T2612 Mini Metro II

Wong Chun {WongChun1234}

2026-05-23

Table of Contents

Background

Problem Idea by `QwertyPi`
Preparation by `__jk__`, `bedrockfake`
Special Thanks to `hywong1`, `WongChun1234` for testing!

(easiest problem of the day 1's set)

The Problem

Given a circular metro line with N stations (0 to $N-1$).

- Station 0 must be **Triangular** (Δ).
- Station $N-1$ must be **Square** ($\square$).
- Remaining $N-2$ stations can be freely classified as Δ or $\square$.

Passengers starting at station i:

- $T[i]$ want to go to the nearest Δ station in clockwise order.
- $S[i]$ want to go to the nearest $\square$ station in clockwise order.

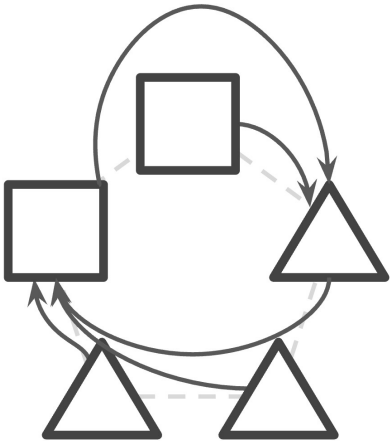
Goal: Minimize the total travel time of all passengers.

Example

An example of $N = 5$:

Simplifying the Visuals

We omit arrows representing passengers whose starting station type **already matches** their target destination type, as they reach their destination instantly with a travel time of 0 minutes.



Subtasks

#	Max	N	Constraints
1	5	$= 3$	No additional constraints
2	11	$\leq 5 \times 10^5$	$S[i] = T[i]$ for $0 \leq i \leq N - 1$
3	10	≤ 500	$\texttt{min}(T[i], S[i]) \leq 2$, $\texttt{max}(T[i], S[i]) = 10^6$
4	22	≤ 500	$T[i], S[i] \leq 500$
5	27	≤ 3000	No additional constraints
6	25	$\leq 5 \times 10^5$	No additional constraints

Statistics

#	Score	Predict #	Actual #
1	5	55	78
2	11	55	64
3	10	45	63
4	22	25	24
5	27	22	22
6	25	10	9

Table of Contents

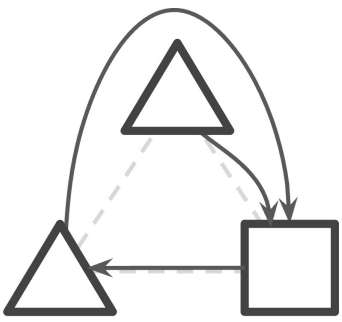
Subtask 1 (5%): $N = 3$

Recall that station 0 is fixed as Δ and station 2 is fixed as $\square$.
We only need to decide the type of station 1.

Case 1: Station 1 is Δ

- $S[0]$ passengers travel $0 \to 2$ (dist = 2)
- $S[1]$ passengers travel $1 \to 2$ (dist = 1)
- $T[2]$ passengers travel $2 \to 0$ (dist = 1)

$\texttt{cost} = 2 \cdot S[0] + S[1] + T[2]$



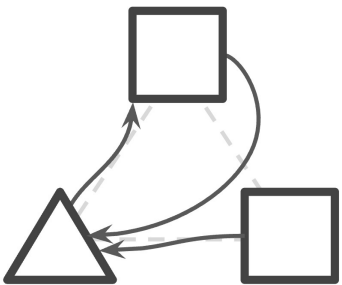
Subtask 1 (5%): $N = 3$

Recall that station 0 is fixed as Δ and station 2 is fixed as $\square$.
We only need to decide the type of station 1.

Case 2: Station 1 is $\square$

- $S[0]$ passengers travel $0 \to 1$ (dist = 1)
- $T[1]$ passengers travel $1 \to 0$ (dist = 2)
- $T[2]$ passengers travel $2 \to 0$ (dist = 1)

$\texttt{cost} = S[0] + 2 \cdot T[1] + T[2]$



Subtask 1 (5%): $N = 3$

Taking the minimum of both costs:

$$\texttt{cost} = \texttt{min}(2 \cdot S[0] + S[1] + T[2],\ S[0] + 2 \cdot T[1] + T[2])$$

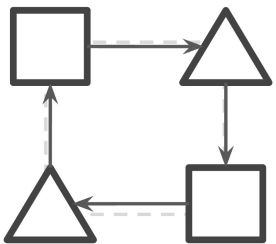
$$= S[0] + T[2] + \texttt{min}(2 \cdot T[1],\ S[0] + S[1])$$

Time complexity: $\mathcal{O}(1)$
Expected Score: 5 (Cumulative: 5)

Subtask 2 (11%): $T[i] = S[i]$ for $0 \leq i \leq N-1$

Observation

When $T[i] = S[i]$, the optimal construction is to alternate station types $(\Delta, \square, \Delta, \square \ldots)$ as strictly as possible.



Subtask 2 (11%): $T[i] = S[i]$ for $0 \leq i \leq N-1$

Observation

When $T[i] = S[i]$, the optimal construction is to alternate station types $(\Delta, \square, \Delta, \square \ldots)$ as strictly as possible.

Proof Idea:

- For passengers at station i (WLOG assume its type is Δ):
  - Passengers of the *matching* type (Δ) travel 0 distance.
  - Passengers of the *opposing* type ($\square$) must board the metro, and the earliest they can alight is at the next station ($i+1$) if its type is $\square$.
- Hence, the theoretical minimum total cost for any valid configuration is:

$$\sum_{i=0}^{N-1} \texttt{min}(S[i], T[i])$$

Subtask 2 (11%): $T[i] = S[i]$ for $0 \leq i \leq N-1$

Observation

When $T[i] = S[i]$, the optimal construction is to alternate station types $(\Delta, \square, \Delta, \square \ldots)$ as strictly as possible.

Proof Idea:

- Since $T[i] = S[i]$, when N is even, alternating stations can achieve the lower bound $\sum_{i=0}^{N-1} \texttt{min}(S[i], T[i]) = \sum_{i=0}^{N-1} S[i]$.
- When N is odd, there must exist adjacent stations of the same type (e.g. $(i, i+1)$), if we iterate through the possible pairs $(i, i+1)$, the remaining stations can be alternated to achieve the lower bound when N is even.
- Therefore when N is odd, the final cost is the additional cost $\texttt{min}(S[i], T[i])$ from station i added to the lower bound.

Subtask 2 (11%): $T[i] = S[i]$ for $0 \leq i \leq N-1$

Considering the parity of N:

- N is even: $\texttt{cost} = \sum_{i=0}^{N-1} S[i]$
- N is odd: $\texttt{cost} = \sum_{i=0}^{N-1} S[i] + \min_{1 \leq i \leq N-2} S[i]$

Time complexity: $\mathcal{O}(N)$
Expected Score: 11 (Cumulative: 16)

Subtask 3: $N \leq 500$, $\texttt{min}(T[i], S[i]) \leq 2$, $\texttt{max}(T[i], S[i]) = 10^6$

Additional constraint:

- $T[0] = S[0] = T[N-1] = S[N-1] = 0$

The difference between $S[i]$ and $T[i]$ is massive.

Intuitively, we would want to match the type of station i to the type with 10^6 passengers.

But can we prove that this is optimal?

Subtask 3 (10%): $N \leq 500$, $\texttt{min}(T[i], S[i]) \leq 2$, $\texttt{max}(T[i], S[i]) = 10^6$

- Consider each passenger i, they will travel at most $N-1$ stations (as it is guaranteed at least one station of each type exists).
- If we match the type of station i to the type with 10^6 passengers:

$$\begin{aligned}\text{cost} &< \sum_{i=0}^{N-1} \texttt{min}(S[i], T[i]) \cdot (N-1) \\ &\leq \sum_{i=0}^{N-1} 2 \cdot (N-1) < 2N^2 \leq 5 \times 10^5 < 10^6\end{aligned}$$

So matching the type of station to that of the type of 10^6 passengers is always optimal! (for $1 \leq i \leq N-2$)

Subtask 3: $N \leq 500$, $\texttt{min}(T[i], S[i]) \leq 2$, $\texttt{max}(T[i], S[i]) = 10^6$

Strategy: Greedily assign types based on whichever is 10^6, then easily simulate the passengers of the remaining type (≤ 2) to find the total minimum cost.

- Can you think of a case where the solution fails if the constraint $T[0] = S[0] = T[N-1] = S[N-1] = 0$ was not satisfied?

Time complexity: $\mathcal{O}(N)$ or $\mathcal{O}(N^2)$
Expected Score: 10 (Cumulative: 26)

Table of Contents

Passenger Cost Contributions

Suppose station i is Δ, so only Δ passengers will end at i.

Observation

Without loss of generality, assume station i is Δ. The number of passengers who stop at station i is equal to the $\sum_{k=j}^{i-1} T[k]$ where station $j-1$ is the nearest Δ station counter-clockwise to station i.

- **Therefore,** the total cost of a configuration is the sum of contributions of passengers ending at station i.
- We can define a cost function $C(j,i)$ as the travel time incurred by all intermediate passengers when $j-1$ and i are stations of type X, and all stations between them are of type Y.

Formulating the DP

Calculating Segment Cost $C_{\Delta}(j, i)$

For all intermediate stations k ($j \leq k < i$), their type is forced to be $\square$:

- $T[k]$ passengers ($\square \to \Delta$) travel clockwise to i, covering $(i - k)$ distance.

$$C_{\Delta}(j, i) = \sum_{k=j}^{i-1} \Big(T[k] \cdot (i - k) \Big)$$

Formulating the DP

Let `dp[i][0]` be the minimum cost for prefix $0 \dots i$ where station i is Δ.
Let `dp[i][1]` be the minimum cost for prefix $0 \dots i$ where station i is $\square$.

Transitions

WLOG station i is Δ, let $j-1$ be the **immediate previous** Δ station. All stations from j to $i-1$ must therefore be $\square$. Consider transitioning from the first station of consecutive types, e.g. $j \to i$:

$$\underset{j-1}{\Delta}, \underbrace{\square, \cdots, \square}_{[j,i-1]}, \underset{i}{\Delta}, \cdots, \Delta$$

$$\texttt{dp[i][0]} = \min_{0 \le j < i} \{\texttt{dp[j][1]} + C_{\Delta}(j,i)\}$$

$$\texttt{dp[i][1]} = \min_{0 \le j < i} \{\texttt{dp[j][0]} + C_{\square}(j,i)\}$$

Subtask 4 (22%): $N \leq 500, T[i], S[i] \leq 500$

The final consecutive stations must be $\square$, so $\texttt{cost} = \texttt{min}(\texttt{dp[i][1]} + C_{\Delta}(i, 0))$ to deal with the wrap-around.

There are $2N$ states, each state has $\mathcal{O}(N)$ transition states, and the transition takes $\mathcal{O}(N)$ to compute C_{Δ} (and $C_{\square}$).

(There exists alternative $\mathcal{O}(N^3)$ DP solutions but for the sake of convenience, we choose this one to present)

Time complexity: $\mathcal{O}(N^3)$
Expected Score: 37 (Cumulative: 48)

Subtask 5 (27%): $N \leq 3000$

The final consecutive stations must be $\square$, so $\texttt{cost} = \texttt{min}(\texttt{dp[i][1]} + C_{\Delta}(i, 0))$ to deal with the wrap-around.

Precomputing $C(j, i)$ for all pairs takes $\mathcal{O}(N^2)$ time.
There are $2N$ DP states, each state has $\mathcal{O}(N)$ transition states, and the transition takes $\mathcal{O}(1)$.

Time complexity: $\mathcal{O}(N^2)$
Expected Score: 64 (Cumulative: 75)

Table of Contents

Analyzing the DP transition

$$\texttt{dp[i][0]} = \min_{0 \le j < i} \{\texttt{dp[j][1]} + C_{\Delta}(j, i)\}$$
$$\texttt{dp[i][1]} = \min_{0 \le j < i} \{\texttt{dp[j][0]} + C_{\square}(j, i)\}$$

$$\texttt{dp[i][0]} = \min_{0 \le j < i} \left\{ \texttt{dp[j][1]} + \sum_{k=j}^{i-1} \Big(T[k] \cdot (i - k) \Big) \right\}$$
$$\texttt{dp[i][1]} = \min_{0 \le j < i} \left\{ \texttt{dp[j][0]} + \sum_{k=j}^{i-1} \Big(S[k] \cdot (i - k) \Big) \right\}$$

Analyzing the DP transition

$$\texttt{dp[i][0]} = \min_{0 \le j < i} \left\{ \texttt{dp[j][1]} + \sum_{k=j}^{i-1} \left(i \cdot T[k] - k \cdot T[k] \right) \right\}$$

$$\texttt{dp[i][1]} = \min_{0 \le j < i} \left\{ \texttt{dp[j][0]} + \sum_{k=j}^{i-1} \left(i \cdot S[k] - k \cdot S[k] \right) \right\}$$

Analyzing the DP transition

Denote $\texttt{ps}_X[i]$ to be the partial sum of type X passenger in $[0 \dots i]$, and $\texttt{ps'}_X[i] = \sum_{k=0}^{i} k \cdot X[k]$.

$$\texttt{dp[i][0]} = \min_{0 \le j < i} \left\{ \texttt{dp[j][1]} + i \cdot \texttt{ps}_\Delta[i-1] - i \cdot \texttt{ps}_\Delta[j-1] - \texttt{ps}'_\Delta[i-1] + \texttt{ps}'_\Delta[j-1] \right\}$$

$$\texttt{dp[i][1]} = \min_{0 \le j < i} \left\{ \texttt{dp[j][0]} + i \cdot \texttt{ps}_\square[i-1] - i \cdot \texttt{ps}_\square[j-1] - \texttt{ps}'_\square[i-1] + \texttt{ps}'_\square[j-1] \right\}$$

$$f_X(i) = i \cdot \texttt{ps}_X[i-1] - \texttt{ps}'_X[i-1]$$
$$g_X(j) = dp[j][X'] + \texttt{ps}'_X[j-1]$$
$$h_X(i,j) = -i \cdot \texttt{ps}_\Delta[j-1]$$

Analyzing the DP transition

Denote $\texttt{ps}_X[i]$ to be the partial sum of type X passenger in $[0 \dots i]$, and $\texttt{ps'}_X[i] = \sum_{k=0}^{i} k \cdot X[k]$.

$$\texttt{dp[i][0]} = f_\Delta(i) + g_\Delta(j) + h_\Delta(i,j)$$
$$\texttt{dp[i][1]} = f_\square(i) + g_\square(j) + h_\square(i,j)$$

$$h_X(i,j) = -i \cdot \texttt{ps}_\Delta[j-1]$$

Familiar? We can apply convex hull trick! :)

Convex Hull Trick

$$\texttt{dp[i][0]} = -\texttt{ps}_{\triangle}[j-1] \cdot i + g_{\triangle}(j) + f_{\triangle}(i)$$
$$\texttt{dp[i][1]} = -\texttt{ps}_{\square}[j-1] \cdot i + g_{\square}(j) + f_{\square}(i)$$

$$m_X = -\texttt{ps}_X[j-1]$$
$$x = i$$
$$c_X = g_X(j)$$

Convex Hull Trick

$$\texttt{dp[i][0]} = (m_{\triangle} x + c_{\triangle}) + f_{\triangle}(i)$$
$$\texttt{dp[i][1]} = (m_{\square} x + c_{\square}) + f_{\square}(i)$$

As $m_X = -\texttt{ps}_X[j-1]$ is decreasing and we query the minimum value, we can use the stack implementation of CHT.

Implementation (adopted from training slides)

```
1. while (Q.size() >= 2 && Q[0].eval(i) > Q[1].eval(i)) {
       Q.pop_front();
   }
2. if (!Q.empty()) {
       dp[i] = Q.front().eval(i) + f(i);
   }
3. line cur = {m(i), c(i)}; // slope and intercept come from j-terms
   while (Q.size() >= 2 && Q.back().intersectX(cur) <=
          Q[Q.size()-2].intersectX(Q.back())) {
       Q.pop_back();
   }
4. Q.push_back(cur);
```

Subtask 6 (25%): No additional constraints

Maintain CHT and handle the final $\square$ stations as mentioned in Subtask 4 and 5.

Time complexity: $\mathcal{O}(N)$
Expected Score: 100 (Cumulative: 100)

Table of Contents

Takeaways

- For DP problems, don't be scared if your initial approach is very inefficient. It is possible to improve one step at a time.
- Try to observe/prove properties in your DP in order to apply known optimization techniques