Task 1: Competition (`competition`)

Authored and prepared by: Sidhant Bansal

Subtask 1

Limits: $1 \leq n \leq 20$

We try all $\binom{n}{a}$ valid partitions of the students, and take the maximum score which can be attained.

Time complexity: $O(2^n n)$

Subtask 2

Limits: $1 \leq n \leq 10^5$, $B[i] = 0 \forall i \in [1, n]$

Since all the $B[i]$ are zero, the sum of Biology scores of the b Biology students is always zero. Hence, the problem reduces to finding the a students with largest Physics scores.

This may be implemented efficiently by sorting the students by $A[i]$.

Time complexity: $O(n \log n)$

Subtask 3

Limits: $1 \leq n \leq 10^5$

We claim that the following algorithm works: Sort all the students by $A[i] - B[i]$ in ascending order. Take the first b students for Biology and the last $a = n - b$ students for Physics.

Proof. Let the sorted array be denoted by $A_1, A_2, \ldots, A_n$ and $B_1, B_2, \ldots, B_n$, i.e. $A_i - B_i \leq A_{i+1} - B_{i+1}, \forall i \in [1, n-1]$

Let $X = \{b+1, b+2, \ldots, n\}$.

Let the solution found by the above algorithm be denoted by $S(X) = \sum_{i \in X} A_i + \sum_{i \notin X} B_i$.

Let there be an optimal solution which is $S(Y)$ where $S(Y) = \sum_{i \in Y} A_i + \sum_{i \notin Y} B_i$, where Y denotes the set of elements for which we would have picked Physics in this optimal solution. Therefore $|Y| = a$.

Let us assume that $X \neq Y$, then $Y - X \neq \phi$ and $X - Y \neq \phi$ which means $\exists \alpha \in Y$ such that $\alpha \notin X$ and $\exists \beta \in X$ such that $\beta \notin Y$.

Let $Y' = Y - \{\alpha\} + \{\beta\}$, then $S(Y') = \sum_{i \in Y'} A_i + \sum_{i \notin Y'} B_i = S(Y) - A_\alpha + A_\beta - B_\beta + B_\alpha = S(Y) + (A_\beta - B_\beta) - (A_\alpha - B_\alpha)$

Notice that $\alpha \notin X$ and $\beta \in X$, where $X = \{b+1, b+2, \ldots, n\}$ implies that $\alpha < \beta$, whic means $A_\beta - B_\beta \geq A_\alpha - B_\alpha$, since the arrays are sorted by the difference.

Therefore, $(A_\beta - B_\beta) - (A_\alpha - B_\alpha) \geq 0$ which implies $S(Y') \geq S(Y)$.

Also notice that $|X - Y'| < |X - Y|$, i.e. Y' is closer to X.

By induction, we can keep on doing this and make $Y'' = X$ after some finite number of swaps and still preserve $S(Y) \leq S(Y') \leq S(Y'') = S(X)$, therefore $S(X)$ is at least as good as the optimal.

□

Time complexity: $O(n \log n)$

Task 2: Departure (`departure`)

Authored and prepared by: Bernard Teo Zhi Yi

Preliminaries

We let D be the maximum number of days each person might require to reach their destination, and P be the maximum number of buses passing through any given location.

Subtask 1

Limits: $1 \leq N \leq 10^4$, $1 \leq M \leq 10^3$, $\text{sign}(S_i) \neq \text{sign}(E_i)$ for all $1 \leq i \leq N$

The constraint $\text{sign}(S_i) \neq \text{sign}(E_i)$ implies that all buses may be boarded from the stadium.

To solve this subtask, we need to make one simple observation: The fastest way to reach any location is attainable using a direct bus (i.e. no transfers). To see why, consider a given location, x; without loss of generality, we assume that $x > 0$. The earliest way to reach x cannot reach earlier than the time that the first rightward-heading bus arrives at x on the first day. This is because a route that arrives at x using a bus that is heading leftward necessarily passes through x using a rightward-heading bus before using the final leftward-heading bus. Now since every bus must also pass the stadium and rightward-heading buses passing x must pass the stadium before reaching x, a person living at x may simply wait at the stadium for this bus, and then taking it directly to x.

It hence suffices, for each person, to iterate through all the buses, computing the time they will reach their destination if they use that bus, and then taking the minimum over all buses.

Time complexity: $O(NM)$

Subtask 2

Limits: $1 \leq N \leq 10^2$, $1 \leq M \leq 10^3$

We make a more general observation: At every decision point, it is optimal to move as fast as possible in the direction of the destination. In other words, whenever there is a bus at a person's current location and heading in the correct direction, they should get on the bus; if there are multiple buses heading in the correct direction, they should get on the fastest bus; if there are buses headed in the wrong direction, they should remain on the roadside. This may be seen by noting that for any locations x, y, and z such that $x < y < z$ or $x > y > z$, a person at location x whose destination is at location z must pass through location y first, and hence a person already at location y headed to location z must be able to arrive at location z no later than the person at location x can do the same.

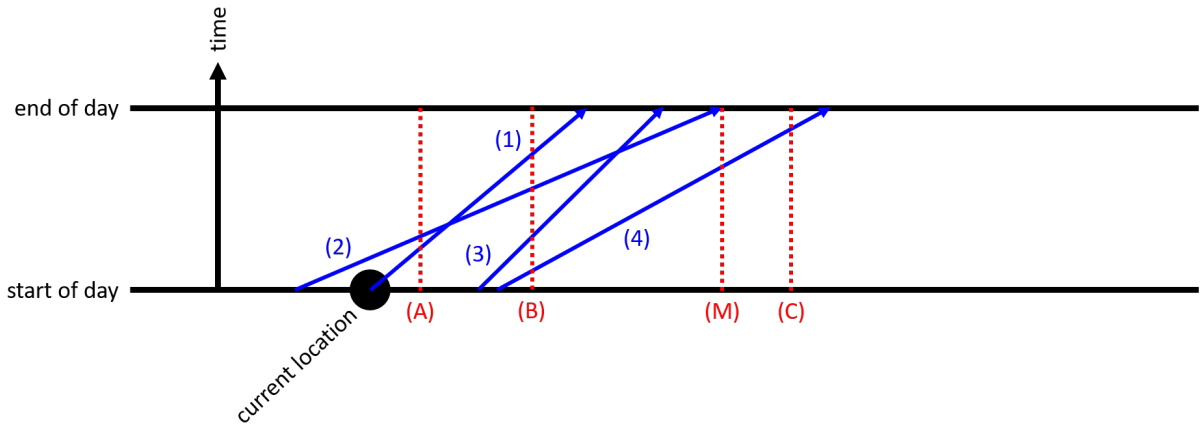
This observation leads to a greedy solution: A person always get onto the fastest bus heading in the correct direction, transferring buses whenever another bus is about to overtake the bus that they are currently on.

As such, we partition the buses and people into two partitions — those buses travelling rightward, together with the people whose destinations are at positive locations; and those buses travelling leftward, together with the people whose destinations at at negative locations. Each of these two parts can be solved separately, so from here onward we assume that destinations are always positive and buses only travel rightward.

This greedy solution implemented exactly as described is unlikely to pass this subtask, and is rather complicated to implement because we need to quickly find the set of locations where each bus will be overtaken by another bus.

We make the following modification to the greedy solution: At the start of each day, each person checks if they can get from the current location to their destination using a direct bus. If so, then they should directly take the bus that arrives at their destination earliest. Otherwise, they should take the bus whose end location is closest to their destination (i.e. furthest to the right), and get off only when the bus reaches its end location.

Proving that this modification retains optimality is best done with the help of an illustration:



The x-axis represents locations along the road, and the y-axis represents time. The blue lines represent buses travelling from some start location at the start of the day, and arriving at some end location at the end of the day. They are straight lines because they travel at a constant speed throughout the day.

Suppose a person is currently at the location marked with the large black circle. If their destination can be reached on the same day (e.g. (A) or (B)), then we need only to show two claim: firstly, the bus b that arrives at the destination earliest and which also passes the current location, is a direct bus from the current location to the destination; secondly, it is not possible to arrive at the destination (on the same day) before bus b arrives at the destination.

The first claim is clear — they can simply wait at the current location for bus b. To prove the second claim, observe that every bus that arrives at the destination earlier than bus b (e.g. (3)

or (4), for destination (B)) must not also pass the current location, due to how we picked bus b. Let h be any bus that arrives at the destination earlier than bus b. It must not have been possible to take a sequence buses from the current location to somewhere and make a same-day transfer to bus h, because if that is possible, there must be a bus that passes the current location and yet arrive at the destination earlier than bus h arrives there, which contradicts our assumption that bus b is the earliest such bus.

Now, we show the latter modification to the greedy solution. For each person that does not have a direct bus from the current location to their destination, the original greedy solution brings them as far as possible in the rightward direction (e.g. (M)) (which brings them as close as possible to their destination), say to location M. Then we apply a similar argument as previously done, treating M as a destination, and therefore showing that the fastest way to reach M from the current location is via a direct bus.

We have now proved that the modified greedy solution is optimal as well, and it may be implemented by considering each person separately — for each person, we simulate their movement by picking the ideal bus each day until they reach their destination.

Time complexity: $O(NMD)$

Subtask 3

Limits: $1 \leq N \leq 10^3$, $1 \leq M \leq 10^5$, $D \leq 10^3$, $P \leq 10^2$

For each person j, let the k_j be the number of full days required for the journey, i.e. $k_j = \left\lfloor \frac{A_j}{B_j} \right\rfloor$, and let $M_{j,0}, M_{j,1}, M_{j,2}, \ldots, M_{j,k_j}$ be the sequence of locations in which person j spends their nights (where $M_{j,0} = 0$, and $M_{j,a}$ is the location in which the person j spends the a^{th} night). Notice that for any two people j_1 and j_2, we have $M_{j_1,a} = M_{j_2,a}$ for all $0 \leq a \leq \min\{k_{j_1}, k_{j_2}\}$ (remember that we are only considering those people who live at positive locations).

We can now speed up the modified greedy solution by simulating all people in parallel, since everybody who has not yet reached their destinations must share the same location each night. We first sort all the people in increasing order of their destinations. Then, on each day, we find the rightmost location that can be reached by a direct bus; every person whose destination is not beyond this location finds their direct bus home, and the remaining people are automatically moved to the rightmost location by updating some common variable. Sorting the people once takes $O(M \log M)$ time, finding the rightmost location reachable by a direct bus takes $O(ND)$ (since we do it once per day), and determining the bus for the final leg of each person's journey takes $O(MP)$ (since we need to check only at most P buses that pass by each location).

Time complexity: $O(M \log M + ND + MP)$

Subtask 4

Limits: $1 \leq N \leq 10^6$, $1 \leq M \leq 10^3$, $P \leq 10^4$

We make an enhancement to the algorithm in subtask 3 to reduce the $O(ND)$ term. Observe that if at the start of the day the people are at some location x, all buses that they could possibly board must have $S_i \leq x$. Furthermore, if say the people were at location y at the start of the previous day (before they came to location x), then every bus with $S_i \leq y$ cannot be "useful" to the people at location x (in the sense that the optimal solution will never board any such bus at or to the right of location x). This is because we would have saved a day if we boarded the bus from location y instead. Hence, we only need to consider those buses where $y < S_i \leq x$.

Thus, we sort all the buses by S_i once before the rest of the simulation, and on each day, we process only the buses that were not accessible on the previous day but which are accessible now (i.e. $y < S_i \leq x$). Notice that this means each bus is only processed on a single day.

Time complexity: $O(N \log N + M \log M + MP)$

Subtask 5

Limits: $1 \leq N \leq 10^6$, $1 \leq M \leq 10^4$, $D \leq 10^2$

This subtask builds on the modified greedy solution in subtask 2.

For each person, on each day, we need to determine the bus that gets to their destination earliest (if their destination is reachable by a direct bus), or the furthest reachable location (if their destination is not reachable by a direct bus). Notice that queries of this form may be answered efficiently using the convex hull trick — we model the buses as lines between the coordinates $(S_i, 0)$ and $(E_i, 1)$, and each query asks for the smallest x-value attainable for a given y-value. We sort the buses by S_i ($O(N \log N)$), similar to subtask 4, to determine the buses that we should consider on each day, and then put them into a convex hull data structure ($O(N)$, since the buses are already sorted). Queries on this data structure take $O(\log N)$ time per person per day, since we do a binary search on the convex hull.

Time complexity: $O(N \log N + MD \log N)$

Subtask 6

Limits: $1 \leq N \leq 10^6$, $1 \leq M \leq 10^6$, $\text{sign}(S_i) \neq \text{sign}(E_i)$ for all $1 \leq i \leq N$

This subtask is for solutions that recognise the convex hull, but did not realise that each bus can only be "useful" on at most one day. For this subtask, only a single convex hull is needed as all people are able to reach their destinations using a direct bus.

Time complexity: $O(N \log N + M \log N)$ or $O(N \log N + M \log M)$

Subtask 7

Limits: $1 \leq N \leq 10^6$, $1 \leq M \leq 10^6$

This final subtask combines the solution from subtask 4 with the convex hull trick. We sort the buses by S_i and sort the people by their destinations, and on each day we build a convex hull from the "useful" buses of that day.

Time complexity: $O(N \log N + M \log M)$

Task 3: Truck (`truck`)

Authored and prepared by: Teow Hua Jun

Subtask 1

Limits:$V_i = 1, T_i = 0$, Each town has at most two roads connected to it

For the first few subtasks, $V_i = 1$ for all $1 \leq i \leq Q$, which means that there are no updates on the toll of the edges. All the requests are queries. Also, the condition of each town being connected to at most 2 other towns means that the tree is a line graph. As any other form of tree would have a town connected to 3 or more towns.

As each edge does not have any toll cost, the truck does not need to carry more than G gold, and to minimise the fuel consumed the optimal path is straight to the destination node which results in the minimum distance.

The answer of each query is thus $G \times dist(A_i, B_i)$. Where $dist(A_i, B_i)$ is the distance between node A_i and B_i.

This can be calculated with a prefix sum in $O(N)$ precomputation and $O(1)$ query time.

Time complexity: $O(N + Q)$

Subtask 2

Limits:$V_i = 1, T_i = 0$

Since there are no updates and $T_i = 0$, the solution is similar to subtask 1.

The answer for each query is still $G \times dist(A_i, B_i)$.

However, calculating $dist(A_i, B_i)$ is a bit harder as the graph is not a line but instead a general tree. To calculate the distance, we can first root the tree arbitrarily at node 1 and precompute the distance of each node to the root, let each node X's distance to the root be $d(X)$.

By using 2^k path decomposition on the tree we can find the lowest common ancestor (LCA) between any pair of nodes in $O(N \log N)$ precomputation and $O(\log N)$ query time.

Let C_i be the LCA of nodes A_i and B_i, the distance between the nodes can be calculated as $d(A_i) + d(B_i) - 2 \times d(C_i)$.

The answer is thus $G \times (d(A_i) + d(B_i) - 2 \times d(C_i))$.

Time complexity: $O((N + Q) \log N)$

Subtask 3

Limits:$V_i = 1$, $T_i = T_j$ for all $i \neq j$, $D_i = 1$

The solution to this subtask requires us to utilise the same method in subtask 2 to get the distance between A_i and B_i, let the distance from A_i to B_i be M. Due to the property of $D_i = 1$, we know that there are exactly M edges between A_i and B_i and each edge has the same toll cost. Let that toll value be X.

Since each gold bar contributes to the fuel consumed, we can calculate the answer by considering how much distance each gold bar travels before it is given away as payment for a toll.

Since, the G gold bars are not given away, the amount they contribute to the answer is GM. We just need to calculate the fuel consumed due to the gold bars given as payment.

Note that the gold bars given as the payment for the first edge consumes 0 amounts of fuel. The gold bars for the second edge consumes X amounts of fuel and the third edge consumes $2X$ amounts of fuel. Namely, the gold bars given as payment for the i^{th} edge consumes $(i-1)X$ units of fuel.

Thus the total fuel consumed is $GM + X + 2X + ... + (M-1)X = GM + \frac{(M-1)M}{2}X$.

We can calculate M for each query in $O(\log N)$ with $O(N \log N)$ precomputation while G and T are constant through out.

The final time complexity: $O((N+Q)\log N)$

Subtask 4

Limits:$V_i = 1$, Each town has at most two roads connected to it

In this subtask, there are no longer constraints on the distance and toll of each edge. However, the graph is a line. We can choose one of the nodes with a degree of 1, which means that it is one of the endpoints of the line graph, call this node U and let it be the leftmost node in the line graph. We can also precompute the distance of each node from U, let it be $d(x)$. Distances between two nodes, $dist(x, y)$, can then be calculated by $d(y) - d(x)$, where y is further from U than x.

For each node, let l_i be the toll of the edge on its left and r_i be the toll of the edge on its right.

For each query of A_i, B_i, there are two cases, A_i is closer to U than B_i, and A_i is further to U than B_i.

In the first case, using the same way of calculating in subtask 3:

$\sum$ (Number of gold bars) $\cdot$ (Distance gold bars travelled)

We calculate that the answer is

$G \cdot dist(A_i, B_i) + \sum dist(A_i, x) \cdot r_x$ where x are the nodes exclusively in between A_i and B_i.

$G \cdot dist(A_i, B_i) + \sum dist(A_i, x) \cdot r_x$

$= G \cdot dist(A_i, B_i) + \sum (d(x) - d(A_i)) \cdot r_x$

$= G \cdot dist(A_i, B_i) + \sum (d(x) \cdot r_x) - \sum d(A_i) \cdot r_x$

$= G \cdot dist(A_i, B_i) + \sum (d(x) \cdot r_x) - d(A_i) \cdot \sum r_x$

Thus, we can see that the answer requires the summation of a range of values that is only dependant on x. Hence, we can precompute all $d(x) \cdot r_x$ and r_x and use a prefix sum to quickly gain the range sum in $O(1)$. The answer can then be calculated in $O(1)$.

For the other case of A_i is further to U than B_i, using similar logic we can reduce the answer to be:

$G \cdot dist(A_i, B_i) + d(A_i) \cdot \sum l_x - \sum (d(x) \cdot l_x)$

A similar prefix sum would also allow the answer to be calculated in $O(1)$ time.

Time complexity: $O(N + Q)$

Subtask 5

Limits: Each town has at most two roads connected to it

In this subtask, it is the exactly the same as subtask 4 but now the tolls of each edge can be changed. As the answer to each query can still be reduced to the same formula, we need a way to quickly calculate $\sum (d(x) \cdot l_x)$, $\sum (d(x) \cdot r_x)$, $\sum r_x$ and $\sum l_x$, while supporting updates.

To do this we can use fenwick trees to support point changes and range queries to calculate the answer, thus each answer can be calculated in $O(\log N)$.

Time complexity: $O((N + Q) \log N)$

Subtask 6

Limits: $N, Q \leq 5000$

The limits of N and Q are small enough to allow the contestant to simply simulate the truck's journey to pass. For each query, we can first perform a depth-first search from the starting node to calculate the number of gold bars required to travel to the destination node. Then depth first search again with a counter that stores the amount of gold the truck is currently carrying and simulate the gold being paid as toll by subtracting from the counter and calculate the fuel consumed after travelling each edge and add it to a global answer.

Each depth first search have a time complexity of $O(N)$.

Final time complexity: $O(NQ)$

Subtask 7

Limits: No additional constraints

Each query is now a path on a tree, if we arbitrarily root the tree at node 1, we can divide each path from A_i to B_i into 2 parts. The path from A_i to C_i and the path from C_i to B_i, where C_i is the LCA of A_i and B_i.

Let p_x be the direct parent node of x and T_x be the toll of the edge that connects x and p_x. Also, let $d(x)$ be the distance from node 1 to x.

Let's first consider the amount of fuel consumed due to the gold paid in edges from the path A_i to C_i. Note that the gold bars paid at each edge consumers a total of $dist(A_i, x) \cdot T_x$, where x are nodes exclusively between A_i and C_i.

Thus as $dist(A_i, x) = d(A_i) - d(x)$, similar to subtask 5, we can reduce the total fuel consumed by gold paid on the (A_i, C_i) path to be:

$\sum dist(A_i, x) \cdot T_x = d(A_i) \cdot (\sum T_x) - \sum(d(x) \cdot T_x)$

Similarly, we can find that the total fuel consumed by gold paid on the (C_i, B_i) path is:

$\sum(dist(A_i, B_i) - dist(B_i, p_x)) \cdot T_x$

$= dist(A_i, B_i) \cdot \sum T_x - \sum(d(B_i) - d(p_x)) \cdot T_x$

$= dist(A_i, B_i) \cdot \sum T_x - d(B_i) \cdot \sum T_x + \sum(d(p_x) \cdot T_x)$

Since, $dist(A_i, B_i), d(B_i)$ and $d(A_i)$ can be found in $O(\log N)$ time. We only need to focus on calculating $\sum T_x$, $\sum(d(p_x) \cdot T_x)$ and $\sum(d(x) \cdot T_x)$, while allowing updates.

To accomplish this, we can once again use fenwick trees and utilise the preorder decomposition of the tree to update and query the values we require using a range update, point query fenwick.

After calculating the two values of each path, we can add $dist(A_i, B_i) \cdot G$ to get our final answer.

Time complexity: $O((N + Q) \log N)$