**7.2.2 Define problem**

* User request 2 points, start point and end point and their settings includes: walking distance, 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**

* City Map: includes list of routes and list of stations :

Each station in list of stations includes attributes: station’s name represent name of station, station’s code represent code of station, station’s street represent where the station locate, station’s latitude and longitude for coordinating station on the map.

* + Each route in list of routes includes attributes: route’s name, route’s no, list of paths, list of trips.
    - * List of paths: each path in list of paths include attributes: path no for represent the ordered station of route, station in from and station … ,
      * List of trips: each trip in list of trips represent start time and end time of route.
* 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.
* Distance

**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 less than walking distance, system will show message for user can walking between two points.
  + Step 2 - Find stations nearby selected points: Iterate list of stations of city map, find station have a distance to selected point 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.
  + 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 – Calculating total distance and total time with entries are start path, start point, end point and end path by using Raptor’s algorithm (reference here). 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.