DIAL A RIDE PROBLEM

PROBLEM SCENARIO:

Given here are different locations in the whole city. Each of the location is provided with its distance to every other location across the city. There are passengers present at these locations at different point of time for a specific period. These passengers are waiting for certain cab to arrive which could take them to their destination location. Also there are cabs which have the responsibility of carrying passenger to their respective destination.

AIM :

The main aim of the problem is to schedule the cabs to move across different location in the city in such a way that they could satisfy requests of the passengers and generate a large revenue out of them.

CONSTRAINTS :

1. At any given instant of time one cab can carry as many passengers as specified by capacity of taxi.
2. Passengers have to be picked up from their source location (location of boarding the cab) within a defined period of time.
3. No restriction over the time at which passengers are dropped to their destination location.
4. The passenger pays according to the shortest distance between the source and destination location.

ASSUMPTIONS :

1. Each of the taxi is considered as an independent entity. The updation of information in one taxi is independent of the other. Eg. At a given time the time associated with one taxi is independent of the other taxis.
2. The speed of taxi is 1km/2minutes.
3. All the time is calculated and given in terms of minutes.

4. All the taxi time is 0 in the beginning indicating mid night.

REPRESENTATION OF INPUT DATA :

The different locations present in the city are provided in the form of adjacency matrix, here each of the location of the city is considered as one node and the distance in between the locations is edge of the two city location nodes in the graph.

The various attributes like capacity defined for a cab, total number of requests in the problem statement, total cabs in the city, Total number of locations in the city are stored within different variables.

As the two main entities in the above problem are cab and passenger which will interact with each other using all other information provided, in order to represent them structures are used.

Taxi Structure:

The taxi structure consists of information as

1. Taxi id → to uniquely identify a taxi.
2. Taxi location → Current location of the taxi.
3. Taxi Time → Time at which a request is processed
4. Capacity → Current capacity of the taxi

Request Structure:

The request structure consists of following information :

1. Request id → To uniquely identify a request.
2. Waiting start time → the time at which the request starts waiting at its source location
3. Waiting end time → the time at which the request ends waiting at its source location.
4. Source → The source location.
5. Destination → The destination location of the request.

Output Structure:

The output structure consists of following information :

1. Taxi id: the taxi which processed a certain request.
2. Request: The request processed by a particular taxi.
3. Revenue : The revenue of a particular taxi

Shortest distance array:

2 Dimensional arrays is used for representing all the locations and the distance in between them.

APPROACH:

Picking while Dropping :

Main approach used here is to keep on adding passenger to the taxi if the taxi satisfy the time constraints specified by the passenger request, until taxi capacity is full meanwhile, keep on updating the revenue time, capacity , location of the taxi.

Also while adding passenger to the taxi check if there is any request already present in the taxi whose destination location is same as current request source location. If it is so drop that passenger to that location and also update the information.

Dropping :

If the number of the passengers in the cab is equal to the capacity of the cab then sort all the request on the basis of there shortest distance from the current cab position.

Request with shortest distance is chosen and dropped to its destination while updating current cab position to its destination then continue with processing remaining requests performing above operation until the entire cab is empty. As the request is allocated to taxi, information about the current location of the cab, current time of the cab, total revenue associated with the cab is updated, ALGORITHM :

1. Take input from the file the variables like capacity of cab, number of locations in the city, number of requests, number of cabs.
2. Take input from the file the input containing distance between the different locations in the city and store it in a 2 dimensional array.
3. Apply Floyd Warshall algorithm to calculate the minimum distance in between all the locations in the city.
4. Create structure for both the cabs and the passenger.
5. The taxi structure have taxi id (for unique identification of the taxi), taxi time (each taxi is associated with a unique which keeps on incrementing for each of the taxi as it keeps on moving), current location of the taxi (the point at which taxi is present at current time period).
6. Passenger structure have request id(unique identification of each passenger), start time of waiting, end time of waiting, source location , destination location.
7. Sort the request on the basis of start time of waiting of the passenger.
8. For each of the request

   a. sort the cabs according to the shortest distance between the cab and the source location of the request in non decreasing order.

   b. for each cab

b1. if the taxi is able to reach to passenger within its specified time limit update the current time and location associated with the taxi also update the revenue and add the request to the cab.

Meanwhile if there is any request in the taxi whose destination is the source location of the current request which is been processed then drop him to its destination decrease the total number passenger in the taxi.

Otherwise Repeat step 8

b2. If at any time the number of passengers exceeds the capacity of taxi then for that taxi keep on dropping the passenger to their respective destination until no passenger is left in the taxi.

OPTIMIZATIONS :

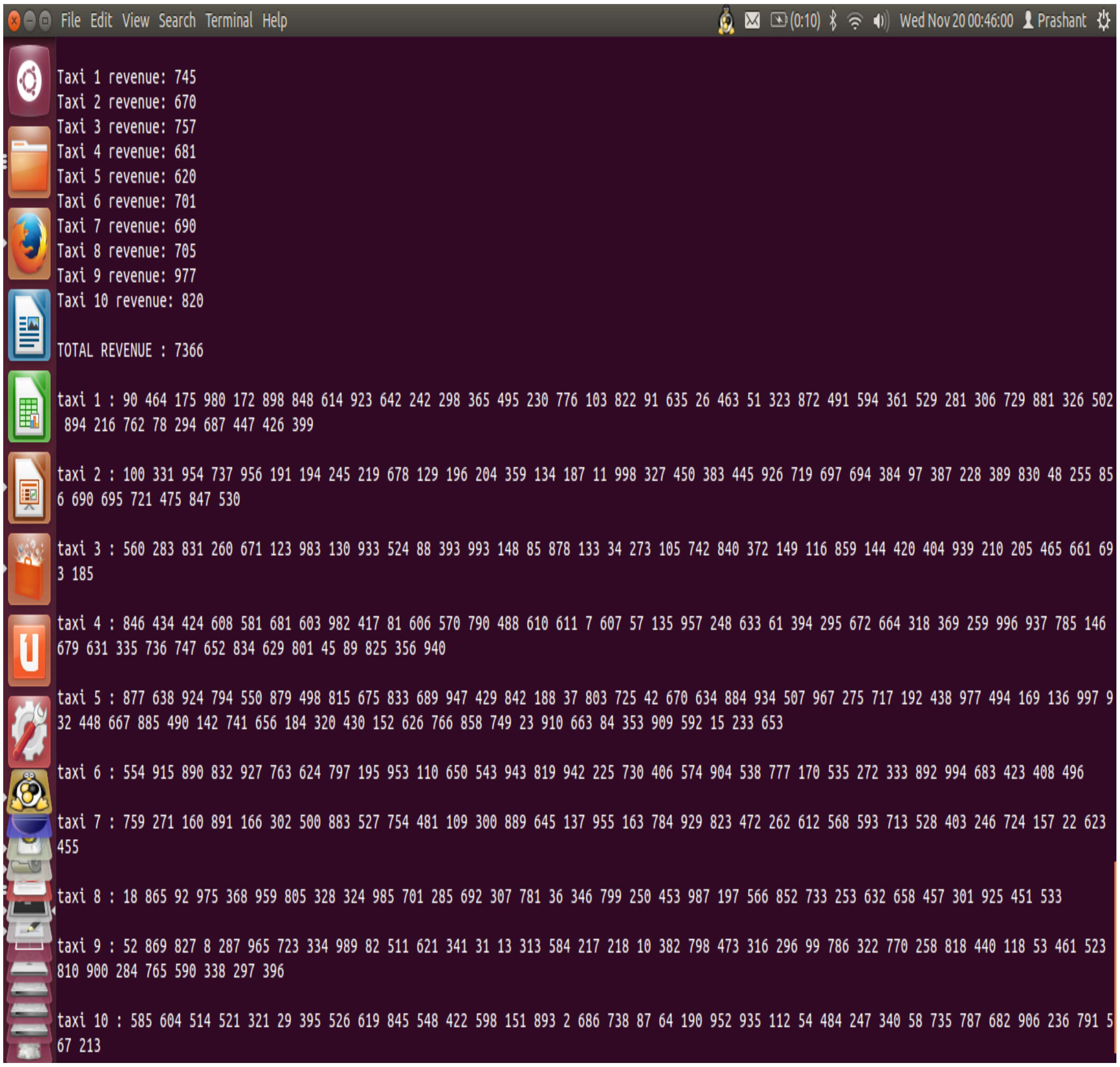
1. Using the concept of Drop While Pick. This will cause to drop the passenger request sitting already in taxi while picking another request from a specified location and decrease overhead of coming back to that location again to drop the same request leading to unnecessary increase in time which could be utilized to pick up some another request

2. Dropping in ascending order of distance from current location of taxi: Once the capacity of taxi is full then sort all the destination location of the request in the cab from current location of the taxi in increasing order of their distance . Drop the request at nearest distance from the taxi's current location and update the taxi location as the destination location of the dropped request. Again sort remaining request until all the request are dropped to their destination.

3. Wait for the request if taxi reached before starting wait time of the request : In order to increase possibility of adding a request to a taxi if a taxi reaches a location before waiting start time of the request then wait for the request to arrive at the its source location.

4. Check for the nearest taxi first and if it satisfies the request constraints laid by the request allocate request to the taxi: This is done to allocate the passenger to a taxi so that minimum update to the time of taxi has to be done. This time in turn could be used for identifying another request for more revenue.

OUTPUT :


File Edit View Search Terminal Help
(0:10) Wed Nov 20 00:46:00 Prashant

```
Taxi 1 revenue: 745
Taxi 2 revenue: 670
Taxi 3 revenue: 757
Taxi 4 revenue: 681
Taxi 5 revenue: 620
Taxi 6 revenue: 701
Taxi 7 revenue: 690
Taxi 8 revenue: 705
Taxi 9 revenue: 977
Taxi 10 revenue: 820

TOTAL REVENUE : 7366

taxi 1 : 90 464 175 980 172 898 848 614 923 642 242 298 365 495 230 776 103 822 91 635 26 463 51 323 872 491 594 361 529 281 306 729 881 326 502
 894 216 762 78 294 687 447 426 399

taxi 2 : 100 331 954 737 956 191 194 245 219 678 129 196 204 359 134 187 11 998 327 450 383 445 926 719 697 694 384 97 387 228 389 830 48 255 85
6 690 695 721 475 847 530

taxi 3 : 560 283 831 260 671 123 983 130 933 524 88 393 993 148 85 878 133 34 273 105 742 840 372 149 116 859 144 420 404 939 210 205 465 661 69
3 185

taxi 4 : 846 434 424 608 581 681 603 982 417 81 606 570 790 488 610 611 7 607 57 135 957 248 633 61 394 295 672 664 318 369 259 996 937 785 146
679 631 335 736 747 652 834 629 801 45 89 825 356 940

taxi 5 : 877 638 924 794 550 879 498 815 675 833 689 947 429 842 188 37 803 725 42 670 634 884 934 507 967 275 717 192 438 977 494 169 136 997 9
32 448 667 885 490 142 741 656 184 320 430 152 626 766 858 749 23 910 663 84 353 909 592 15 233 653

taxi 6 : 554 915 890 832 927 763 624 797 195 953 110 650 543 943 819 942 225 730 406 574 904 538 777 170 535 272 333 892 994 683 423 408 496

taxi 7 : 759 271 160 891 166 302 500 883 527 754 481 109 300 889 645 137 955 163 784 929 823 472 262 612 568 593 713 528 403 246 724 157 22 623
455

taxi 8 : 18 865 92 975 368 959 805 328 324 985 701 285 692 307 781 36 346 799 250 453 987 197 566 852 733 253 632 658 457 301 925 451 533

taxi 9 : 52 869 827 8 287 965 723 334 989 82 511 621 341 31 13 313 584 217 218 10 382 798 473 316 296 99 786 322 770 258 818 440 118 53 461 523
810 900 284 765 590 338 297 396

taxi 10 : 585 604 514 521 321 29 395 526 619 845 548 422 598 151 893 2 686 738 87 64 190 952 935 112 54 484 247 340 58 735 787 682 906 236 791 5
67 213
```

REFERENCES :

1. www.stackoverflow.com
2. Venkatesh Vishwarup
3. ALGORITHMS COURSE BOOK (CORMEN)