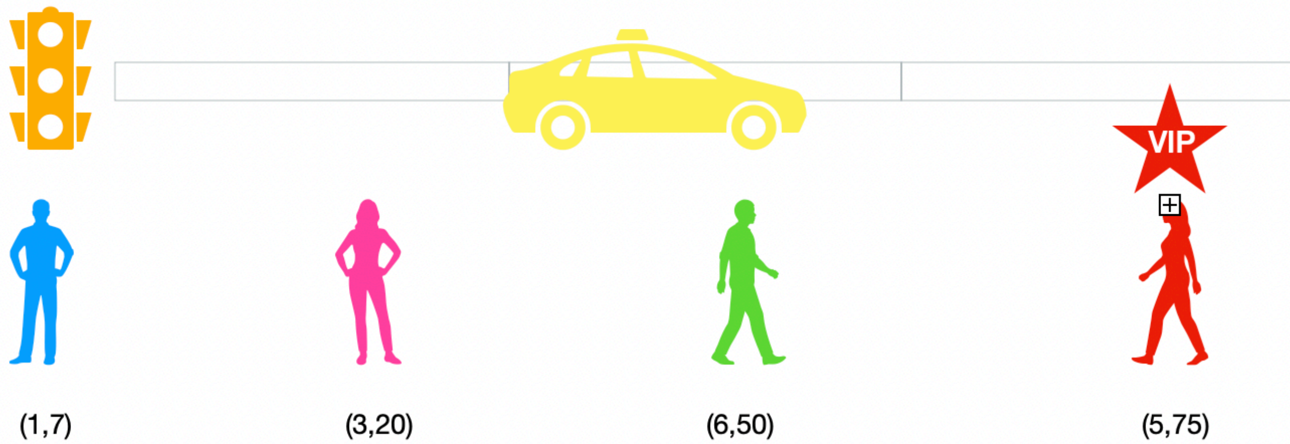
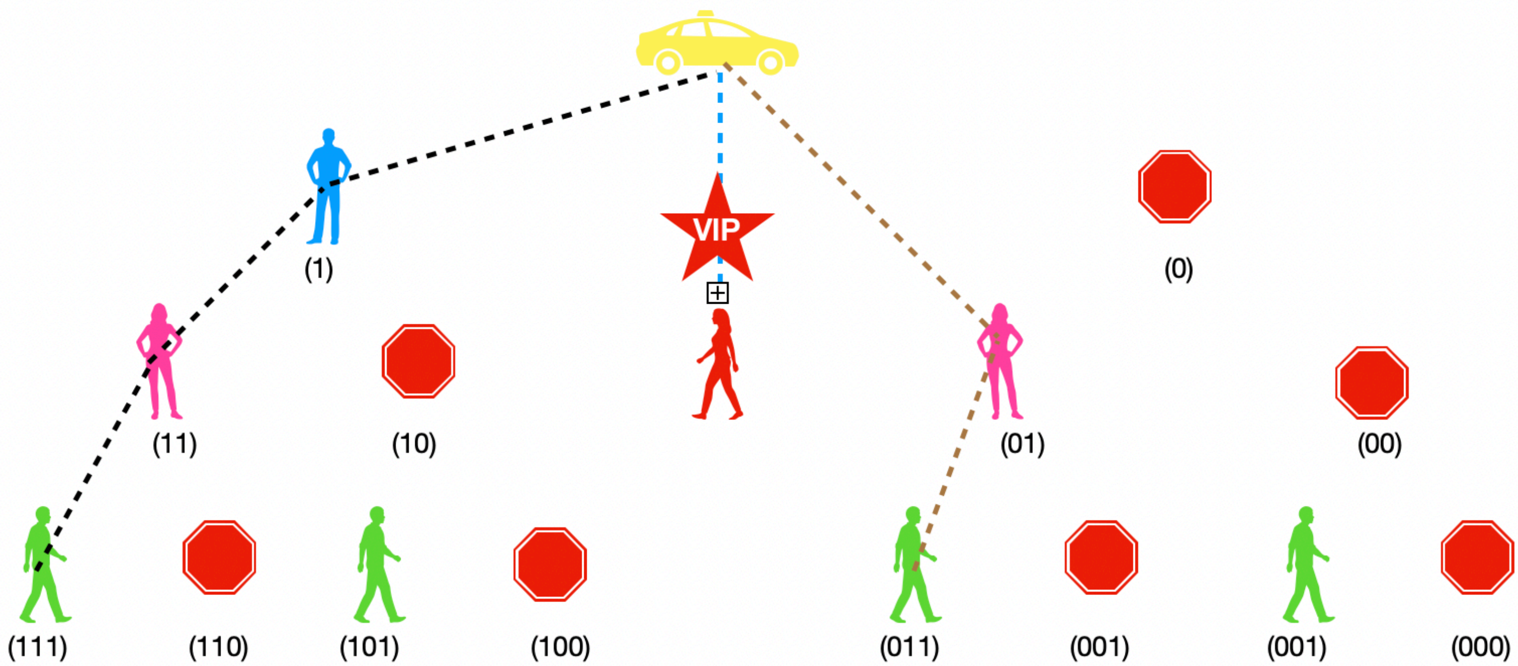




In this problem, as the example of Knapsack: exhaustive search, we can easily find the maximum fees by choosing the (111) route , and in case of having a limitation for total riders such as two , by choosing (011) route and in case of calling a VIP rider, by comparing the fee of vip with (111) total fees or in case of limitation by comparing with (011)

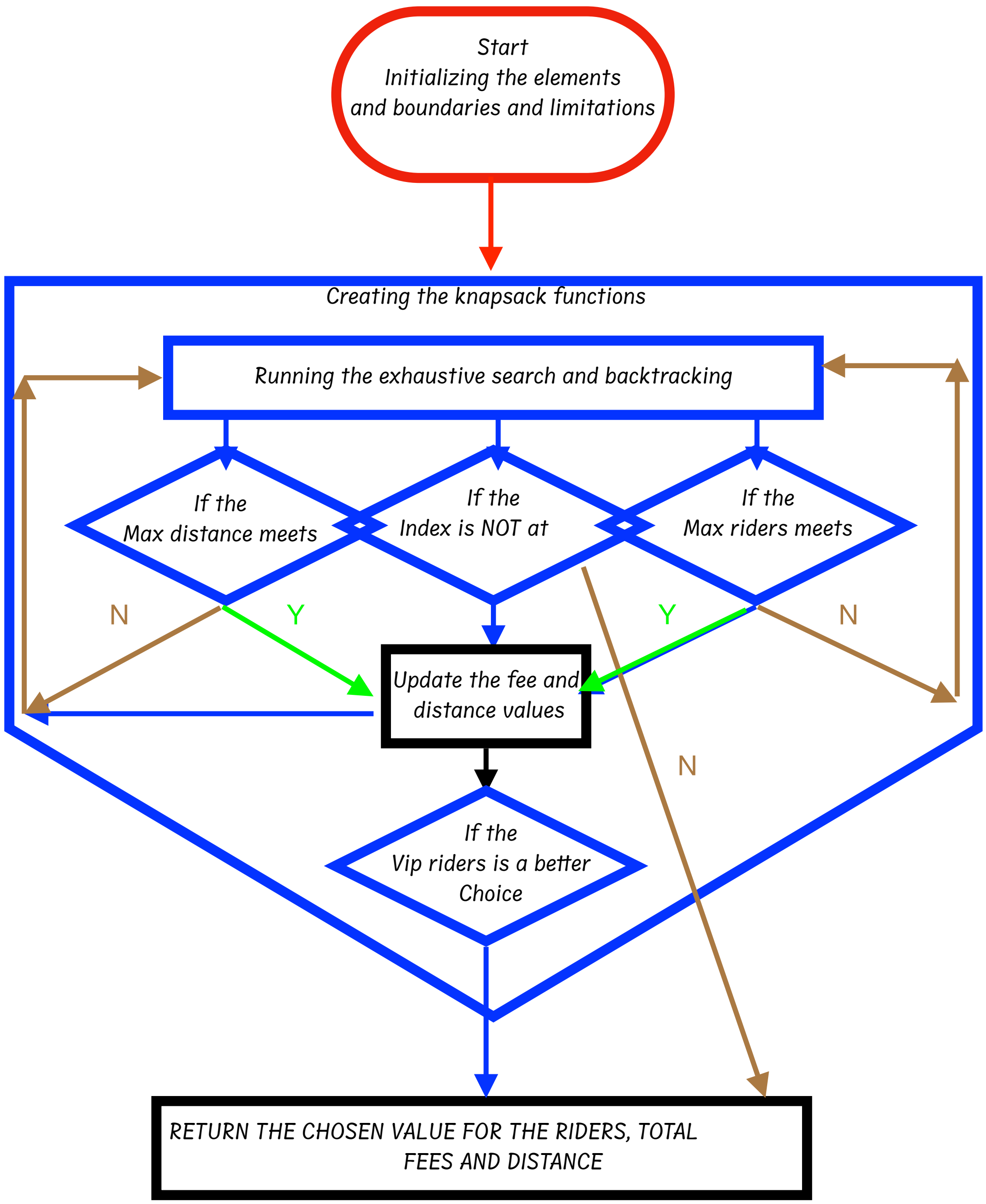
In this example we have the number of riders 3 and vip 1 but easily can having different amounts and for having a milage limitation it can be set as max distance in the program and in each test case the maximum fees under the maximum milage can be compared with the fee and milage of the vip rider.

A- no limitation of number of riders and no vip rider and only distance limitation :
This part is easy and no challenging complexity

B- having limitation in number of riders and distance:
In this case counts of chosen riders can be found by counting the '1' counts in each route and simply putting a condition on this

C- Vip riders added to the problem which can simply be solved by comparing the answer of parts A and B with the fee and distance of vip riders and check if the distance meets and fee is greater than the total fees returned by A and B.

Let's explain it by a flowchart.



determine the execution time for each test case:

*The time complexity of knapsack problem is O(N*D) in dynamic search which n is counts of riders and w is the max distance and is smaller than O(2^n).*

Simply as the max distance rises the time goes up.
By having more boundaries and limitation time can goes down as this fact program can be stopped earlier. And in case of having VIP riders, in the best case the program can compare the fee for the VIP and while it is greater than all other riders there is no need to excuse the knapsack while the answer is picking the VIP rider immediately.

T(c) <=T(b)<=T(a)<=O(2^n)