Due to the high demand of limited ground stations, there are many situations in which several satellites would compete for access to a ground station. Moreover, each ground station can only communicate with one satellite at a time; and each satellite has only a few minutes before it travels beyond the ground stations' reach. In optimizing the ground station's connections, I would optimize for **fairness** and **total contact time**.

Total contact time is a direct measure of how much "use" the ground station has; but optimizing for it alone would allow ground stations to ignore satellites with smaller pass periods. This results in certain types of satellites not receiving the data that they need; which is why I would optimize for both fairness and total contact time.

To handle satellite-based scheduling, there are two main approaches I came up with:

The first method looks towards the future and calculates all of the passes within a certain time period (for example, a week). In determining each of the passes, the program would differentiate between "conflicted" time blocks and the "unconflicted" time blocks. Each "conflicted" time block would be a "decision" that is to be made. The ground station would calculate all of the possible 'decisions' that could be made, comparing which sequence of 'decisions' results in the most equal timings between satellites. This prioritizes fairness without losing out on total contact time. In terms of data structures, I would use a list with an index for every "conflicted" time block. As we calculate each of the decisions, we would fill up each index with the satellite picked to fill the block. However, this is extremely computationally expensive (O(2^n)), especially since each satellite we add increases the number of passes by 7 times or more.

A less computational intensive method would be to decide time allocation based on current-moment observations. While the program runs, it keeps track of how many minutes have been allocated to each satellite. Whenever there is more than one satellite seeking to connect to a ground station, the program will allocate the conflicted time period such that both of the satellites will have an equal "minute-count" in their records. Whenever a new satellite joins the network, they will be initialized to have the average "minute-count" of all of the satellites, so they are not overly prioritized. To do this, it is most efficient to keep track of each satellite's minute-count through a **dictionary**, since key/value access is O(1). While this approach lacks foresight, it takes the most efficient step in the moment, reducing the computational cost.

In both of these approaches, I have only considered the relationship between one ground station and each satellite. To get a more accurate estimate of satellite-data coordination, we could centralize the ground stations' control systems. With the previous approaches, the programs attempt to give each satellite equal access to one specific ground station. However, some satellites are on orbits which give them access to fewer ground stations. As a result, to meet our objective of **fairness**, it makes sense to try to equalize satellites' **total contact time**. This requires each ground station to know which other ground stations are being used by each satellite. This can be applied to both of the above approaches.
For the first approach, in calculating all of the possible combinations of "decisions" made, these decisions would include satellite passes with other ground stations as well. For the second

approach, the “minute-count” of each satellite could reflect the **total** minute count of each satellite: the sum of all the connection durations they have had with different ground stations.