1 ASSUMPTION

minimum gap between two consecutive flights has been taken as 1.5 hrs .
I have assumed that we know this information of cancellation due to malpractice well ahead of the scheduling , as no information with this regard has been given . One more reason which motivated me to take this assumption is that all the cancelled flights do not have the same timings , which means we have their list before their scheduled time . If this was the other way around then I could have added one condition in the BFS loop for n==1 of these departure times being strictly after the original departure time to handle this case .

2 INPUTS

1. The data of passengers has been stored in the passengers list.
2. **Flights.csv**
   For flights I have created a class named Flight which stores all the data about a given flight which includes
   - Departure Airport
   - Arrival Airport
   - Departure Time
   - Arrival time
   - A boolean value which stores true if the flight has been cancelled
   - A passenger list which has the passenger id of all those passengers who are travelling in this flight
   - Capacity of the flight
   - Empty space in the flight (capacity-number of passengers travelling in this flight)
   - A list of all possible alternate paths which can be used in case the flight is cancelled

   All the information about a flight can be accessed by only knowing its flight id
3. Cancelled Flights data is updated by the function cancel_flight in class Flight

3 CREATING THE GRAPH

A list of flight class is created (flight_list[])in which indexing is done by using the integer value of flight id . This list altogether contains each and every information about all the flights and I update the possible_routes and number of passengers .

Variables affected_people , allotted_people , total_layovers , total_time_diff have been created to update the stats.csv file .
About the Graph network : Using networkx library I have created a MultiDi-graph G which connects all the airports with edges . These edges represent flights , there maybe multiple flights between two airports with varying de-parture time , arrival time and capacities , this was the main reason to use MultiDiGraph and not Di-Graph .By iterating in the flight list i first add the airports the given flight is travelling between as nodes in my Graph network of all network . Then I update edges by again iterating in the flight_list along with an attribute given to each edge which is the flight id of the flight travelling between the airports which is responsible for the formation of the current edge . The reason for which I stored the flight id is that , while creating the graph we may have multiple edges between same nodes so this flight id will allow us to differentiate between these edges and also from an edge we can access all the information about the flight forming this edge sch as its arrival, departure time , capacity , empty space etc .
Note : I have formed an edge in the graph only if the flight involved has not been cancelled

4 BFS ALGORITHM

Since the list of canceled flights has been given , I have to find an alternate route for each of the passenger travelling between these nodes which satisfies the given constraints of the task .
I iterate through all the flights which have been cancelled and try to find all the valid_routes from the departure airport and arrival airport apart from those which are cancelled . For doing this I use the BFS algorithm in which I store two lists for each path , the first one being the nodes which will be there in the path ,i.e, the departure , intermediate and arrival airports of the current path and the second one is the flight ids of the flights which are travelling between two consecutive airports in the path . I push these two lists in the queue for BFS . At the beginning source node (departure airport) is pushed in path and the list of flights is empty .

4.1 BFS loop :

Queue q is used for the BFS. At each iteration I pop this and check the nodes which have been uploaded in the current path variable. If the path contains only one node then it is the source node and we iterate in its neighbors and append in the queue if the neighbor is not arrival airport, if it is the same as the arrival airport, it means we have another flight between these two airports and we can append this direct flight in our valid_routes list along with the flight id of this edge. If number of nodes in current path is 2, we can iterate through the neighbors of the last edge. This time we have to check one crucial condition

that the next flight a passenger takes must have some gap so that passenger can easily take the flight and his/her travelling schedule is not too hectic, I have set this gap duration to be 1.5 hours. If the next flight allows this gap, then we append the current path with this node and the current flight list with this flight's id and push back in queue, if this node is already the arrival airport, we do not push it into the queue but we append it into the valid_routes list . The last possibility can be that the current path has 3 nodes , in this case the 4^{th} node must be the arrival airport otherwise the condition that number of layovers < 3 will not be satisfied . So , we pop the queue and check if the last node has an edge between itself and the arrival airport and it should satisfy the condition of minimum gap between consecutive flights. If all the conditions are satisfied , the path is added to the valid routes otherwise we continue with the loop . Now I have a list of all the alternate paths of length <=3 which can be taken for each flight which has been cancelled , and these paths also have a minimum gap of 1.5 hrs between any two flights . I assign possible_routes for this flight equal to this list valid_paths .

5 Finding The maximum possible flow between the two involved airports

The next task is to assign these flight routes to the passengers which planned to travel in their cancelled flights . For this I have used a customised sorting of this list valid routes to satisfy the priorities of every passenger . Firstly , I have a complete list of possible paths between the two airports , therefore I will try to relocate all the passengers which the empty space in this flight allows . Secondly , the layovers must be minimum and the time diff that the two routes provide must be minimum , i.e , each passenger wants to reach as close as possible to the time of their original time . I sort the list in this order only , so now I have sorted the routes in the priority order which the passengers want . For assigning the routes to the passengers , I make two variables x, y. x helps to iterate in the self.passengers list and y in self.possible_routes . First I iterate through all the flights in the route and calculate the minima of their empty_space and then i take the minima of this value and passengers which are left to be allocated, this number gives the maximum flow through this route and I have named it route_flow. Now I decrease the empty space of all the flights of this route by the value route flow and then allocate route_flow number of passengers this route and append them to the reallocated list. Decreasing the empty_space ensures that in case this particular flight is used again, then there is no overflow of passengers which will be an illegal action, and cause problems. If there are no more routes left but some passengers have not been allotted a flight, then we we assign 0 to these passengers which represents that these were not allotted any route .
This is done for all the cancelled flights and in the end I have made a pandas dataframe from the reallocated list and make a allot.csv from this.

Note : While allotting flights to the passengers , what I have done is that I assigned on the basis of first come first serve basis(i.e, I have assumed that if the passenger comes first in the passengers.csv then he/she must be given priority, this way one more thing can be ensured that if some passengers are travelling in a group and have booked tickets consecutively then there will be a higher possibility that these will be assigned the same flight in addition to the benefit that those who have booked first should get a route first). Another method can be to randomly assigning the. If I had assign routes randomly then i could have used random function and swapping this element to change the active range, something similar to the selection sort.

6 RoadBlocks

1. Earlier I had made Di-Graph instead of MultiDiGraph unaware of the fact that there can be multiple edges between two nodes . MultiDiGraph was difficult to use and the traversals in edges between same nodes gave me a hard time.

2. I was trying to apply maximum flow algorithm, but due to the constraints of the problem, I was unable to accommodate all the required parameters into the generic algorithm, and thus had to rrewrite it based on the needs of the problem