**7.2.2 Define problem**

* User requests 2 points, start point and end point and their settings includes: walking distance and transfer turn. System get requested points and settings, then search for suitable routes, for user travel from start point to end point.

**7.2.3 Attribute Definition**

* **Parameter:**

|  |  |  |
| --- | --- | --- |
| **Name** | **DataType** | **Explanation** |
| cityMap | CityMap | includes list of routes and list of stations |
| Start | Location | Start searched location including latitude and longitude. |
| End | Location | End searched location including latitude and longitude. |
| startAddress | String | Address name of start location, for rendering client’s result. |
| endAddress | String | Address name of end location, for rendering client’s result. |
| departureTime | LocalTime | Departure time. Including hour and minute. |
| walkingDistance | double | Maximum walking distance between two locations. |
| K | int | Maximum transfer turns for traveling. |
| searchType | Enum SearchType | There are five search types:  TWO\_POINT, THREE\_POINT, THREE\_POINT\_OPT, FOUR\_POINT, FOUR\_POINT\_OPT |

* **Output:** List of Result object. Each result object includes:
* List of stations round by start point: called LIST\_START\_STATIONS represent stations are found round by start point.
* List of stations round by end point: called LIST\_END\_STATIONS represent stations are found round by end point.
* Start path: represent the path includes calculated distance, time from one station in LIST\_START\_STATIONS to start point.
* End path: represent the path include calculated distance, time from one station in LIST\_END\_STATIONS to end point.
* List of results: called LIST\_RESULTS represent the list result is returned from Raptor’s algorithm and each result in this list will show to user.

**7.2.4 Solution**

* Search route algorithm is based on Raptor’s algorithm.
* To solve problem, we follow these steps:
  + Step 1 Calculate distance between two stations. If distance is less than walking distance, system will return message that user can walk between two points.
  + Step 2
    - Find all stations near start location:

**for all** station **in** city’s stations **do:**

**If** distance (start location, station) <= walking distance **then:**

**Add to LIST\_START\_LOCATIONS**

**Endif**

**endfor**

**Then add to LIST\_START\_STATIONS.**

For all stations Iterate list of stations of city map, find all stations that has a distance to start location less than walking distance, then the suitable station will be added into list of stations round by selected point. Within two selected point, we have two output: LIST\_START\_STATIONS and LIST\_END\_STATIONS.

* + - Find all stations nearby end location: same steps when building LIST\_START\_STATIONS. And then, we assign to LIST\_END\_STATIONS.
  + Step 3 – LIST\_START\_STATIONS, we build start path from one station in list to start point.
  + Step 4 – LIST\_END\_STATIONS, we build end path from one station in list to end point.
  + Step 5

For each station in LIST\_START\_STATIONS:

Find the shortest path

* + Calculating total distance and total time with entries are start path, start point, end point and end path by using Raptor’s algorithm. Each result is returned from Raptor’s algorithm will be added into list of results, we call it is LIST\_RESULTS:
    - Total distance is total of three distance below:
      * Distance of start path.
      * Distance of end path.
      * Distance from one station in LIST\_START\_STATIONS to one station in LIST\_END\_STATIONS.
    - Total time is total of three time below:
      * Time of start path.
      * Time of end path.
      * Time from one station in LIST\_START\_STATIONS to one station in LIST\_END\_STATIONS.
  + Step 6 – If LIST\_RESULTS is empty, we show message for notify user know no route found from start point to end point. If LIST\_RESULT is not empty, we sort the list follow three priorities below:
    - The total transfer turn is high priority: we sort list base on increasing transfer turn.
    - The total time is medium priority: we sort list base on increasing time.
    - The total distance is low priority: we sort list base on increasing total distance.
  + Step 7 – With LIST\_RESULTS sorted, we get the first three element for showing to user.