shelters where each shelter is represented as [city , capacity]
- `num_populated_cities`: Number of populated cities
- `populated_cities` : A list of cities where each city is represented as [city , prime-age, elderly] , representing the number of **prime-age** and **elderly** citizens in that city. All other cities are **deserted**.

- `max_distance_elderly` : The maximum distance an elderly citizen can travel

Output

- `long long *path_size`: An array of size `num_populated_cities`, where `path_size[i]` represents the number of cities encountered along the `i`-th evacuation path.

- `long long **paths`: A 2D array representing `num_populated_cities` evacuation paths. Each `paths[i]` is a list of city indices that define the evacuation route for the `i`-th path.

- `long long *num_drops`: An array of size `num_populated_cities`, where `num_shelters[i]` indicates the number of shelters encountered along the `i`-th path.

- `long long ***drops`: A 3D array where `drops[i]` represents the shelters (and cities for elderly evacuees) along the `i`-th path. Each entry in `drops[i]` contains a list of shelters or cities where drops were made, preserving their order along the path. Each entry consists of three values:

    1. **Shelter/City number**

    2. **Number of prime-age people dropped off**

    3. **Number of elderly people dropped off**

Constraints

- 1 <= num_roads <= 10,00,000
- 1 <= length <= 1000
- 1 <= capacity <= 1000
- 1 <= num_cities <= 1,00,000
- 1 <= num_shelters <= num_cities
- 1 <= num_populated_cities: <= 10,000
- 1 <= Number of prime-age <= 10000
- 1 <= Number of elderly <= 10000
- 0 <= max_distance_elderly <= 5000

All distances are in km.

Example

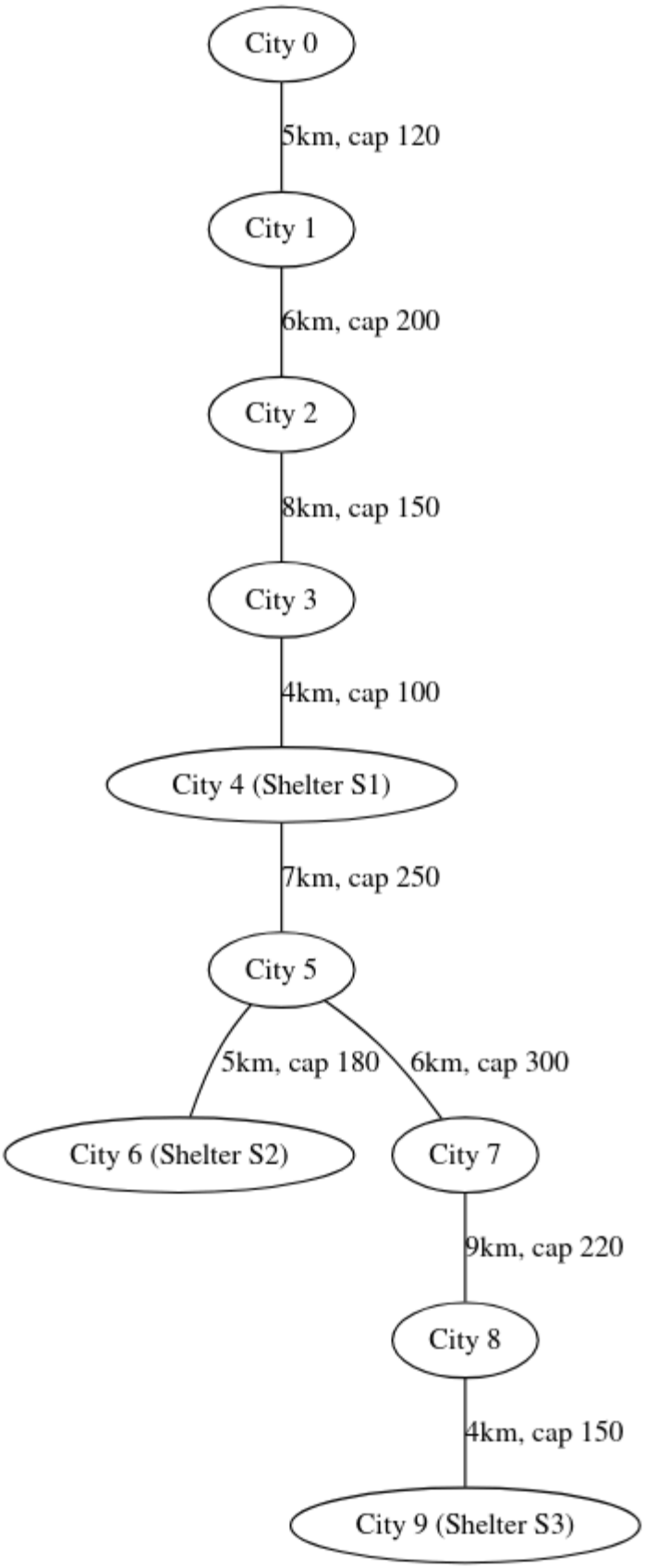
Assume max_distance_elderly = 10 Km

Populated Cities

City	Prime-age	Elderly
City 0	400	80
City 2	300	60
City 5	500	90

Shelters

Shelter	Location	Capacity
S1	City 4	500
S2	City 6	600
S3	City 9	400


City 0
5km, cap 120
City 1
6km, cap 200
City 2
8km, cap 150
City 3
4km, cap 100
City 4 (Shelter S1)
7km, cap 250
City 5
5km, cap 180
6km, cap 300
City 6 (Shelter S2)
City 7
9km, cap 220
City 8
4km, cap 150
City 9 (Shelter S3)

There may be multiple valid evacuation strategies. One possible solution is as follows:

Evacuees from City 0, City 2, and City 5 will begin moving towards Shelter S1 (City 4).

1. City 5 Evacuation:
    - The first batch of 250 evacuees from City 5 reaches City 4 in 84 minutes.
    - The second and third batches follow, taking a total of 252 minutes (84 × 3).
    - At City 4, 90 elderly and 410 prime-age evacuees from City 5 are dropped off, filling Shelter S1.
    - The remaining 90 evacuees from City 5 must continue towards Shelter S2 (City 6).
2. City 2 Evacuation:
    - Evacuees from City 2 reach City 3 in 288 minutes.
    - Since elderly citizens can travel only 10 km, they must remain in City 3.
    - The prime-age evacuees from City 2 continue towards City 4, arriving after an additional 144 minutes.
    - However, since Shelter S1 (City 4) is already full, they must move towards Shelter S2 (City 6) or Shelter S3 (City 9).
    - Assuming they move towards City 6, Shelter S2 can accommodate all 300 prime-age evacuees from City 2, as it currently has only 90 evacuees from City 5.
3. City 0 Evacuation:
    - Elderly evacuees from City 0 will not reach any shelter. Hence, they will be dropped at city 1.
    - By the time, prime-age population arrives at City 4, it is already full.
    - They must continue moving towards City 6 or City 9.
    - Since Shelter S3 (City 9) has enough space, it is preferable for them to go there, ensuring all prime-age evacuees from City 0 are accommodated.
4. Priority Considerations:
    - Evacuees from City 2 will reach Shelter S1 earlier, as they are positioned further along the path.
    - Due to the 10 km travel restriction, elderly evacuees from City 0 and City 2 were dropped at City 1 and City 3, respectively.
    - The remaining 300 prime-age evacuees from City 2 were directed towards Shelter S1 (City 4).
    - Since Shelter S1 has a total capacity of 500, it can accommodate only 200 prime-age evacuees from City 0.

This solution efficiently allocates evacuees while respecting road capacities, shelter limitations, and elderly travel constraints.

The solution will look as follows -

paths
0 -> 1 -> 2 -> 3 -> 4 -> 5 -> 7 -> 8 -> 9
2 -> 3 -> 4 ->5 -> 6
5 -> 4 -> 5 -> 6

drops
[1, 0, 80], [9, 400, 0]
[3, 0, 60], [6, 300, 0]
[4, 410, 90], [6, 90, 0]

You have to set your final answers to **long long *path_sizes, long long **paths, long long* num_drops** and **`long long*** drops`**

path_sizes is an array of size num_populated_cities, where the *i*-th element represents the size of the *i*-th path. The size of **paths** will also be equal to the num_populated_cities, where each i-th element points to another array of size **path_sizes[i]**, which stores the corresponding path. **drops** is a three-dimensional array of size num_populated_cities, where each *i*-th element is a two-dimensional array. Each **drops[i]** contains multiple rows, where each row represents a set of drop details. The number of rows in **drops[i]** is given by **num_drops[i]**, and each row consists of three values representing city/shelter number, number of prime people dropped and number of old people dropped. If no one is being dropped, there is no need to include it in the drop information. Additionally, the order of drops within **drops[i]** should match the order of the corresponding path in **paths[i].**

Grading Criteria

Your solution will be evaluated based on a simulation that processes the **evacuation paths** provided in the output. The simulation will compute:

1. Our evaluation script will abort the process (your program), if the runtime exceeds 10 minutes. Ensure that answers are dumped in the output file **before 10 minutes**.
2. **Total Evacuation Time** – The simulated time taken until the last evacuee is dropped off. Our script will use your solution — the evacuation paths and drop points — to run a simulation and compute the total simulation time. **Please note** that the evacuation time refers to the simulated time in the scenario, not the actual runtime of your program.
3. **Number of Evacuated People** – The total number of evacuees successfully saved.
4. We will be using **40 private test cases** for grading.

- **30 test cases** will cover a variety of scenarios and edge cases to check how efficiently you handle different situations. These test cases will not be very large.
- The **remaining 10 test cases** will be **stress tests**, designed using real-world benchmarks (with slight modifications to fit the problem constraints). These will test your solution's ability to handle **large**, **sparse**, **dense**, and **different types of graphs**.

5. Formula used to calculate points for each test case -

   If no evacuees reach any shelter, the score will be **0**.

   Otherwise, the score is calculated as:

$$\text{score} = 1 \times \left(\frac{P - P_{\min} + \alpha}{P_{\max} - P_{\min} + \alpha} \right) + 0.4 \times \left(\frac{T_{\max} - T + \beta}{T_{\max} - T_{\min} + \beta} \right)$$

   T_{min} is the shortest evacuation time among all submissions.

   T_{max} is the longest evacuation time among all submissions.

   P_{min} represents the lowest number of evacuated individuals among all submissions.

   P_{max} represents the highest number of evacuated individuals among all submissions.

   Alpha and Beta are some constants decided by the evaluator.

   Evacuated People (P) and Evacuation time (T) are from your submission.

   This raw score will be normalized between 0 and 1 using the below given formula -

$$S' = \frac{S}{S_{\max}}$$

   S is the raw score of your submission
   S_{max} is the maximum raw score among all the submissions.

Key Considerations

- The validity of each solution will be assessed using our simulation script, as outlined in the grading criteria.
- The measured time refers to the simulated evacuation time based on the movement of people, not the execution time of your program.
- Multiple correct solutions may exist, as different strategies can lead to successful evacuations.
- Plagiarism is strictly prohibited. Given the nature of this problem, scores are expected to vary. If identical scores are detected, a detailed review will be conducted to ensure originality.
- A populated city can also serve as a shelter. In such cases, you have to ensure the constraints like shelter capacity are respected.

We trust that you will strategize and lead the evacuation with the precision and courage worthy of Commander Erwin himself. **Tatakae!**

Bonus marks

We will be awarding a bonus of **2 marks** for this assignment as follows:

- First, only students whose submissions receive a **non-zero score on all test cases** will be considered for bonus marks.
- For each eligible student, we will calculate their **average score across all test cases**.
- Once we have the average scores of all eligible students, we will compute the **overall average** of these individual averages.
- If a student's individual average score is **more than (**overall average + 0.674 × σ), they will receive bonus marks. σ is standard deviation.

In simple terms:
Suppose there are 10 students and 20 test cases.
Each student's average score across the 20 test cases will be calculated (say A1, A2, ..., A10).
Let M be the average of these values.
Students whose average Ai satisfies $A_i \geq M + 0.674 \times \sigma$ will be awarded bonus marks,
where σ is the standard deviation of A1, A2, ..., A10.

Q & A

I will write all the answers in this document so that both the questions and answers stay in one place and are easily accessible to everyone.

1) It will be really helpful if any materials/pointers can be provided, which I can look/read upon to get an idea and an understanding of how to usually go about solving such problems, so that I can proceed further ahead into the assignment.

Ans -
Unfortunately, we don't have any material to share. However, if you are stuck and unsure about how to start – please follow these steps

A. First, implement a basic simulation of the problem. Don't worry about optimizations yet — a simple brute-force simulation that respects all the program's constraints is a good starting point.
B. Once you have a working brute-force solution, use your learnings from Assignment 3 to apply either a variation of MST or shortest path algorithms. Focus first on minimizing the time or distance to the shelter. Ignore wait times due to road contention for now.
C. After completing step B, explore alternative approaches that handle road contention more efficiently — for example, by taking alternate paths, waiting to use the same path, or using other strategies.
D. Then, aim to maximize the number of people saved. Initially, don't worry about the simulated time.
E. If you manage to build a solution that addresses all the above points, you can further experiment with different heuristics and approaches to improve the solution.

The main message that we want you to take away from this is to build incrementally. Start very simple, and then optimize on top of it.

2) When a group of people arrive at a city and are destined to take a route, is it necessary they should immediately use the road when it is free, or am I allowed to stall them and move them at a later point in time.

Ans -
You cannot stall them. We expect that the people will immediately start using the next path(if available). It is possible that you have 3 paths A, B, C and path A, C is free but path B is being used currently. If you decide to move along path B then you will have to wait for path B to become usable again , but if you decide to take path A or C, then you must not wait and immediately use it if it is free.

3) Say, Population A travels from X -> Y from time T= 10 till T = 20
Population B (10 persons) arrives at X at T = 15
Population C (100 persons) arrives at X at T = 16
Both B and C want to continue to Y. Who is given priority?

Ans -
It depends on how many people from B and C, you want to move along the path. Let's assume you don't want to drop anyone at X, then population B and C will have to wait till T = 20 and then to resolve contention we will consider the city whose "moving" population is more. Since we don't plan to drop anyone , population of C would be higher than B and hence C must use the road.

4)Give an example for the case where we are applying the penalty when no of people dropped in the shelter exceed the capacity.

Ans-
This is completely based on your strategy to drop people. One possible case could be - Imagine A <--> B <--> C, B is a shelter with capacity 10. A has 15 old people and you moved them from A to B. After reaching B, if old people have almost traveled max_distance_elderly and cannot travel any further(or go back to A) then they have to be dropped compulsorily at shelter B. In such case, you will have to drop 15 old people in a shelter with capacity 10. So you have saved 10 people but due to additional 5 people , you will incur a penalty of 5 and hence, only 5 people(10 - 5 = 5) will be considered as "saved".

5)Can we drop the elderly in the initial starting city when they are not able to reach the shelter.

Ans.
Yes, you can drop them anywhere, they won't be considered as "saved" though.

6)Doubt regarding A4: In the example graph input (as given in the assignment: we have already filled the shelter 1 (all from city 5 – as we did in the first step); why do we again fill them with 200 primeAge people from city 0? – as mentioned in point 4 of priority considerations so do we take into account the priority conditions or not while traversal ? (because I found these 2 points contradictory while I was simulating as per your written points)

Ans.
We filled shelter 1 completely with city 5 population. But we are not dropping anyone from city 0 at shelter 1. I might be misunderstanding the question. If you are thinking that people traversing via shelter 1 are filling the city, then that is not the case -- The capacity is only for "drops"and there is no limit on the number of people who can arrive and depart from the city (traversal)

7) in the assignment 4 the edge contention is based on timing in which case assume population from one city finish moving in 100.5 minutes then can i make my population who was waiting to take that edge at 101 or it will be assumed that the edge was taken at

100.5 or simply can i discritize the time into minutes(int or long or similar) or should I handle it as a continuous value(float or double)

Ans.
I don't think you'll get a float value like 100.5 minutes. Since everyone's speed is fixed at 5 km/h, it means 1 km takes exactly 12 minutes. Also, the length is always a positive integer (long long) in the starter code. Please let me know if I've misunderstood anything.

8) Sir in the A4 description, it is mentioned that the first 30 test cases will not be large. Can you specify a hard upper limit ? I know we can have 100,00 vertices at max with 1,00,000 edges but I assume this is for a stress test (the last 10 test cases) .

Ans -
15 test cases - max 10 populated cities, max 100 vertices, max 1000 edges
15 test cases - max 100 populated cities, max 1000 vertices, max 10000 edges
10 test cases - max constraints given in the assignment (stress tests)
Total - 40 test cases.

9) In the example in the project description , google doc when the closest shelter is too far for the elderly they were dropped off at the next city, can I drop them off at the original city (I mean never move them at all) ??

Ans -
Yes, you are allowed to drop people at the starting city as well.
However, keep in mind: if your entire population consists only of old people and you plan to drop all of them at the starting city itself, your path must still include the starting vertex.
In other words, **the path cannot be empty** — it must contain at least one vertex.

10) For the case of penalty , if the total drop exceeds the capacity of a shelter , then the penalty is applied , but like what % on the prime_age , and what % on the elderly , or depends on the sequence of drop?

Ans -
The number of prime and old people does not affect the penalty calculation. The penalty is determined solely based on the number of people exceeding a shelter's capacity. Specifically, at the end of the simulation, if a shelter has a capacity of 100 but you have dropped 140 people there, the number of people saved at that shelter is adjusted by subtracting the excess population. In this case: people saved = 100 - 40 = 60 people will be considered as "saved". Notice that the number of people exceeding the shelter's capacity (40) is subtracted from the shelter's full capacity (100) to compute the actual number of people saved.
Also note: the penalty will never exceed the shelter's capacity itself.

11) "It would be good to know what the evaluation script looks like (in terms of input and output)."
Ans -

The evaluation script will take your output — specifically the **paths** and **drop locations** — as input. It will then produce two results: the **simulation time** and the **number of people saved**.

During evaluation, the script will perform the following checks:

1. **Population check**: Ensure that the total number of people dropped matches the total population across all cities (i.e., no one is left behind or unaccounted for).
2. **Path validity**:
    - Each path must contain at least one vertex (i.e., the path must not be empty).
    - The path must be valid — meaning edges must exist between every pair of consecutive vertices in the path.
3. **Overflow penalty**: Apply a penalty if a shelter exceeds its capacity (i.e., if too many people are dropped at a shelter).