# 位置点の近似照合

## 1　位置点の近似照合

・位置点の近似照合とは

GPSで蓄積された移動履歴の比較や検索を可能にするための類似性を定義し類似性に基づいた類似度計算を行う

## 1　類似度計算の流れ

・入力は2点列の位置点の軌跡とする

→まず、2点列の軌跡の類似度を定義する必要がある

→全体の類似度を定義するため、点と点が一致、点と線が一致したときの
重みを一致度とし、一致度の和を類似度と定義する

→類似度は「最長共通(近接)部分列を求めるアルゴリズム」で求められる
(帰納的関係により、動的計画法を用いて求める)

→動的計画法でテーブルに格納した値を、後ろからたどって
最長共通(近接)部分列(2点列の軌跡)を抜き出す

## 1　全体の流れ

・移動履歴やその部分の比較や検索を可能にしたい
移動履歴の類似度を定義し、類似性に基づく類似度計算、照合を行う

→2つのGPSデータから、緯度経度を抜き出し2点間の距離を算出

ある一定の距離$\delta$までは同一地点とみなし、一致したとして
それまでの類似度に1を加算する

入力された点列を$S$, Tとする
$S$の先頭から$i$番目までの列$S[1, \dots \dots, i]$と
Tの先頭から$j$までのT$[1, \dots \dots, j]$の類似度を$dp(i, j)$と定義

類似度$dp(0, j), dp(i, 0)$は類似度0とする

## 1　2点列の類似度を求めるアルゴリズム(最長共通部分列)

類似度は以下のように定義する

$$dp(i,j) = max \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \begin{cases} dp(i-1, j-1) + (D(s_i, t_j) \leq \delta) \\ dp(i-1, j) \\ dp(i, j-1) \end{cases} \end{cases}$$

$\left(D\left(s_i, t_j\right) \leq \delta\right)$が真のとき1, 偽のとき0

これは2つの文字列の共通部分列(最長共通部分列)を求める問題と定義が等しい
↓
最長共通部分列を解くアルゴリズムで類似度を求めることができる

## 1　最長共通部分列問題

最長共通部分列問題とは･･･
与えられた2つの文字列の最長共通部分列を見つけ出す問題

共通文字列の順番は同じでなければならないが、文字列が連続している必要はない

例：$S = < A, Y, B, Z, C > \; T = < A, B, C, X >$ ,を入力とする

$S, T$ の最長共通部分列は「$ABC$」
$S, T$ の最長共通部分列の長さは3

# 1 最長共通部分列

最長共通部分列を求める過程

入力列を$S$ =< $A$, $Y$, $B$, $Z$, $C$ >,　$T$ =< $A$, $B$, $C$, $X$ >とする

(テーブルで0番目の要素を定義している)

$dp(i,j)$は、$S$を左から$i$文字、
$T$を左から$j$文字切り出した部分文字列の
共通文字列の長さ(類似度)になる

(例)
$dp(3,3) = 2$

$dp(m,n)$まで到達したときの値が
最長共通部分列の長さ(類似度)になるため、

入力列$S$と$T$の最長共通部分列の長さ(類似度)は
$dp(5,6) = 3$

| | 0 | A | Y | B | Z | C |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 1 | 1 | 1 | 1 | 1 |
| B | 0 | 1 | 1 | 2 | 2 | 2 |
| C | 0 | 1 | 1 | 2 | 2 | 3 |
| X | 0 | 1 | 1 | 2 | 2 | 3 |

## 1バックトラックで同一地点の座標を取り出す(最長共通部分列)

バックトラックで最長共通部分列を抜き出す過程

$$dp(i,j) \neq \left\{ \begin{matrix} dp(i-1,j) \\ dp(i-1,j-1) \\ dp(i,j-1) \end{matrix} \right\}$$

これを満たす$dp(i,j)$ を見つけて
(テーブルの座標)を取り出す作業

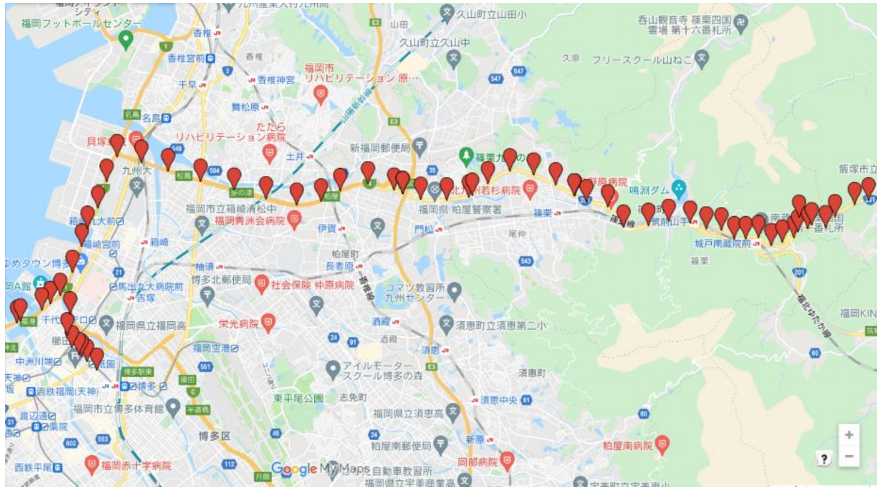
$dp(i,j) = dp(i-1,j-1)$を満たしたら、左上に移動
$dp(i,j) = dp(i-1,j)$ を満たしたら、上に移動
$dp(i,j) = dp(i,j-1)$を満たしたら、左に移動

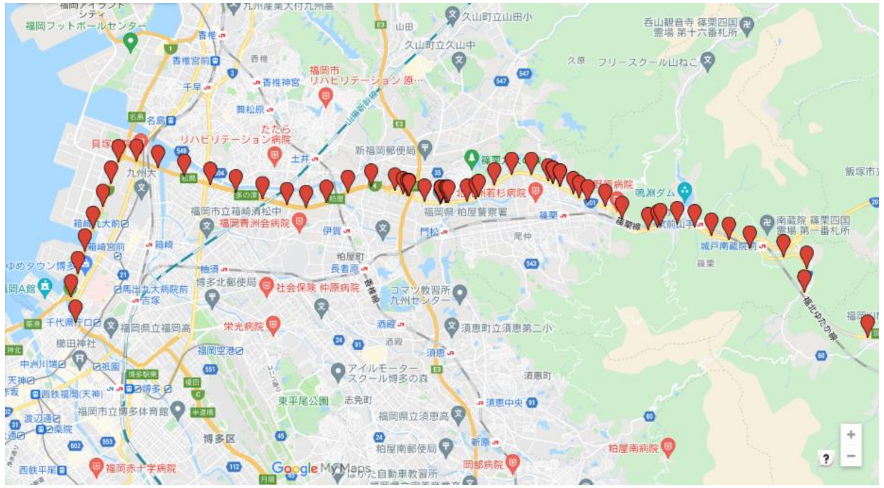
この操作で「CBA」という文字列を得られる

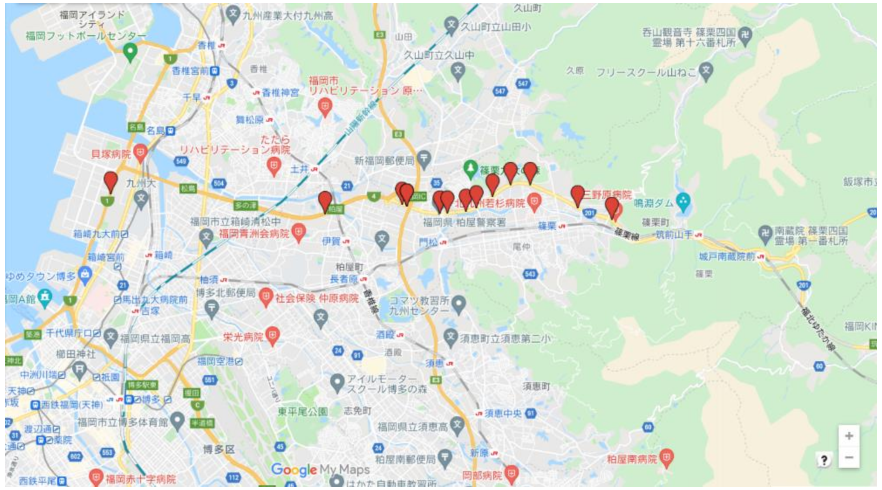
ここで得られる文字列は与えられたデータの逆順になっている

| | 0 | A | Y | B | Z | C |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | 1 | 1 | 1 | 1 | 1 |
| B | 0 | 1 | 1 | 2 | 2 | 2 |
| C | 0 | 1 | 1 | 2 | 2 | 3 |
| X | 0 | 1 | 1 | 2 | 2 | 3 |

# 5　バックトラックで同一の地点の座標を取り出す(最長共通部分列)



# ２点間の線分への拡張

## 1　定義の拡張

- 点と点の軌跡のみで類似度を定義していた
- これからは、点と2点間の線分の軌跡も含めて類似度の定義を拡張する

- 最長共通部分列を求めるアルゴリズムでは点と点の類似度しか求められないので、最長共通部分列問題の定義を拡張する必要がある

## 2　類似度の定義(最長近接部分列)

・点と線分が交互に並ぶ列を定義する

$$\hat{P} =< p_0, (p_0, p_1), p_1, (p_1, p_2), p_2, \dots\dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$
$$\hat{Q} =< q_0, (q_0, q_1), q_1, (q_1, q_2), q_2, \dots\dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

隣り合う点の間で定義される線分

$\hat{P}, \hat{Q}$ の添え字を組にした列を
$\psi = < (\psi_0(0), \psi_1(0)), \dots\dots, (\psi_0(k-1), \psi_1(k-1)) >$ として

点列 $P$ と点列 $Q$ の類似度 $|\psi|$ を以下のように
一致度 $w(\psi_0(i), \psi_1(i))$ の和と定義する

$$|\psi| = \sum_{i=0}^{k-1} w(\psi_0(i), \psi_1(i))$$

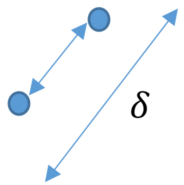
$w(\psi_0(i), \psi_1(i))$を、$\hat{P}(i), \hat{Q}(i)$が
一致したときの重みと定義する

## 3　一致度の定義(最長近接部分列)

(1) $P$の点$p_i$と$Q$の点$q_j$が同一視できる距離$\delta$内にある

$$d(i,j) \leq \delta$$

一致度 → $w(i,j) = 1$



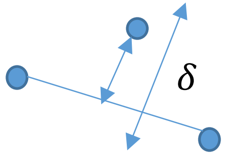

(2) 点$p_i$と隣接する2つの点$q_j, q_{j+1}$を結ぶ線分が距離$\delta$内にある

$$d(p_i, (q_j, q_{j+1})) \leq \delta$$

点$q_j$と隣接する2つの点$p_i, p_{i+1}$を結ぶ線分が距離$\delta$内にある

$$d(q_j, (p_i, p_{i+1})) \leq \delta$$

一致度 → $w(i,j) = \frac{1}{2}$




(3) それ以外

一致度 → $w(i,j) = 0$　(線分と線分は同一視しない)

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

最長近接部分列を求める動的計画法によるアルゴリズム
以下の帰納的関係によって求められる

$$dp(i,j)=\begin{cases} dp(i,j)+1 & d(i,j)\le\delta, \quad \text{点と点が一致} \\ dp(i,j)+\dfrac{1}{2} & d(i,j)\le\delta, \quad \text{点と線が一致} \\ max\begin{Bmatrix} dp(i-1,j) \\ dp(i-1,j-1) \\ dp(i,j-1) \end{Bmatrix} & d(i,j)>\delta \end{cases}$$

計算量は$O(nm)$　点列の長さの積になる
2つの配列で列の長さが$n,m$の場合

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

入力列を$\widehat{P}, \widehat{Q}$とする。どういう動きをするか考える

$$\widehat{P} = < p_0, (p_0, p_1), p_1, (p_1, p_2), p_2, \dots \dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\widehat{Q} = < q_0, (q_0, q_1), q_1, (q_1, q_2), q_2, \dots \dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

一点と一点のときは$(p <--> q)$のみ一致度を考えるだけだったが、今回は1点と2点間の線分も含むので,$(p <--> q)$,$(p <--> (q-1, q))$,$\left(q <--> (p-1, p)\right)$の一致度も考慮しなければならない

$(p <--> q)$の類似度を$dp(i,j).qp$
$(p <--> (q-1, q))$の類似度を$dp(i,j).lp$
$\left(q <--> (p-1, p)\right)$の類似度を$dp(i,j),ql$
と区別する

## 4　2点列の類似度を求めるアルゴリズム(最長共通部分列)

最長共通部分列の場合

$dp(i, j)$(赤線までの類似度)を求めたい

$$\widehat{P} = < \underline{p_0, p_1, \dots\dots, p_{i-1}, \boldsymbol{p_i}}, \dots\dots, p_{m-1} >$$

$$\widehat{Q} = < \underline{q_0, q_1, \dots\dots, q_{j-1}, \boldsymbol{q_j}}, \dots\dots, q_{n-1} >$$

# 4　2点列の類似度を求めるアルゴリズム(最長共通部分列)

最長共通部分列の場合

（1） $dp(i,j) = dp(i-1,j)$

$$\hat{P} = < \underline{p_0, p_1, \ldots \ldots, p_{i-1}}, \boldsymbol{p_i}, \ldots \ldots, p_{m-1} >$$

$$\hat{Q} = < \underline{q_0, q_1, \ldots \ldots, q_{j-1}, \boldsymbol{q_j}}, \ldots \ldots, q_{n-1} >$$

# 4　2点列の類似度を求めるアルゴリズム(最長共通部分列)

最長共通部分列の場合

$(2) dp(i, j) = dp(i, j-1)$

$$\hat{P} = < p_0, p_1, \dots\dots, p_{i-1}, \boldsymbol{p_i}, \dots\dots, p_{m-1} >$$

$$\hat{Q} = < q_0, q_1, \dots\dots, q_{j-1}, \boldsymbol{q_j}, \dots\dots, q_{n-1} >$$

# 4　2点列の類似度を求めるアルゴリズム(最長共通部分列)

最長共通部分列の場合

(3) $dp(i,j) = dp(i-1, j-1) + 1$

$$\hat{P} = < p_0, p_1, \dots\dots, p_{i-1}, \boldsymbol{p_i}, \dots\dots, p_{m-1} >$$

$$\hat{Q} = < q_0, q_1, \dots\dots, q_{j-1}, \boldsymbol{q_j}, \dots\dots, q_{n-1} >$$

## 4　2点列の類似度を求めるアルゴリズム(最長共通部分列)

最長共通部分列の場合
$dp(i,j)$

$$\hat{P} = < p_0, p_1, \dots \dots, p_{i-1}, \boldsymbol{p_i}, \dots \dots, p_{m-1} >$$

$$\hat{Q} = < q_0, q_1, \dots \dots, q_{j-1}, \boldsymbol{q_j}, \dots \dots, q_{n-1} >$$

$$dp(i,j) = max \begin{array}{l} dp(i-1,j-1)+1 \ \ (d(i,j) \le \delta のとき) \\ dp(i-1,j) \\ dp(i,j-1) \end{array}$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(1) $dp(i, j).\,lp$・・・$(p <\!-\!> (q-1, q))$

$$\hat{P} = < p_0, (p_0, p_1), p_1, \dots \dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots \dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} = < q_0, (q_0, q_1), q_1, \dots \dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots \dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(1) $dp(i,j).lp = \underline{dp(i,j-1).ql} + \frac{1}{2}$

$$\hat{P} = < \underline{p_0, (p_0, p_1), p_1, \ldots\ldots, p_{i-1}, (p_{i-1}, p_i)}, \boldsymbol{p_i}, \ldots\ldots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} = < \underline{q_0, (q_0, q_1), q_1, \ldots\ldots, q_{j-1}}, (q_{j-1}, q_j), \boldsymbol{q_j}, \ldots\ldots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

# 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(2) $dp(i,j).lp = dp(i,j-1).qp$

$$\hat{P} = < p_0, (p_0, p_1), p_1, \dots \dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots \dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} = < q_0, (q_0, q_1), q_1, \dots \dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots \dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(3) $dp(i,j).lp = dp(i-1,j).lp$

$$\hat{P} =< p_0, (p_0, p_1), p_1, \dots\dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots\dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} =< q_0, (q_0, q_1), q_1, \dots\dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots\dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

# 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

$dp(i,j).lp$・・・$(p <-> (q-1, q))$

$$\widehat{P} = < p_0, (p_0, p_1), p_1, \ldots\ldots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \ldots\ldots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\widehat{Q} = < q_0, (q_0, q_1), q_1, \ldots\ldots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \ldots\ldots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

以上のことから

$$dp(i,j).lp = max \begin{array}{l} dp(i,j-1).ql + \frac{1}{2} \\ dp(i,j-1).qp \\ dp(i-1,j).lp \end{array}$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

（1） $dp(i,j).ql$ ・・・ $(q <-> (p-1,p))$

$$\hat{P} = < p_0, (p_0, p_1), p_1, \dots \dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots \dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} = < q_0, (q_0, q_1), q_1, \dots \dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots \dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(1) $dp(i,j).ql = \underline{dp(i-1,j).lp} + \frac{1}{2}$

$$\hat{P} =< \underline{p_0, (p_0, p_1), p_1, \dots\dots, p_{i-1}}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots\dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} =< \underline{q_0, (q_0, q_1), q_1, \dots\dots, q_{j-1}, (q_{j-1}, q_j)}, \boldsymbol{q_j}, \dots\dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(2) $dp(i, j) = dp(i-1, j).qp$

$$\hat{P} = < p_0, (p_0, p_1), p_1, \dots \dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots \dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} = < q_0, (q_0, q_1), q_1, \dots \dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots \dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

# 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(3) $dp(i, j).ql = dp(i, j-1).ql$

$$\hat{P} =< p_0, (p_0, p_1), p_1, \dots \dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots \dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} =< q_0, (q_0, q_1), q_1, \dots \dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots \dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(1) $dp(i, j).\mathrm{ql}$ ・・・ $(q <-> (p-1, p))$

$$\hat{P} = < p_0, (p_0, p_1), p_1, \dots \dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots \dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} = < q_0, (q_0, q_1), q_1, \dots \dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots \dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

以上のことから

$$dp(i,j).ql = max \begin{array}{l} dp(i-1,j).lp + \frac{1}{2} \\ dp(i-1,j).qp \\ dp(i,j-1).ql \end{array}$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

$(1)dp(i,j).qp\cdots(p <-> q)$

$$\widehat{P} = < p_0, (p_0, p_1), p_1, \dots \dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots \dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\widehat{Q} = < q_0, (q_0, q_1), q_1, \dots \dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots \dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

# 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(1) $dp(i.j).qp = dp(i-1, j-1).qp + 1$

$$\widehat{P} =< p_0, (p_0, p_1), p_1, \dots\dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots\dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\widehat{Q} =< q_0, (q_0, q_1), q_1, \dots\dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots\dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(2) $dp(i,j).qp = dp(i,j).lp$

$$\hat{P} = < p_0, (p_0, p_1), p_1, \dots\dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots\dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} = < q_0, (q_0, q_1), q_1, \dots\dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots\dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

(3) $dp(i,j).qp = dp(i,j).ql$

$$\hat{P} = < p_0, (p_0, p_1), p_1, \dots \dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots \dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

$$\hat{Q} = < q_0, (q_0, q_1), q_1, \dots \dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots \dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

## 4　2点列の類似度を求めるアルゴリズム(最長近接部分列)

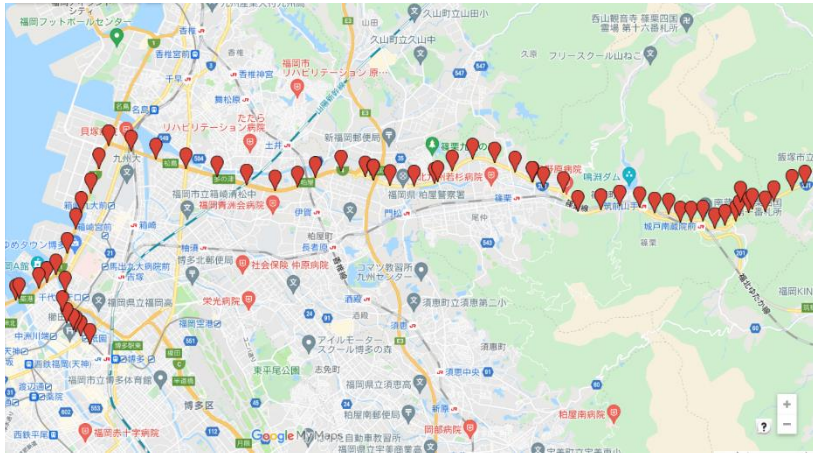
(1) $dp(i,j).qp \cdots (p <-> q)$

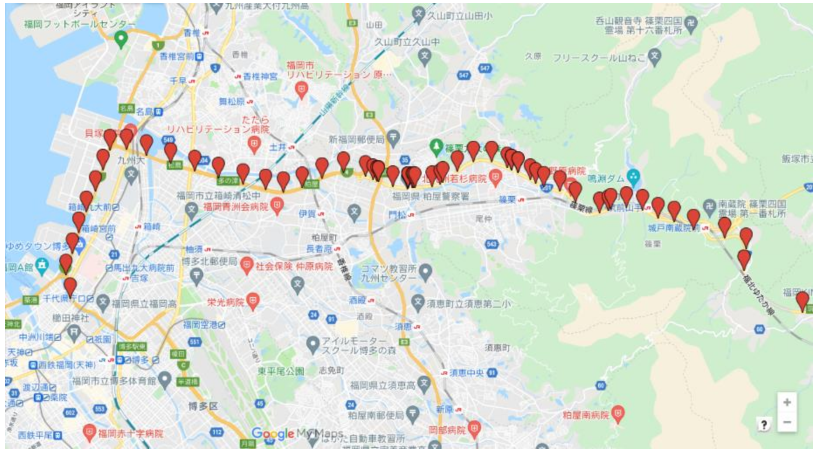
$$\hat{P} = < p_0, (p_0, p_1), p_1, \dots\dots, p_{i-1}, (p_{i-1}, p_i), \boldsymbol{p_i}, \dots\dots, (p_{m-2}, p_{m-1}), p_{m-1} >$$

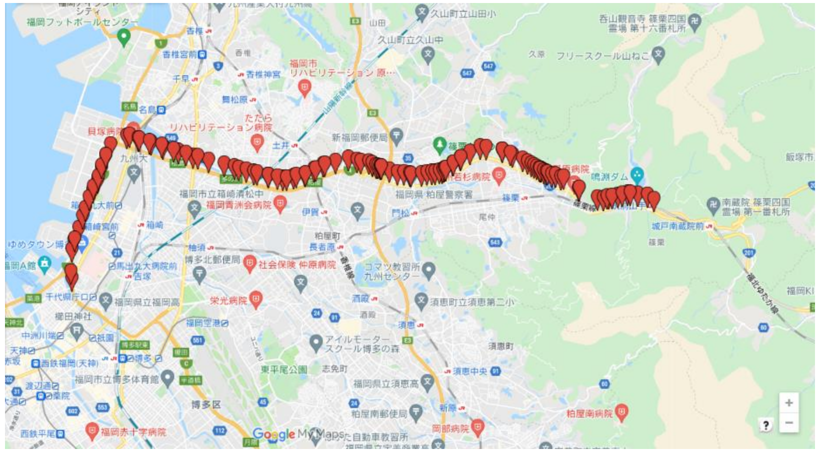
$$\hat{Q} = < q_0, (q_0, q_1), q_1, \dots\dots, q_{j-1}, (q_{j-1}, q_j), \boldsymbol{q_j}, \dots\dots, (q_{n-2}, q_{n-1}), q_{n-1} >$$

$$dp(i,j).qp = max \begin{cases} dp(i-1, j-1).qp + 1 \\ dp(i,j).lp \\ dp(i,j).ql \end{cases}$$

# 5 バックトラックで同一の地点の座標を取り出す(最長近接部分列)







2点間の線分は、2点間の中間の点で可視化した