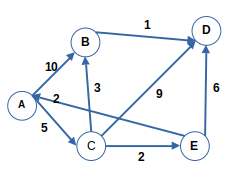
Code Challenge documentation

for the problem given above that to find the shortest path to different building for delivery throught routing calculation. To calculate this there is different algorigthm to my more adaptation I choose to use dijikstra algorithm for the calculation.

Lets pick one example and implement it with code

the delivery want reach five building A , B, C, D, E we use single source dijgstra algorithm shortest path method



our starting vertext is A all the building must be visited once

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Visted building | A | B | C | D | E |
| A | 0 | ∞ | ∞ | ∞ | ∞ |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Know let start the calculation

d[u] + c(u,v) < d[v]

then d[v] = d[u] + c(u,v)

d[A] + c(A,B) = 0 + 10 < ∞

d[B] = 10

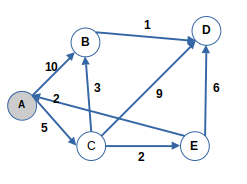
d[A] + c(A,C) < d[C]

d[C] = 0 + 5 < ∞

d[C] = 5

then d[C] < d[B] so

the shortest path from building A is C so the selected vertex is C



d[C] + c(C,B) = 5 + 3 < ∞

d[B] = 8 so we change d[B] = 10 to d[B] = 8

d[C] + c(C,D) < d[D]

d[D] = 5 + 9 < ∞

d[D] = 14

d[C] + c(C,E) < d[E]

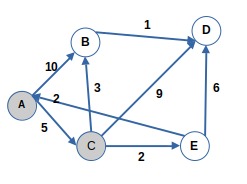
d[E] = 5 + 2< ∞

d[E] = 7

d[E] < d[B] and d[E] d[D] so

the shortest path from building C is E so the selected vertex is E

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Visted building | A | B | C | D | E |
| A | 0 | ∞ | ∞ | ∞ | ∞ |
| C |  | 10 | 5 | ∞ | ∞ |
| E |  | 8 |  | 14 | 7 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |



d[E] + c(E,A) = 7 + 3 > 0

d[A] = 0

d[E] + c(E,D) < d[D]

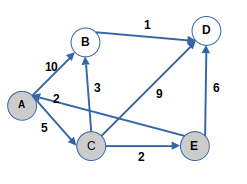
d[D] = 7 + 6 < 14

d[D] = 13

d[D] was 14 so changed to 13

there is two not visited building d[B] and d[D]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Visted building | A | B | C | D | E |
| A | 0 | ∞ | ∞ | ∞ | ∞ |
| C |  | 10 | 5 | ∞ | ∞ |
| E |  | 8 |  | 14 | 7 |
| B |  | 8 |  | 13 |  |
|  |  |  |  |  |  |



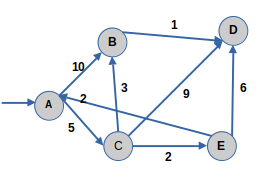
d[B] < d[D] so we select d[B] = 8

d[B] + c(B,D) < d[D]

d[D] = 8 + 1

d[D] = 9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Visted building | A | B | C | D | E |
| A | 0 | ∞ | ∞ | ∞ | ∞ |
| C |  | 10 | 5 | ∞ | ∞ |
| E |  | 8 |  | 14 | 7 |
| B |  | 8 |  | 13 |  |
| D |  |  |  | 9 |  |



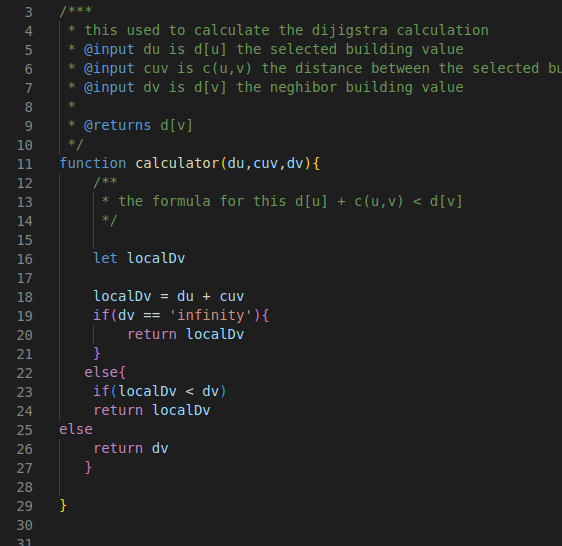
The shortest path from A to D is 9 know lets get the path

path:- D = B C A = 5 + 3 + 1 = 9

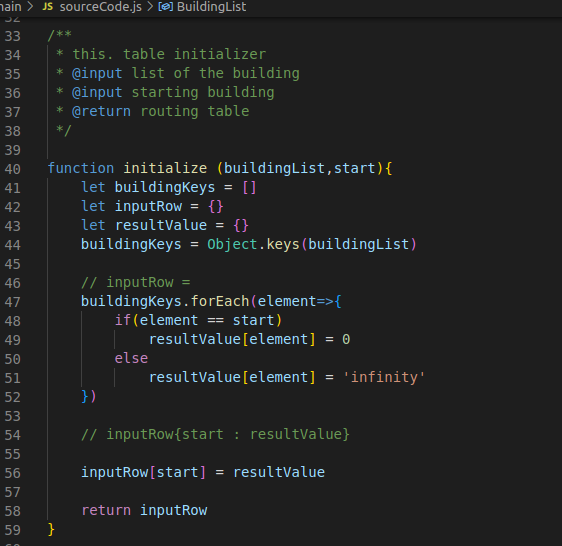
then the shortest path is A C B D

Source Code

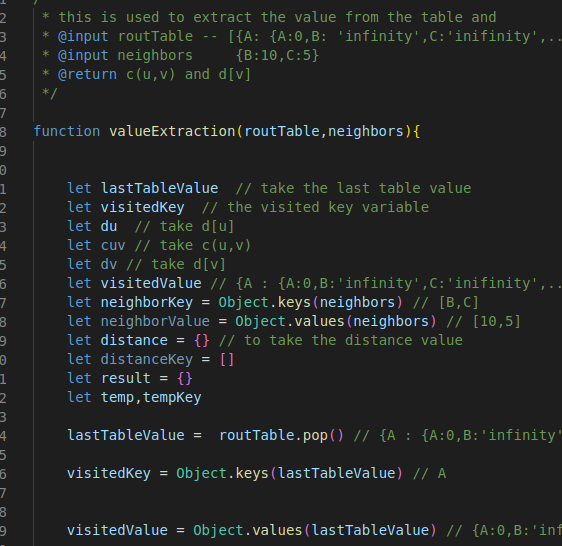
this one is use to calculate dijkstra algorithm

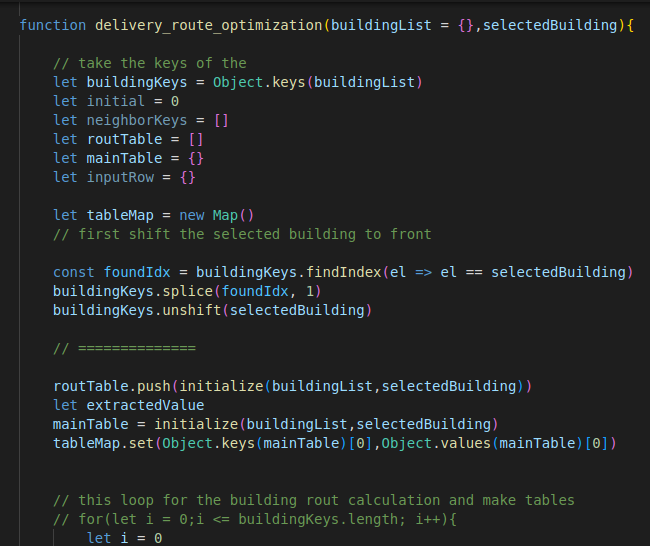


the next function is used to initialize the next the table row

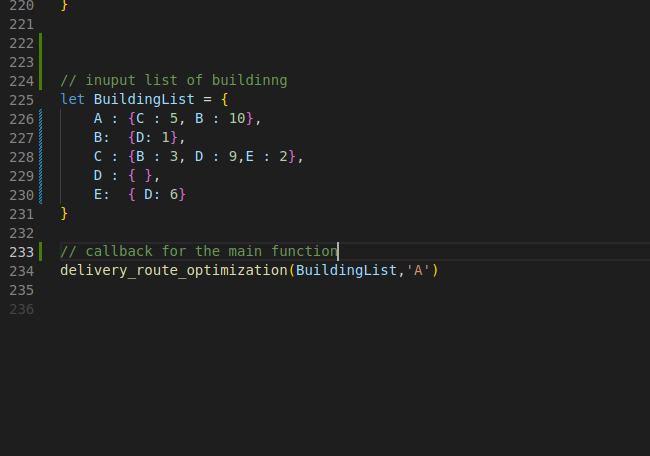


this used to extract the next rows

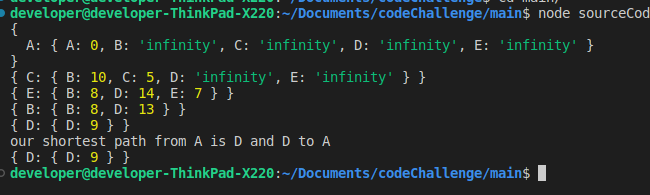


main function

**Test**



**result**



**Time and space complexity**

since the input building list is unsorted we used the worest case senario of time complexity and where V is number of vertices in graph and E is the number of Edge or Buildings . it is worse case scenario if it is a complete graph and every building has to be visited. In worst case complexity for Dijkstra's algorithm with O(V2) when implementing using an unsorted array and no priority queue.

Our inner loop statements occur O(V + E) times, where V is number of vertices and E is number of edges or Buildings, with the decrease key operation taking O(logV) meaning the total time complexity for our implementation is O((V + E)\*logV) as

E → V this simplifies to O(ElogV). Where we have the largest number of decrease key operations (which take logV).

Note that the time complexity is solely based on the number of elements in buildingList i.e the input length, so if the length of the buildingList array will increase the time of execution will also increase.

Since we have E = 7 V = 5

**the time complexy :-**

**ElogV=7log5=4.893ms**

**The space comlexity is :-**

**O(|V| + |E|) = 5 + 7 = 13 in micro**