《論　文》

物流輸送ネットワークにおけるクリティカルパス探索プログラム
―物流配送ネットワークにおける最適ルートの探索―

蜂　谷　　博

物流配送計画において，ある出発地点から，対象の全地域に物資を配送する場合に，最適なルート（Critical path）を探索するプログラムを開発する。

最適なルートとは，⑴最短距離で到達できる経路，⑵最短の所要時間で到達できる経路，⑶最小の輸送コストで到達できる経路　などが考えられる。⑶輸送コストの最小化という問題は，距離・時間・選択ルートなど複合的な要因によるものなので，本論文ではひとまず棚上げとする。本稿では⑴最短距離経路，あるいは⑵最短所要時間経路を求めるプログラムの開発を目指す。

最短経路の探索または最短所要時間経路の探索は，評価するデータが距離であるか時間であるかの違いであって，アルゴリズム的にはほぼ共通である。プログラム開発技術でも同じことが言え，評価するデータを距離データとするか，所要時間データとするかによって最短距離を探索するか，最短所要時間を求めるかを変更することができる。すなわちデータを入れ替えることによって，同一のアルゴリズム（プログラム）で問題を解決できる。詳細は本文の中で述べる。

§１．目的およびねらい

本稿では，物流配送ネットワークにおいて，ある拠点から出発して全地域に物資を届ける問題において，輸送するための最適な経路を探索するプログラムを開発することを目的とする。

中心とするアルゴリズムは，ダイクストラ法と呼ばれるモデルを採用する。このような配送ネットワークにおける最適な経路の探索問題のアルゴリズムは，ダイクストラによって考案され完成されている。本稿では，このダイクストラによるアルゴリズムを追試し，Ｃ言語によるプログラムとしてインプリメントし，さらにいくつかの改良を試み

る。

ダイクストラによるアルゴリズムの概略は，インプット・データである各都市間の距離データを隣接行列（Adjacent matrix）として与え，Adjacent Matrixから最適な経路（critical path）を探索するという仕組みである。

教科書的な問題の解法は，ダイクストラによるアルゴリズムによって完成されている。（[注] ダイクストラは，構造化プログラミングの提案でも知られている。）

ここで教科書的というのは，対象とする都市数があまり多くなく（10都市程度），人間がそのアルゴリズムを追跡して容易に理解できる規模という意味である。比較的小さなモデルで検証ができていることから，そのアルゴリズムの正しいこと（妥当性）が証明されるわけであるが，実用的な問題や，さらに多くの都市が存在する問題では，使い勝手という点では改良すべき問題が残されている。

解決すべき問題とは，以下の点である。

⑴インプット・データの与え方

ダイクストラによるアルゴリズムでは，インプット・データは，隣接行列（Adjacent Matrix）として，配列の初期値（図１）の形でプログラムの中に書き込まれる。

```
//#define   N   10     /* 節点の数   ノードポイント   */
//#define   M   9999

//   char   node[N+3][8]={"〜","金澤","富山","上越","長岡","東京","高山","松本","八王子","名古屋","静岡","--","==" };   /* 節点名 */
//   char   city_tbl[P+3][10]
//                          {"〜", 金, 富, 越, 長, 東,"高",松,"八","名","静",   /* 節点名 */
//int   a[N+1][N+1] ={{ 0,   0,   0,   0,   0, 0,    0,   0,   0,    0,   0  }, /* 隣接行列 */
//                     { 0,   0, 90,   M,   M, M, 138,   M,   M, 348,   M   },
//                     { 0, 90,   0, 84,   M, M,   66,   M,   M,    M,   M   },
//                     { 0,   M, 84,   0, 84, M,    M, 120, M,     M,   M   },
//                     { 0,   M,   M, 84,   0, 180, M,   M, 240,   M,   M   },
//                     { 0,   M,   M,   M, 180, 0,   M,   M, 126,   M, 150 },
//                     { 0, 138, 66, M,   M,   M,   0, 120,   M, 156,   M   },
//                     { 0,   M,   M, 120, M,   M, 120, 0,   120,   M, 132 },
//                     { 0,   M,   M,   M, 240, 120, M, 120,   0,   M, 150 },
//                     { 0, 348, M,   M,   M,   M, 156, M,     M,   0,   120 },
//                     { 0,   M,   M,   M,   M, 150, M, 180, 150, 120,   0 },
//                    };
```

図１　隣接行列（Adjacent Matrix）金澤=>東京

そのため，最適経路を探索すべき地域が変わると，プログラムそのもののソース・コードを修正し直さなければならない。対象とする地域があまり大きくない「教科書的なモデル」ではこれでもよいが，対象地域データを変更したり，対象地域がもっと広域になった場合に，隣接行列を人力で作り，タイプインすることはかなり大変である。さらに作成した隣接行列を書き込む（タイプし直す）としても，タイプミスをし易い。そ

こで，プログラムの中で自動的に隣接行列を生成する機能を開発することにした。対象地域が変わった場合，その都市間データをデータ・ファイルから読み込むことによって，隣接行列をプログラムの中で生成できるように改良した。

そのため，次の 4 段階のプログラム改良・開発を行なった。

（1.1） phase.1：隣接行列からクリティカルパスを探索するプログラム

（1.2） phase.2：都市間データから，隣接行列を生成するプログラム

（1.3） phase.3：データ・ファイルから都市間データを読み込んで，隣接行列を生成するプログラム

（1.4） phase.4：プログラムの統合化。(Ⅰ)〜(Ⅲ)の 3 本のプログラムを一つのプログラムに機能を統合する。

(2) 出力情報の改良

ダイクストラのアルゴリズムでは，最適解（最短）とされる都市の経由ルートを解として出力するだけであったが，本プログラムでは，経由都市名と距離（あるいは所要時間）データを提示するように改良した。

	プログラムの機能	入力データ	出力データ
1.1	データ入力・ファイル操作	都市区間データファイル	都市区間データファイル
1.2	Generate_Matrix	都市区間データ	隣接行列
1.3	Dijkstra_Algorithm	隣接行列	最短経路
1.4	統合プログラム	都市区間データファイル	最短経路

§ 2．プログラム開発のフェーズ

ここでは，次のような段階的な手順でソフトウェアを開発した。

(1) phase.1　隣接行列から最適経路を探索するプログラムの開発

ダイクストラによるアルゴリズムによって，C 言語でソースコードを作成する。これはこのプロジェクトの中心となるアルゴリズムである。

インプット・データは，2 次元配列の初期値として，隣接行列のデータをプログラム中に書き込まれる（配列の初期値として記述する）。

出力情報は，出発点から各配送都市へのすべての経由地のリストが書き出される。

(2) phase.2　隣接行列をデータとしてファイルから読み込むことができるように改良する。

これによって，検索対象とする地域のデータを，プログラムとは独立させる。プログ

ラムとデータを独立させることによって，別の地域の問題に対しても，プログラムを変更することなく適用できるようになる。完成したプログラムは，そのまま実行でき，検索対象地域データの隣接行列のみを取り換えばよい。これによって，いくつかの異なった地域データについて，クリティカルパスを探索できる。

隣接行列は，Excelを使って人力で作成する。それに基づいて，隣接行列データを作り，テキストファイルとして読み込ませる。対象地域の都市数が10都市なら，10×10の2次元配列となる。間違いなく隣接行列を作り上げるのは，Excelを使うとして人力ではかなり面倒な作業である。20都市程度までが間違えずにできる限界である。

⑶ phase.3　都市間データ（テキストファイル）を読み込み，隣接行列を生成するプログラムを開発する。

隣接行列を人力で作成することは，対象とする地域が小規模の場合は，比較的容易であるが，20都市以上になるとその隣接行列を人手で作ることはかなり大変である。そこで筆者は，対象地域データが多数の都市になっても，その都市間データをインプットして，プログラムの中で隣接行列を生成するサブ・プログラム（generate_mx）を開発した。ここではテーブル・ルックアップ法によって，該当区間のデータを隣接行列に作り上げることができる。

都市間データは，図２のような形式のデータであり，地図から拾うことが容易である。

このプログラムの開発によって，データの入力間違いや，２次元配列上の位置の誤りなどを防ぐことができる。

インプット・データは，地図上の都市間の輸送時間ないしは距離データを隣接行列として表現されたデータである。地図上の所要時間または距離データから，隣接行列を作るのは第一段階（phase.1）では人力で行った。しかし隣接行列（Adjacent Matrix）を人力で作り出すことは，都市数が10程度までならできるが，20以上の都市を結ぶネットワークではかなり大変である。そこで，都市間距離あるいは都市間所要時間をデータ・ファイルから読み込んで，隣接行列を生成するプログラムを開発した。

出発都市	到着都市	距離データ

図２　隣接都市間データの形式

出発地と到着地とは，直接的に隣接している都市であり，距離データはその都市間距離である。ここで都市間距離の代わりに所要時間とすれば，最短所要時間を求めることができ，距離データなら距離の最短経路を求めることが可能となる。

⑷ phase.4　隣接都市データをテキスト・ファイルとして読み込み，プログラム内部で隣接行列を生成し，それによって最短経路を探索するプログラムに改良する。

phase.4 では，phase.1 のダイクストラ・アルゴリズムを主軸に置き，その前処理として phase.3 で開発した都市区間データから隣接行列を生成するプログラムを一つの複合的なプログラムとして組み込む。これによって，教科書的な比較的小さなモデル・データから，四国４県の主要都市（約48都市）や，東日本の主要都市（約200都市）の実用的な都市間データでも，一つのプログラムで最適経路を探索することができるようになった。

§ 3．開発のためのデータ・モデルの段階

また，開発の手順として，比較的小さなサンプルデータから実用データまで，次のような段階に分けて，順次大規模化・高度化してゆくことにした。

データ・モデル

⑷ 開発用テストデータ：10都市サンプルモデル，｛a, b, c, d, e, f, g, h, I, j｝市とする。
開発用の教科書的サンプルモデル。理論上の架空地域モデル。

⑹ 開発用テストデータ２：10都市のデータ，
金澤・富山・上越・長岡・東京・高山・松本・八王子・名古屋・静岡
日本の中部地方と関東地方の主要都市（主に県庁所在地+アルファ）を対象エリアとする輸送モデルを考える。アルゴリズムとしては⑷で開発したプログラムをそのまま適用し，妥当な結果が得られるかどうかを検証する。このモデルでは，日本海・新潟・関東ルートと，中部山岳横断ルート，および東海道ルートの３経路が考えられる。

⑹ 四国の都市間データ：48都市（図３）のモデル
四国の主要都市・町・村など，48ポイント（節: ノード・ポイント）のネットワーク。
図３では 三崎港を起点とし，四国内の各都市（市町村）に到達するための最適経路と，所要時間を示す。データを区間所要時間データから，距離データに変更すれば，同じプログラムで最短距離の探索ができる。
四国の都市（市町村）は，大半が海岸線にそって形成されている。しかも四国中央には，険峻な山岳地帯があり，道路網は山岳地帯では遮断されている。近年高速

道路の建設・開通が進んではいるが，まだ高速道路やトンネルによって直接的に結ばれている都市は多くはない。本稿では，高速道路網の整備以前の状態の一般国道を利用することを前提に考えた。高速道路の利用は想定せず，一般主要道を利用しての配送を考えることにした。これは対象の地域を単純なモデルとするためである。

関東平野などでは，主要都市間をpoint to pointでダイレクトにアクセスできるが，高速道路網を考えないとすると，四国は，主要都市間をpoint to pointでダイレクトにアクセスすることはまれで，海岸線を迂回して到達するというルート特性がみられる。ある都市から別の都市に内陸で直接到達できるケースは，隣接都市以外は，まれになるという地形的特性がある。

この状態もプログラム上のソリューションにどう反映されるか，興味がもたれる。

(d) 東日本の主要都市の約200都市データ

次に，青森県の北端から，関東・東京に向かって，主要都市間の距離データを地図から拾い，約200都市の都市区間データを使ってクリティカル・パスを求めてみた。

ここでも，一般国道を利用するモデルで探索を行った。実際には，今日の日本では，近距離は一般国道を利用し，遠距離は高速道路を利用するのが普通である。

しかし高速道路と一般国道を組み合わせて利用するデータ・モデルは，このアルゴリズムだけでは解を求めることができないため，とりあえず次の課題としてここでは単純なモデルだけを実験することにした。

プログラムを適用して得られた実験結果は，想定通りの最適経路が得られた。プログラムは　図３に示すプログラムがそのまま適用できる。

８都市モデルあるいは　10都市モデルは，プログラム開発用のテストデータである。これを使って最初のversionのプログラムを完成し，それにより実用的なデータを使用してアルゴリズムを検証しつつバージョンアップしてゆく。(b)段階の10都市モデルでデバッグが完了すれば，(c)四国モデル，(d)本州モデルは，データを入れ替えただけで本質的にはプログラム（ロジック）を改良せずともそのまま実行できる。これはプログラムの汎用性を考えて設計したからである。

§ 4．考察

ある地点から，別の地点までの最適なルート（クリティカルパス）を探索することは，最近のナビゲーション・システムではすでに実用化され，自動車に装備されている。さ

らにインターナビ（携帯電話を経由してインターネットと接続されているナビゲーション・システム）では，道路混雑状況などをリアルタイムに情報収集でき，その時々に即応して最適なルートを運転者に提示できる。最新のカー・ナビゲーションではこのように優れた機能が実用的に供されている。それに対して，本研究では，出発地と目的地を１対１としてその区間のクリティカルパスを探索するだけではなく，任意の出発地から全市区町村へのそれぞれのクリティカルパスを探索する。いわば自動車搭載のナビゲーションが個別の自動車の運行のための最適経路の探索をするのに対して，本プログラムでは企業体が全地域に向けての配送計画を立案するときに最適な経路を探索することができる。

このプログラムを作成していたのは，2010年秋から2011年冬にかけての３ヶ月であった。ちょうどプログラムが完成した直後，2011年３月11日に，東北・東日本大震災が発生し，三陸地方の海岸線の道路が広域にわたって不通となった。個別の自動車に搭載されているナビゲーションシステムでは，ある地点から目的地まで，point to pointの経路を探索することには向いているが，地震・津波・原子力発電所の事故などによる広域的な道路網の不通に対しては，多数の配送車両をどの経路で目的地に向かわせるか巨視的に戦略を立て，経路を指示するための探索をする場合に，このダイクストラ・アルゴリズムによる広域的な経路探索が有効である。

参考文献

⑴Ronald H. Ballou, Business Logistics/Supply Chain Management (5th Edition), Prentice Hall, 2003

⑵奥村晴彦「コンピュータ･アルゴリズム事典」技術評論社　昭和62年10月初版
第９章　グラフ理論，「隣接行列」「最短路探し（ダイクストラのアルゴリズム）」pp278-285
pascal 言語によるプログラム

⑶奥村晴彦「C言語によるアルゴリズム事典」技術評論社　平成３年２月　初版
最短経路問題（shortest path problem），Dijkstraのアルゴリズム，pp83-84

⑷河西朝雄「C言語による　はじめてのアルゴリズム入門」技術評論社　平成４年初版　第７章　グラフ，7-5 最短路問題　ダイクストラ法　pp337-334

⑸河西朝雄　改訂版「C言語によるはじめてのアルゴリズム入門」技術評論社，平成４年初版　第７章　グラフ，7-5 最短路問題　ダイクストラ法　pp353-359

⑹小倉久和 「はじめての離散数学」 近代科学社，2011年３月初版　11章　木とグラフ　最適探索，ダイクストラのアルゴリズム　pp178-179

図３　最短経路探索プログラムと四国モデルの実行結果

```
/*      File=C_Logistics_c¥¥Dijkstra_ver_7.c        2011/5/2,4,6,8,13 */
/*                                                                          */
/*      File=C_Logistics_c¥¥Genarate_matrix_5.c                             */
/*      File= Logistics_1¥Logistics_5.c       by Hachiya, 2010/8/17         */
/*      金澤==>東京  ロジスティクス・ネットワーク       2011/1/07,08,09       */
/*      Adjacent Matrix Generator for Logistics Network        2011/1/15,3/22,完成     */
/*      2011/3/27,28,29,30,31,4/1,2,3,4,5  完成            */
/*      2011/4/6,7,8,9,10,11,12,14,15,up,20,21,up,22       */
/*                          */
#include    <stdio.h>
#include    <string.h>
//#include     <iostream.h>

#define     P     100       //   300
#define     M     9999      //   -1
// #define     N     200

char    city[P+3] [2] [10]    = { {"/","/"}, {"/","/"} };
//                = { {"金澤","富山"},      {"富山","上越"},      {"上越","長岡"},      {"長岡","東京"},
//                  {"金澤","高山"},      {"高山","松本"},      {"松本","八王子"}, {"八王子","東京"},
//                  {"金澤","名古屋"},    {"名古屋","静岡"},    {"静岡","東京"},
//                  {"富山","高山"},      {"高山","名古屋"},    {"上越","松本"},      {"長岡","八王子"},
//                  {"静岡","松本"},      {"静岡","八王子"},    {"松本","名古屋"}, {"金澤","米原"},
//                  {"米原","名古屋"},    {"米原","京都"},      {"/","/"}    };

//char    city_tbl[P+3][10]  =  {"～","金澤","富山","上越","長岡","東京","高山","松本","八王子","名古屋","静岡
","","",""};
char    city_tbl[P+3][10] = {"～","--","--"};
char    city_T[P+3][4];
char    city_T2[P+3][6];
int     n_city_tbl = 0;

int     a[P+3][P+3] ={ 0 };     /*  隣接行列  */
int     b[P][P]    ={ 0 };

int       x[P+3];
int       id[P+3];
int       icond    ,icnt    ,dbglvl = 4;

int    data_in( int    icnt    );
//                                    char    file_name[],    char    city[N][8],    char city2[N][8],    int    b[N][N] )
//
void    genarate_mx( int    icnt );
//
int     output_file( int    icnt );
//
int     dijkstra( int    icnt );
//
int     prt_a_matx(    int    icnt, int    jcnt );
//
//
```

```
void   main(void)
{
    int    i, j, k=0, icnt=0 ,n_of_node=10   ,n_node=0;
    char   loop;

    char              f_name[160]={"..."   },   dummy[80]={"..."},   dummy2[80]={"..."},   dummy3[80]={"..."},
dummy4[80]={"..."};

    char     *file_name ={"C:¥¥Documents and Settings¥¥Owner¥¥My Documents¥¥data_file¥¥f1_全
国.dat"};
//  char     *file_name ={"C:¥¥Documents and Settings¥¥Owner¥¥My Documents¥¥data_file¥¥F_金
澤.dat"};
//          char                    *file_name      ={"C:¥¥Documents      and      Settings¥¥Owner¥¥My
Documents¥¥Logistics_1¥¥Kanazawa.dat"};
//                              "F:¥¥Logistics_1¥¥kanazawa.dat"

    printf("¥nMain program ファイル入力，隣接行列生成 Start, by H.Hachiya   2011/2/28");
    for(i=0;i<P;i++)
        for(j=0;j<P;j++)    a[i][j] = -1;
    for(i=0;i<P;i++)   strcpy( city[i][0],   "..." );
    for(i=0;i<P;i++)   strcpy( city_T[i],    ".." );
    for(i=11;i<P;i++)   strcpy( city_tbl[i], ".." );

    printf("¥nType-in Debug Level  ( 0  ～  7 )=");
        scanf("%d", &dbglvl );

    getchar();

    icnt = data_in( icnt );   //   データ入力
//
    getchar();
//
    genarate_mx( icnt );
//
    printf("¥n 隣接行列の生成・完成¥n");
    getchar();
    n_of_node = n_city_tbl;
        loop = 'y';
        while( loop == 'y'   )   {
      icond   = dijkstra( icnt );
            printf("¥n 別の始点からの経路の検索をしますか (y/n)=");   scanf("%s",&loop );
        }
//
//    output_file(    n_of_node );
//
    printf("¥n ダイクストラ・アルゴリズム 完了");
    printf("¥nProgram Normal End ... 正常終了¥n");
    getchar();
    return ;
}
/*  ================================================   */
/*     File=C_logistics¥¥Data_in.cpp      by hachiya,   2011/3/08     */
/*     Data input from File,   全国都市区間データ       2011/3/15     */
/*     都市区間データを読み込んで、隣接マトリクスを生成する。       */
```

```
//
int   data_in( int   icnt   )
//                                  char   file_name[],   char   city[P][8],   char city2[P][8],   int   b[P][P] )
{
    int         i=0;
    int         j, fn;
    FILE        *fpFile;
    char              f_name[160]={"..."   },   dummy[80]={"..."},   dummy2[80]={"..."},   dummy3[80]={"..."},
dummy4[80]={"..."};
//
          char   *file_name[10]
            = { "F:¥¥data_file¥¥f01_本州テスト.dat"
                ,"F:¥¥data_file¥¥f04_四国.dat"
                ,"C:¥¥Documents and Settings¥¥Owner¥¥My Documents¥¥data_file¥¥f04_四国.dat"
                ,"C:¥¥Documents   and   Settings¥¥Owner¥¥My   Documents¥¥data_file¥¥f01_ 本 州 テ ス
ト.dat"
                ,"C:¥¥Documents and Settings¥¥Owner¥¥My Documents¥¥data_file¥¥f0_本州テスト.dat"
                ,"C:¥¥Documents and Settings¥¥Owner¥¥My Documents¥¥data_file¥¥f1_本州.dat"
                ,"C:¥¥Documents and Settings¥¥Owner¥¥My Documents¥¥data_file¥¥F_金澤.dat"
                ,"C:¥¥Documents and Settings¥¥Owner¥¥My Documents¥¥Logistics_1¥¥Kanazawa.dat"
                ,"F:¥¥Logistics_1¥¥kanazawa.dat"
          };

    printf("¥nData-in subprogram Start   ¥n");
    printf("¥n 入力ファイルを下記から選択して、番号で指定してください");
          for(i=0;i<8;i++)   printf("¥n%2d   <%-s>",i,file_name[i] );
          printf("¥n  選択したファイルの番号  = ");
          scanf("%d", &fn );

//          getchar();
//  fpFile = fopen("C:¥¥Logistics_1¥¥kanazawa.dat","r" ) ) != NULL )
//        fopen( file_name, "r" );

    if( ( fpFile = fopen( file_name[fn] ,"r" ) ) != NULL )
        {
          printf("¥n----<%s>¥n---- File opend -----¥n",file_name[fn] );   // getchar();
          icnt = 0;
          fscanf( fpFile, "%41s %s %s", &dummy,&dummy2,&dummy3 );
          icnt++;
          if( dbglvl > 2 ) {
          printf("¥n%4d <%s> <%s> <%s>", icnt,dummy, dummy2, dummy3 );
          getchar();
          }
//
          i = 0;
          while( fpFile != NULL ) {
              fscanf( fpFile, "%4d %12s %12s %10d %10d",   &id[i] ,&city[i][0], &city[i][1], &x[i], &j   );
              if( id[i] < 0   )   break;
              if( dbglvl > 3 ) {
              if( i % 10 == 0 )   getchar();
                  printf("¥n%4d          %06d     :     <%-12s>     :     <%-12s>          %06d     %6d",
icnt,id[i],city[i][0],city[i][1],x[i],j );
                  }
              icnt++;   i++;
```

```
            }
          fclose ( fpFile );
        }
      else
       {   printf("¥n¥n<%s>¥n ---- File could not opened, 読み込むべきファイルが存在しない？
===",file_name);
           getchar();
       }
    printf("¥n¥nData-in Sub-program Normal End ... 読み込み正常終了   rec=%4d¥n", icnt   );
    return   icnt;
}
//
/*     File=C_logistics¥¥Data_in.cpp       by hachiya,   2011/3/08      */
/*     Data input from File,   全国都市区間データ        2011/3/15      */
/*     都市区間データを読み込んで、隣接マトリクスを生成する。       */
//
/*     File=C_Logistics_c¥¥Genarate_matrix_5.c                                    */
/*     File= Logistics_1¥Logistics_5.c       by Hachiya,     2010/8/17                */
/*     金澤==>東京  ロジスティクス・ネットワーク          2011/1/07,08,09         */
/*     Adjacent Matrix Generator for Logistics Network   2011/1/15,3/22,完成    */
/*     2011/3/27,28,29,30,31,4/1,2,3,4,5  完成                */
//
//     Subroutine Sub-progarm     Genarate Matrix    ========================
void   genarate_mx( int icnt )
{
    int     i, j, k, jj, kk,    nn , nop=0, kcnt=23, jcnt;

    char   node[26][2]        = {"A" ,"B","C","D", "E","F","F","G","G",    "H","H","H","I","J","J","J" };
    char   s_node[P+P] [2] = {" " ," "," "," "," "," ","1","2","1","2",    "1","2","3","I","1","2","3"};
    char   cnect[P+P][2]     = {" " ,"A","A","A", "B","C","E","D","F",    "F","I","G","E","G","H","I" };

//     int     time[P+P]   = {   90, 84, 84, 180,      138, 90, 60, 126,       348, 48, 150,
//                                  66, 156,       120, 132,    132, 48    };
//     int     x[P]    = {   50, 80, 60, 210,      300, 120, 120, 120,       210, 150, 150,
//                             90, 150,    120, 132, 180, 150,/*180,*/ -111, -999, -888, -777, -666, -99, -99 };
    nn     = n_city_tbl;
    jcnt = icnt;
//     getchar();
    printf("¥n Subroutine Sub-progarm     Genarate Matrix    2011/3/30");
//     printf("¥nFile= Logistics_1¥¥Logistics_5.c       by Hachiya, 2020/8/17, 2011/1/07,    ");
//     printf("¥n 金澤～東京   間   道路網    経由地データ    %3d=nn    %3d=icnt",nn,icnt);
    printf("¥n 都市テーブル    初期状態      %4d=n_city_tbl¥n", n_city_tbl );
    if( dbglvl > 5 ) {
        for(i=0;i<=n_city_tbl;i++) {
            if( i % 5 == 0 ) printf("¥n");
            printf("%4d <%-8s>",i,city_tbl[i] );
           }
        getchar();
       }
          if( dbglvl > 2 ) {
        printf("¥n 都市間データ .....  入力データ         件数=%2d", icnt);
        getchar();
        for(i=0;i<=icnt;i++) {
            if( city[i][0][0] == '/' ) break;
```

```
            if( dbglvl > 5 ) {
                if( i % 10 == 0 ) getchar();
                printf("¥n%3d   <%-8s> == <%-8s>,   %4d",i, city[i][0], city[i][1], x[i] );
                        }
             }
       getchar();
       icnt = i - 1;   /*   icnt : 実効区間データ件数   */
          }
          /*      */
    printf("¥nAdjcent Matrix Table Initialize   1 -----    :");
    for(i=0;i<=icnt;i++ )   {
        for(j=0;j<=icnt;j++ ) {
            a[i][j] = M;
          }
        }
    for(i=0;i<=icnt;i++)   a[i][0] = 0;
    for(j=0;j<=icnt;j++)   a[0][j] = 0;
    for(i=0;i<=icnt;i++)   a[i][i] = 0;
    jcnt = nn + 2;

    if( dbglvl > 6 )     prt_a_matx( icnt,   jcnt );
//    getchar();
//
//     都市間データから  隣接行列  に区間データを書き込む
    n_city_tbl = nn;
    jj = 0; kk = 0;
    for(i=0;i<=icnt;i++) {
        for(j=1;j<=n_city_tbl;j++) {
            if( strcmp( city[i][0] , city_tbl[j] ) == 0 ) {
                jj = j;
                goto L_11;
               }
            }
//*      Addition to city_tbl[ next ]     テーブルに未登録の都市名が出現した   */
        if( j > nn ) {
            if( dbglvl > 4 ) {
                printf("¥n*** New  都市  %3d=i, <%s>=新都市    <%s> %3d =nn,%3d =j, <%s>"
                                ,i,city[i][0],city[i][1],nn,j,city_tbl[nn-1] );
                        }
             n_city_tbl++;
             strcpy( city_tbl[ n_city_tbl ],   city[i][0] );
             //              out                    <=====   in
             jj++;
             nn++;
             a[nn][nn]      =   0;
                    }
      /*    */
L_11:
    for(k=1;k<=n_city_tbl;k++) {
        if( strcmp( city[i][1] , city_tbl[k] ) == 0 ) {
            kk = k;
            goto L_22;
            }
        }
```

```
        /*   Addition to city_tvl[ next ]   */
        if( k > nn ) {
            if( dbglvl > 4 )
                printf("¥n+++ New 都市 -2 %3d=i, <%s>, %3d =j, %3d =nn, <%s>=新都市    <%s>"
                                                ,i,city[i][0],j,nn,city[i][1],city_tbl[nn-1] );
             n_city_tbl++;
             strcpy( city_tbl[n_city_tbl],   city[i][1] );
             //              out                    <=====   in
             kk++;
             nn++;
         };
L_22:
       if( dbglvl > 3 )
            printf("¥n%3d=i, %3d=jj, %3d =kk,   <%-8s>   <%-8s> %5d", i,jj,kk, city[i][0], city[i][1], x[i] );
        a[j][k]   =   x[i];
        a[k][j]   =   x[i];
        a[i][i]   =   0;
        nop = 1;
       }
      printf("¥nAdjcent Table Created - OK     =======      :");
      if( dbglvl > 3 ) printf("¥n 都市間テーブル     %3d=都市数",nn);

           for(i=0;i<=n_city_tbl;i++) {
        if( i % 10 == 0 ) getchar();
                    city_T[i][0]   =   city_tbl[i][0];
                    city_T[i][1]   =   city_tbl[i][1];
                    city_T[i][2]   =   '¥0';   //   NULL;

                    for(j=0;j<4;j++)   city_T2[i][j] =   city_tbl[i][j];
        city_T2[i][4] = '¥0';   // NULL;

              if( dbglvl > 3 )   printf("¥n%3d <%-8s> <%-2s>",i,city_tbl[i],city_T[i] );
       }
    city_T[i+1][0] = '-';
    city_T[i+1][1] = '¥0';
    city_T[i+1][2] = '¥0';
//
//
//
//* ======================================   ====================================== */
//* ======================================   ====================================== */
//
//     getchar();
    jcnt = nn + 1;
    printf("¥nSuccesed Adjcent Table -- 3 ------     :");
    if( dbglvl > 3 )     prt_a_matx( n_city_tbl,   jcnt );
//
    if( dbglvl > 3 )    {
        printf("¥nupdated   都市間   道路網    経由地データ   %3d=nn   %3d=icnt¥n",nn,icnt);
        for(i=0; i<=icnt+5; i++) {
            if( city[i][0][0] == '/' ) break;
            printf("¥n %2d, <%-8s> = <%-8s>,%6d",i, city[i][0], city[i][1], x[i] );
          }
       if( dbglvl > 4 ) {
```

```
            printf("¥n 都市間テーブル     %3d=都市数",nn);
            for(i=0; i<=nn+5; i++) {
            if( i % 10 == 0 ) getchar();
            printf("¥n%3d   <%-8s>   <%-4s>   <%-2s>",i,city_tbl[i],city_T2[i],city_T[i] );
           }
         }
      }
    i = output_file(   n_city_tbl );      //
    printf("¥nSubroutine Subprogram   < Genarate Matrix >   Normal Exit");
    return;
}
/*     End of progrm   ================================================================   */
/*     隣接行列をデータファイルとして出力し保存する     */
/*                                                    */
int    output_file( int   n_of_node )
{
    int   i, j,   icnt;

    FILE      *fpFile;
    icnt = 0;

    if(( fpFile = fopen("C:¥¥Logistics_1¥¥f1_o_本州.dat","w" ) ) != NULL )
        {
         fprintf(  fpFile,  "¥nC:¥¥Logistics_1¥¥f1_o_本州.dat,    隣接行列データ      %4d=n_of_node¥n",
n_of_node );
          printf(                "¥nC:¥¥Logistics_1¥¥f1_o_本州.dat,    隣接行列データ      %4d=n_of_node¥n",
n_of_node );
         getchar();
         for( i=0; i<=n_of_node; i++ )   fprintf( fpFile, "%10s", city_tbl[i] );
         for( i=0; i<=n_of_node;i++ )   {
              fprintf( fpFile, "¥n%4d¥n", i );
              for( j=0; j<=n_of_node; j++ )   {
              fprintf( fpFile, "%8d", a[i][j]   ); }
             }
         fclose ( fpFile );
        }
       else {   printf("¥n¥nLogistics_1¥¥f1_o_本州.dat ---- File cold not opened, ===");
                 getchar();
             }
    return   icnt;
}
//          隣接行列  a[N][N]   のチェックプリント  //
int   prt_a_matx( int   icnt, int   jcnt )
{
    int   icnd=0;
    int   i, j, k, js=0,   je=1,   jadd= 13;

    printf("¥n Print matrix   a[N}[N} ******%4d=icnt   %3d=jcnt ***",icnt, jcnt);
    for(k=je;k<=icnt;k=k+jadd ) {
         je = js + jadd;
         if( icnt <= je ) je = icnt;
         if( je < js )   break;
         printf("¥nPrt_a_matx--- %3d=k, %3d=js, %3d=je, %3d=icnt¥n            ",k,js,je,icnt);
         for(j=js;j<=je;j++)     printf("%5d", j );   printf("¥n             ");
```

```
        for(j=js;j<=je;j++)     printf("    %2s ", city_T[j] );
        for(i=0;i<=icnt;i++ )   {
            if( i % 10 == 0 )   getchar();
            printf("¥n%3d %-2s ", i,city_T[i] );
            for(j=js;j<=je;j++ )     printf("%5d", a[i][j] );
           }
        if( je <= icnt )     js = je+1;
       }
    return   icnd;
}
/*    file =Logistics¥L_Ballou__7_2_B      金沢～東京ルート探索_修正データ*/
/*    file=logistics¥A7_5_v22 最短経路探索.c     by 蜂谷  2011/01/01          */
/*    最短経路探索問題    ダイクストラ法                          2010/10/29,30     */
/*    v22.0                                                        2011/1/1,2 完     */
/*    B1.0                    改良中  1/30,2/01,02,10,11,  4/07              */
//#include   <stdio.h>
//#include   <string.h>
//#define   N   10      /* 節点の数    ノードポイント    */
//#define   M   9999

//   char    node[N+3][8]={"～","金澤","富山","上越","長岡","東京","高山","松本","八王子","名古屋","静岡
","--","==" };   /* 節点名  */
//   char    city_tbl[P+3][10]
//                            {"～",  金,  富,  越,  長,  東,"高","松","八","名","静",    /* 節点名  */
//int   a[N+1][N+1] ={ { 0,    0,    0,    0,   0, 0,     0,    0,    0,     0,    0   }, /* 隣接行列  */
//                          { 0,    0, 90,    M,    M, M, 138,    M,    M, 348,    M   },
//                          { 0, 90,    0, 84,    M, M,    66,    M,    M,     M,    M   },
//                          { 0,    M, 84,    0, 84, M,     M, 120, M,     M,    M   },
//                          { 0,    M,    M, 84,    0, 180, M,    M, 240,    M,    M   },
//                          { 0,    M,    M,    M, 180, 0,    M,    M, 126,    M, 150 },
//                          { 0, 138, 66, M,    M,    M,    0, 120,    M, 156,    M   },
//                          { 0,    M,    M, 120, M,    M, 120, 0,    120,    M, 132 },
//                          { 0,    M,    M,    M, 240, 120, M, 120,    0,    M, 150 },
//                          { 0, 348, M,    M,    M,    M, 156, M,     M,    0,    120 },
//                          { 0,    M,    M,    M,    M, 150, M, 180, 150, 120,    0 },
//                         };
int    dijkstra( int   icnt )
{
int    dist[P+3][P+3];
int    sumd[P+3][P+3];
int    n_of_path[P+3] = { 0 };
char   path[P+3][P+3][10];

//int     main(void)
//{
    int   i, j, k, p, start, arrive, min;
    int   leng[P+3];                    /* 節点までの距離  */
    int   v[P+3];                       /* 確定フラグ      */
    int   index[P+3];                   /* 前の節点へのポインタ  */
//     char node[P+3][8] = {"～","金","富","越","長","東","高","松","八","名","静","--","==" };   /* 節点名  */

    printf("¥n file=logistics¥¥A7_5_v31 最短経路探索  Dijkstra_Algorithm.c   by 蜂谷  2011/01/01, 4/07");
    printf("¥n    最短経路探索問題    ダイクストラ法    ");
    printf("¥n 都市データから、出発点と到着点を番号で指定してください¥n");
```

```
    for(i=0; i<n_city_tbl;i++ ) {
        printf("%3d %-6s",i,city_tbl[i] );
                    if( i % 8 == 0 ) printf("¥n");
         }
         printf("¥n 出発点  => "); scanf("%d",&start );
    printf("¥n 到着点  => "); scanf("%d",&arrive );

    for( i=1; i<=P; i++ ) {
        leng[i] = M;    v[i] = 0; n_of_path[i] =0;
        }
    for( i=0; i<=P; i++ ) {
        for(j=0;j<=P;j++)   { sumd[i][j] = 0; dist[i][j] = 0; }
       }
    for(i=0;i<P;i++)   for(j=0;j<=P;j++)   strcpy( path[i][j],    "～" );
    leng[start]   = 0;
    index[start] = 0;     /*  始点はどこも示さない  */
//
    printf("¥n   節点= %2d,   節点名=<%s>   から、最短の節点を探索する。¥n", start, city_tbl[start] );
    printf("¥n   n_city_tbl=%3d     ¥n",n_city_tbl );
    getchar();
    printf("¥n /*  最小の節点を確定する。   V[P] :  確定フラグ   */");
    for( j=1; j<=n_city_tbl; j++)   {
        min = M; // P;        /*  最小の節点を探索する  */
        for( k=1; k<=n_city_tbl; k++ ) {
            if( v[k] == 0 && leng[k] < min ) {
                p = k; min = leng[k];
                }
            }
        v[p] = 1;
        if( dbglvl > 5 ) {
             printf("¥nv[%2d] =%4d : ",p, v[p]);
            for( k=1;k<=n_city_tbl; k++ )   printf("%4d", v[k] );
            getchar();
                }
        if( min == M ) {
            printf("     グラフは連結でない     j=%3d     <%s>=city¥n",j,city_tbl[j] );
            getchar();
            return (0);
           }
        /* p を経由して  k  に至る長さが、それまでの最短路より小さければ   更新  */
        i = 0;
        for( k=1; k<=n_city_tbl; k++ ) {
            if( ( leng[p] + a[p][k] ) < leng[k] ) {
                    leng[k] = leng[p] + a[p][k];
                    index[k] = p;
                    i++;
                  }
            }
        }
         printf("¥n  節点=  %2d,    節点名=<%s>    から、最短の節点を探索する。[完成した]¥n",  start,
city_tbl[start] );
    getchar();
    printf("¥n   最短ルートの表示.....");
    printf("¥n   分 :  到達点   <--  経由点¥n");
```

```
    for( j=1; j<=n_city_tbl; j++ ) {                /*  最短ルートの表示  */
        printf("¥n%4d : %3d", leng[j],j );     //     ノードのID
        p = j;
        while( index[p] != 0 ) {
            printf(" <-- %2d ", index[p] );
            p = index[p];
           }
//       if( j > 14 ) {printf("¥nj=%3d    pause ",j ); getchar(); };
        printf("¥n         :%4s", city_T2[j] );     // ノードポイント,都市名
        p = j;
        i = 0;
        strcpy( path[j][i], city_tbl[ p ] );
        while( index[p] != 0 ) {
            if(   city_tbl[ p ] [0] == '.' ) break;
            p = index[ p ];
            printf(" <--%4s", city_T2[ p ] );
            i++;
            strcpy( path[j][i], city_tbl[ p ] );
           }
        p = j;
        i = 0;
        printf("¥n%4d :%4d", leng[j], a[p][index[p]] ); // 次のノードまでの所要時間
        dist[j][i]   = a[p][index[p] ];
//
        while( index[p] != 0 ) {
            p = index[ p ];
            printf(" <--%3d ", a[p][ index[p] ] );
            dist[j][i]   = a[p][index[p] ];
            i++;
           }
        dist[j][P]   = i;
        n_of_path[j] = i;
        p = j;
        k = 0;
//
        printf("¥n%4d :%4d", leng[j], a[p][index[p]] );   //   所要時間の累計
        for(k=0; k<n_of_path[j] ; k++ )   printf(" <==%3d ", dist[j][k] ); printf(" dist[j][k]  区間時間");
        for(k=n_of_path[j] ; k>=0; k--)   dist[j][k+1] = dist[j][k];
                    dist[j][0] = a[p][index[p]];
        for(k=n_of_path[j] ; k>=0; k--)   sumd[j][k] = sumd[j][k+1] + dist[j][k];
        k = 0;
        printf("¥n sum :%4d", sumd[j][0]   );      /*  所要時間累計   */
        for(k=1; k<n_of_path[j] ; k++ )   printf(" <==%3d ", sumd[j][k] );
        printf("¥n ");
          getchar();
        // if( j % 5 == 0 ) getchar();
       }
    if( dbglvl > 2 ) {
    getchar();
    printf("¥n    分 : 到達点  <--  経由点¥n");
    for(j=1;j<=n_city_tbl;j++ ) {
        printf("¥n%3d =j %4d   ",j,n_of_path[j] );
        for(k=0;k<=n_of_path[j];k++)   printf("%-4s ", path[j][k] );
       }
```

```
    getchar();
    printf("¥n dist[j][k] ");
    for(j=0;j<=n_city_tbl;j++ ) {
        printf("¥n%3d := j :",j);
        for(k=0;k<=n_of_path[j];k++)   printf("%5d", dist[j][k] );
       }
    getchar();
    printf("¥n sumd[j][k] ");
    for(j=0;j<=n_city_tbl;j++ ) {
        printf("¥n%3d := j :",j);
        for(k=0;k<=n_of_path[j];k++)   printf("%5d", sumd[j][k] );
       }
         }

    printf("¥nDijkstra Algorithm   program Normal End");
    getchar();
    return (0);
}

Main program ファイル入力，隣接行列生成 Start, by H.Hachiya   2011/2/28
Type-in Debug Level （0 ～ 7）=3

Data-in subprogram Start

入力ファイルを下記から選択して、番号で指定してください
 0   <F:¥data_file¥f01_本州テスト.dat>
 1   <F:¥data_file¥f04_四国.dat>
 2   <C:¥Documents and Settings¥Owner¥My Documents¥data_file¥f04_四国.dat>
 3   <C:¥Documents and Settings¥Owner¥My Documents¥data_file¥f01_本州テスト.dat>
 4   <C:¥Documents and Settings¥Owner¥My Documents¥data_file¥f0_本州テスト.dat>
 5   <C:¥Documents and Settings¥Owner¥My Documents¥data_file¥f1_本州.dat>
 6   <C:¥Documents and Settings¥Owner¥My Documents¥data_file¥F_金澤.dat>
 7   <C:¥Documents and Settings¥Owner¥My Documents¥Logistics_1¥Kanazawa.dat>
 選択したファイルの番号  = 2

----<C:¥Documents and Settings¥Owner¥My Documents¥data_file¥f04_四国.dat>
---- File opend -----

    1 <File=F:¥¥data_file¥¥f04_四国.dat> <2011/3/4,4/7,5/6,> <四国_所要時間(分)デ
ータ>

Data-in Sub-program Normal End ... 読み込み正常終了   rec=   79


  Subroutine Sub-progarm     Genarate Matrix   2011/3/30
都市テーブル   初期状態          0=n_city_tbl

都市間データ ..... 入力データ        件数=79


Adjcent Matrix Table Initialize   1 -----    :
Adjcent Table Created - OK    ========      :
```

```
Succesed Adjcent Table -- 3 ------    :

Logistics_1¥f1_o_本州.dat ---- File cold not opened, ===

Subroutine Subprogram   < Genarate Matrix >   Normal Exit
隣接行列の生成・完成


 file=logistics¥A7_5_v31 最短経路探索 Dijkstra_Algorithm.c   by 蜂谷  2011/01/01,
4/07
  最短経路探索問題   ダイクストラ法
都市データから、出発点と到着点を番号で指定してください
  0 ～
  1 三崎     2 八幡浜   3 大洲     4 内子     5 伊予市   6 御三戸   7 松山     8 北条

  9 小松    10 大西    11 鈍川温泉 12 今治    13 糸山    14 西条    15 新浜    16 川之
江
 17 阿波池田 18 観音寺 19 琴平    20 金蔵寺 21 坂出    22 五色台 23 高松    24 湯元

 25 屋島    26 鳴門    27 徳島    28 眉山    29 穴吹    30 小松島 31 見能林 32 清峰

 33 室戸岬 34 野市    35 龍河洞 36 御免    37 土佐山田 38 高知    39 領石    40 大豊

 41 栃の瀬 42 見ノ越 43 貞光    44 佐川    45 須崎    46 面河渓 47 石鎚山 48 窪川

 49 宇和島 50 中村    51 土佐清水 52 足摺岬 53 竜串    54 宿毛    55 御荘    56 船越

 57 滑床
出発点  => 1

到着点  => 20

  節点=   1,  節点名=<三崎>   から、最短の節点を探索する。

  n_city_tbl= 58

 /* 最小の節点を確定する。  V[P]：確定フラグ  */
節点=   1,  節点名=<三崎>   から、最短の節点を探索する。[完成した]


  最短ルートの表示.....
  分：到達点  <-- 経由点

   0：    1
     :三崎
   0：    0
   0：    0 dist[j][k]  区間時間
 sum：    0
```

```
 50：   2 <--  1
    :八幡 <--三崎
 50：  50 <--  0
 50：  50 <==  0  dist[j][k] 区間時間
sum：  50


 80：   3 <--  2  <--  1
    :大洲 <--八幡 <--三崎
 80：  30 <-- 50  <--  0
 80：  30 <== 50  <==  0  dist[j][k] 区間時間
sum：  80 <== 50


 95：   4 <--  3  <--  2  <--  1
    :内子 <--大洲 <--八幡 <--三崎
 95：  15 <-- 30  <-- 50  <--  0
 95：  15 <== 30  <== 50  <==  0  dist[j][k] 区間時間
sum：  95 <== 80  <== 50


130：   5 <--  4  <--  3  <--  2  <--  1
    :伊予 <--内子 <--大洲 <--八幡 <--三崎
130：  35 <-- 15  <-- 30  <-- 50  <--  0
130：  35 <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
sum：130 <== 95  <== 80  <== 50


185：   6 <--  4  <--  3  <--  2  <--  1
    :御三 <--内子 <--大洲 <--八幡 <--三崎
185：  90 <-- 15  <-- 30  <-- 50  <--  0
185：  90 <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
sum：185 <== 95  <== 80  <== 50


155：   7 <--  5  <--  4  <--  3  <--  2  <--  1
    :松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
155：  25 <-- 35  <-- 15  <-- 30  <-- 50  <--  0
155：  25 <== 35  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
sum：155 <==130  <== 95  <== 80  <== 50


185：   8 <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
    :北条 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
185：  30 <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
185：  30 <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
sum：185 <==155  <==130  <== 95  <== 80  <== 50


235：   9 <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
    :小松 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
235：  80 <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
235：  80 <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
sum：235 <==155  <==130  <== 95  <== 80  <== 50
```

```
 215：  10 <--  8  <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
      :大西 <--北条 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 215：  30 <-- 30  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 215：  30 <== 30  <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区
間時間
 sum：215 <==185  <==155  <==130  <== 95  <== 80  <== 50

 235：  11 <--  8  <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
      :鈍川 <--北条 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 235：  50 <-- 30  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 235：  50 <== 30  <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区
間時間
 sum：235 <==185  <==155  <==130  <== 95  <== 80  <== 50


 235：  12 <-- 10  <--  8  <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
      :今治 <--大西 <--北条 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 235：  20 <-- 30  <-- 30  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 235：  20 <== 30  <== 30  <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[
j][k] 区間時間
 sum：235 <==215  <==185  <==155  <==130  <== 95  <== 80  <== 50


 245：  13 <-- 10  <--  8  <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
      :糸山 <--大西 <--北条 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 245：  30 <-- 30  <-- 30  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 245：  30 <== 30  <== 30  <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[
j][k] 区間時間
 sum：245 <==215  <==185  <==155  <==130  <== 95  <== 80  <== 50


 245：  14 <--  9  <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
      :西条 <--小松 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 245：  10 <-- 80  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 245：  10 <== 80  <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区
間時間
 sum：245 <==235  <==155  <==130  <== 95  <== 80  <== 50


 260：  15 <-- 14  <--  9  <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
      :新浜 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 260：  15 <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 260：  15 <== 10  <== 80  <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[
j][k] 区間時間
 sum：260 <==245  <==235  <==155  <==130  <== 95  <== 80  <== 50


 300：  16 <-- 14  <--  9  <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
      :川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 300：  55 <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 300：  55 <== 10  <== 80  <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[
j][k] 区間時間
```

```
 sum：300 <==245  <==235  <==155  <==130  <== 95  <== 80  <== 50


 350：  17 <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--  3  <--  2  <--1
      :阿波 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 350：  50 <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--0
 350：  50 <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 30  <== 50  <==
0   dist[j][k] 区間時間
 sum：350 <==300  <==245  <==235  <==155  <==130  <== 95  <== 80  <== 50


 330：  18 <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--  3  <--  2  <--1
      :観音 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 330：  30 <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--0
 330：  30 <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 30  <== 50  <==
0   dist[j][k] 区間時間
 sum：330 <==300  <==245  <==235  <==155  <==130  <== 95  <== 80  <== 50


 360：  19 <-- 18  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--  3  <--
2  <--  1
      :零平 <--観音 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大洲 <--八
幡 <--三崎
 360：  30 <-- 30  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 30  <-- 5
0  <--  0
 360：  30 <== 30  <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 30  <== 5
0  <==  0  dist[j][k] 区間時間
 sum：360 <==330  <==300  <==245  <==235  <==155  <==130  <== 95  <== 80  <== 50


 350：  20 <-- 18  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--  3  <--
2  <--  1
      :金蔵 <--観音 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大洲 <--八
幡 <--三崎
 350：  20 <-- 30  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 30  <-- 5
0  <--  0
 350：  20 <== 30  <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 30  <== 5
0  <==  0  dist[j][k] 区間時間
 sum：350 <==330  <==300  <==245  <==235  <==155  <==130  <== 95  <== 80  <== 50


 370：  21 <-- 20  <-- 18  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--
3  <--  2  <--  1
      :坂出 <--金蔵 <--観音 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大
洲 <--八幡 <--三崎
 370：  20 <-- 20  <-- 30  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 3
0  <-- 50  <--  0
 370：  20 <== 20  <== 30  <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 3
0  <== 50  <==  0  dist[j][k] 区間時間
 sum：370 <==350  <==330  <==300  <==245  <==235  <==155  <==130  <== 95  <== 8
0  <== 50


 410：  22 <-- 21  <-- 20  <-- 18  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--
```

```
4  <--  3  <--  2  <--  1
     :五色 <--坂出 <--金蔵 <--観音 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内
子 <--大洲 <--八幡 <--三崎
 410:  40 <-- 20  <-- 20  <-- 30  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 1
5  <-- 30  <-- 50  <--  0
 410:  40 <== 20  <== 20  <== 30  <== 55  <== 10  <== 80  <== 25  <== 35  <== 1
5  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum:410 <==370  <==350  <==330  <==300  <==245  <==235  <==155  <==130  <== 9
5  <== 80  <== 50


 410:  23 <-- 21  <-- 20  <-- 18  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--
4  <--  3  <--  2  <--  1
     :高松 <--坂出 <--金蔵 <--観音 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内
子 <--大洲 <--八幡 <--三崎
 410:  40 <-- 20  <-- 20  <-- 30  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 1
5  <-- 30  <-- 50  <--  0
 410:  40 <== 20  <== 20  <== 30  <== 55  <== 10  <== 80  <== 25  <== 35  <== 1
5  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum:410 <==370  <==350  <==330  <==300  <==245  <==235  <==155  <==130  <== 9
5  <== 80  <== 50


 425:  24 <-- 23  <-- 21  <-- 20  <-- 18  <-- 16  <-- 14  <--  9  <--  7  <--
5  <--  4  <--  3  <--  2  <--  1
     :湯元 <--高松 <--坂出 <--金蔵 <--観音 <--川之 <--西条 <--小松 <--松山 <--伊
予 <--内子 <--大洲 <--八幡 <--三崎
 425:  15 <-- 40  <-- 20  <-- 20  <-- 30  <-- 55  <-- 10  <-- 80  <-- 25  <-- 3
5  <-- 15  <-- 30  <-- 50  <--  0
 425:  15 <== 40  <== 20  <== 20  <== 30  <== 55  <== 10  <== 80  <== 25  <== 3
5  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum:425 <==410  <==370  <==350  <==330  <==300  <==245  <==235  <==155  <==13
0  <== 95  <== 80  <== 50


 435:  25 <-- 24  <-- 23  <-- 21  <-- 20  <-- 18  <-- 16  <-- 14  <--  9  <--
7  <--  5  <--  4  <--  3  <--  2  <--  1
     :屋島 <--湯元 <--高松 <--坂出 <--金蔵 <--観音 <--川之 <--西条 <--小松 <--松
山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 435:  10 <-- 15  <-- 40  <-- 20  <-- 20  <-- 30  <-- 55  <-- 10  <-- 80  <-- 2
5  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 435:  10 <== 15  <== 40  <== 20  <== 20  <== 30  <== 55  <== 10  <== 80  <== 2
5  <== 35  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum:435 <==425  <==410  <==370  <==350  <==330  <==300  <==245  <==235  <==15
5  <==130  <== 95  <== 80  <== 50


 470:  26 <-- 27  <-- 29  <-- 43  <-- 17  <-- 16  <-- 14  <--  9  <--  7  <--
5  <--  4  <--  3  <--  2  <--  1
     :鳴門 <--徳島 <--穴吹 <--貞光 <--阿波 <--川之 <--西条 <--小松 <--松山 <--伊
予 <--内子 <--大洲 <--八幡 <--三崎
 470:  25 <-- 50  <-- 15  <-- 30  <-- 50  <-- 55  <-- 10  <-- 80  <-- 25  <-- 3
5  <-- 15  <-- 30  <-- 50  <--  0
 470:  25 <== 50  <== 15  <== 30  <== 50  <== 55  <== 10  <== 80  <== 25  <== 3
```

```
5  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum : 470 <==445  <==395  <==380  <==350  <==300  <==245  <==235  <==155  <==13
0  <== 95  <== 80  <== 50


 445 :  27 <-- 29  <-- 43  <-- 17  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--
4  <--  3  <--  2  <--  1
     :徳島 <--穴吹 <--貞光 <--阿波 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内
子 <--大洲 <--八幡 <--三崎
 445 :  50 <-- 15  <-- 30  <-- 50  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 1
5  <-- 30  <-- 50  <--  0
 445 :  50 <== 15  <== 30  <== 50  <== 55  <== 10  <== 80  <== 25  <== 35  <== 1
5  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum : 445 <==395  <==380  <==350  <==300  <==245  <==235  <==155  <==130  <== 9
5  <== 80  <== 50


 460 :  28 <-- 27  <-- 29  <-- 43  <-- 17  <-- 16  <-- 14  <--  9  <--  7  <--
5  <--  4  <--  3  <--  2  <--  1
     :眉山 <--徳島 <--穴吹 <--貞光 <--阿波 <--川之 <--西条 <--小松 <--松山 <--伊
予 <--内子 <--大洲 <--八幡 <--三崎
 460 :  15 <-- 50  <-- 15  <-- 30  <-- 50  <-- 55  <-- 10  <-- 80  <-- 25  <-- 3
5  <-- 15  <-- 30  <-- 50  <--  0
 460 :  15 <== 50  <== 15  <== 30  <== 50  <== 55  <== 10  <== 80  <== 25  <== 3
5  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum : 460 <==445  <==395  <==380  <==350  <==300  <==245  <==235  <==155  <==13
0  <== 95  <== 80  <== 50


 395 :  29 <-- 43  <-- 17  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--
3  <--  2  <--  1
     :穴吹 <--貞光 <--阿波 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大
洲 <--八幡 <--三崎
 395 :  15 <-- 30  <-- 50  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 3
0  <-- 50  <--  0
 395 :  15 <== 30  <== 50  <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 3
0  <== 50  <==  0  dist[j][k] 区間時間
 sum : 395 <==380  <==350  <==300  <==245  <==235  <==155  <==130  <== 95  <== 8
0  <== 50


 475 :  30 <-- 27  <-- 29  <-- 43  <-- 17  <-- 16  <-- 14  <--  9  <--  7  <--
5  <--  4  <--  3  <--  2  <--  1
     :小松 <--徳島 <--穴吹 <--貞光 <--阿波 <--川之 <--西条 <--小松 <--松山 <--伊
予 <--内子 <--大洲 <--八幡 <--三崎
 475 :  30 <-- 50  <-- 15  <-- 30  <-- 50  <-- 55  <-- 10  <-- 80  <-- 25  <-- 3
5  <-- 15  <-- 30  <-- 50  <--  0
 475 :  30 <== 50  <== 15  <== 30  <== 50  <== 55  <== 10  <== 80  <== 25  <== 3
5  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum : 475 <==445  <==395  <==380  <==350  <==300  <==245  <==235  <==155  <==13
0  <== 95  <== 80  <== 50


 515 :  31 <-- 30  <-- 27  <-- 29  <-- 43  <-- 17  <-- 16  <-- 14  <--  9  <--
```

```
7  <--  5  <--  4  <--  3  <--  2  <--  1
     :見能 <--小松 <--徳島 <--穴吹 <--貞光 <--阿波 <--川之 <--西条 <--小松 <--松
山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 515 :  40 <-- 30  <-- 50  <-- 15  <-- 30  <-- 50  <-- 55  <-- 10  <-- 80  <-- 2
5  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 515 :  40 <== 30  <== 50  <== 15  <== 30  <== 50  <== 55  <== 10  <== 80  <== 2
5  <== 35  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum : 515 <==475  <==445  <==395  <==380  <==350  <==300  <==245  <==235  <==15
5  <==130  <== 95  <== 80  <== 50


 653 :  32 <-- 31  <-- 30  <-- 27  <-- 29  <-- 43  <-- 17  <-- 16  <-- 14  <--
9  <--  7  <--  5  <--  4  <--  3  <--  2  <--  1
     :清峰 <--見能 <--小松 <--徳島 <--穴吹 <--貞光 <--阿波 <--川之 <--西条 <--小
松 <--松山 <--伊予 <--内子 <--大洲 <--八幡 <--三崎
 653 : 138 <-- 40  <-- 30  <-- 50  <-- 15  <-- 30  <-- 50  <-- 55  <-- 10  <-- 8
0  <-- 25  <-- 35  <-- 15  <-- 30  <-- 50  <--  0
 653 : 138 <== 40  <== 30  <== 50  <== 15  <== 30  <== 50  <== 55  <== 10  <== 8
0  <== 25  <== 35  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum : 653 <==515  <==475  <==445  <==395  <==380  <==350  <==300  <==245  <==23
5  <==155  <==130  <== 95  <== 80  <== 50


 475 :  33 <-- 34  <-- 36  <-- 38  <-- 44  <--  6  <--  4  <--  3  <--  2  <--1
     :室戸 <--野市 <--御免 <--高知 <--佐川 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 475 : 120 <-- 10  <-- 30  <-- 40  <-- 90  <-- 90  <-- 15  <-- 30  <-- 50  <--0
 475 : 120 <== 10  <== 30  <== 40  <== 90  <== 90  <== 15  <== 30  <== 50  <==
0  dist[j][k] 区間時間
 sum : 475 <==355  <==345  <==315  <==275  <==185  <== 95  <== 80  <== 50


 355 :  34 <-- 36  <-- 38  <-- 44  <--  6  <--  4  <--  3  <--  2  <--  1
     :野市 <--御免 <--高知 <--佐川 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 355 :  10 <-- 30  <-- 40  <-- 90  <-- 90  <-- 15  <-- 30  <-- 50  <--  0
 355 :  10 <== 30  <== 40  <== 90  <== 90  <== 15  <== 30  <== 50  <==  0  dist[
j][k] 区間時間
 sum : 355 <==345  <==315  <==275  <==185  <== 95  <== 80  <== 50


 360 :  35 <-- 37  <-- 39  <-- 38  <-- 44  <--  6  <--  4  <--  3  <--  2  <-- 1
     :龍河 <--土佐 <--領石 <--高知 <--佐川 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 360 :  15 <-- 10  <-- 20  <-- 40  <-- 90  <-- 90  <-- 15  <-- 30  <-- 50  <-- 0
 360 :  15 <== 10  <== 20  <== 40  <== 90  <== 90  <== 15  <== 30  <== 50  <==
0  dist[j][k] 区間時間
 sum : 360 <==345  <==335  <==315  <==275  <==185  <== 95  <== 80  <== 50


 345 :  36 <-- 38  <-- 44  <--  6  <--  4  <--  3  <--  2  <--  1
     :御免 <--高知 <--佐川 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 345 :  30 <-- 40  <-- 90  <-- 90  <-- 15  <-- 30  <-- 50  <--  0
 345 :  30 <== 40  <== 90  <== 90  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区
間時間
 sum : 345 <==315  <==275  <==185  <== 95  <== 80  <== 50
```

```
 345：  37 <-- 39  <-- 38  <-- 44  <--  6  <--  4  <--  3  <--  2  <--  1
      :土佐 <--領石 <--高知 <--佐川 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 345：  10 <-- 20  <-- 40  <-- 90  <-- 90  <-- 15  <-- 30  <-- 50  <--  0
 345：  10 <== 20  <== 40  <== 90  <== 90  <== 15  <== 30  <== 50  <==  0  dist[
j][k] 区間時間
 sum：345 <==335  <==315  <==275  <==185  <== 95  <== 80  <== 50


 315：  38 <-- 44  <--  6  <--  4  <--  3  <--  2  <--  1
      :高知 <--佐川 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 315：  40 <-- 90  <-- 90  <-- 15  <-- 30  <-- 50  <--  0
 315：  40 <== 90  <== 90  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum：315 <==275  <==185  <== 95  <== 80  <== 50


 335：  39 <-- 38  <-- 44  <--  6  <--  4  <--  3  <--  2  <--  1
      :領石 <--高知 <--佐川 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 335：  20 <-- 40  <-- 90  <-- 90  <-- 15  <-- 30  <-- 50  <--  0
 335：  20 <== 40  <== 90  <== 90  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区
間時間
 sum：335 <==315  <==275  <==185  <== 95  <== 80  <== 50


 425：  40 <-- 17  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--  3  <--
2  <--  1
      :大豊 <--阿波 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大洲 <--八
幡 <--三崎
 425：  75 <-- 50  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 30  <-- 5
0  <--  0
 425：  75 <== 50  <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 30  <== 5
0  <==  0  dist[j][k] 区間時間
 sum：425 <==350  <==300  <==245  <==235  <==155  <==130  <== 95  <== 80  <== 5
0


 470：  41 <-- 40  <-- 17  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--
3  <--  2  <--  1
      :栃の <--大豊 <--阿波 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大
洲 <--八幡 <--三崎
 470：  45 <-- 75  <-- 50  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 3
0  <-- 50  <--  0
 470：  45 <== 75  <== 50  <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 3
0  <== 50  <==  0  dist[j][k] 区間時間
 sum：470 <==425  <==350  <==300  <==245  <==235  <==155  <==130  <== 95  <== 8
0  <== 50


 470：  42 <-- 43  <-- 17  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--
3  <--  2  <--  1
      :見ノ <--貞光 <--阿波 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大
洲 <--八幡 <--三崎
 470：  90 <-- 30  <-- 50  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 3
0  <-- 50  <--  0
```

```
 470：  90 <== 30  <== 50  <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 3
0  <== 50  <==  0  dist[j][k] 区間時間
 sum：470 <==380  <==350  <==300  <==245  <==235  <==155  <==130  <== 95  <== 8
0  <== 50


 380：  43 <-- 17  <-- 16  <-- 14  <--  9  <--  7  <--  5  <--  4  <--  3  <--
2  <--  1
     :貞光 <--阿波 <--川之 <--西条 <--小松 <--松山 <--伊予 <--内子 <--大洲 <--八
幡 <--三崎
 380：  30 <-- 50  <-- 55  <-- 10  <-- 80  <-- 25  <-- 35  <-- 15  <-- 30  <-- 5
0  <--  0
 380：  30 <== 50  <== 55  <== 10  <== 80  <== 25  <== 35  <== 15  <== 30  <== 5
0  <==  0  dist[j][k] 区間時間
 sum：380 <==350  <==300  <==245  <==235  <==155  <==130  <== 95  <== 80  <== 5
0


 275：  44 <--  6  <--  4  <--  3  <--  2  <--  1
     :佐川 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 275：  90 <-- 90  <-- 15  <-- 30  <-- 50  <--  0
 275：  90 <== 90  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum：275 <==185  <== 95  <== 80  <== 50


 315：  45 <-- 44  <--  6  <--  4  <--  3  <--  2  <--  1
     :須崎 <--佐川 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 315：  40 <-- 90  <-- 90  <-- 15  <-- 30  <-- 50  <--  0
 315：  40 <== 90  <== 90  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum：315 <==275  <==185  <== 95  <== 80  <== 50


 235：  46 <--  6  <--  4  <--  3  <--  2  <--  1
     :面河 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 235：  50 <-- 90  <-- 15  <-- 30  <-- 50  <--  0
 235：  50 <== 90  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum：235 <==185  <== 95  <== 80  <== 50


 265：  47 <-- 46  <--  6  <--  4  <--  3  <--  2  <--  1
     :石鎚 <--面河 <--御三 <--内子 <--大洲 <--八幡 <--三崎
 265：  30 <-- 50  <-- 90  <-- 15  <-- 30  <-- 50  <--  0
 265：  30 <== 50  <== 90  <== 15  <== 30  <== 50  <==  0  dist[j][k] 区間時間
 sum：265 <==235  <==185  <== 95  <== 80  <== 50


 290：  48 <-- 50  <-- 54  <-- 55  <-- 49  <-- 58  <--  2  <--  1
     :窪川 <--中村 <--宿毛 <--御荘 <--宇和 <--大江 <--八幡 <--三崎
 290：  40 <-- 30  <-- 30  <-- 70  <-- 30  <-- 40  <-- 50  <--  0
 290：  40 <== 30  <== 30  <== 70  <== 30  <== 40  <== 50  <==  0  dist[j][k] 区
間時間
 sum：290 <==250  <==220  <==190  <==120  <== 90  <== 50
```

```
120：  49 <-- 58  <--  2  <--  1
     :宇和 <--大江 <--八幡 <--三崎
120：  30 <-- 40  <-- 50  <--  0
120：  30 <== 40  <== 50  <==  0  dist[j][k] 区間時間
sum：120 <== 90  <== 50


250：  50 <-- 54  <-- 55  <-- 49  <-- 58  <--  2  <--  1
     :中村 <--宿毛 <--御荘 <--宇和 <--大江 <--八幡 <--三崎
250：  30 <-- 30  <-- 70  <-- 30  <-- 40  <-- 50  <--  0
250：  30 <== 30  <== 70  <== 30  <== 40  <== 50  <==  0  dist[j][k] 区間時間
sum：250 <==220  <==190  <==120  <== 90  <== 50


310：  51 <-- 50  <-- 54  <-- 55  <-- 49  <-- 58  <--  2  <--  1
     :土佐 <--中村 <--宿毛 <--御荘 <--宇和 <--大江 <--八幡 <--三崎
310：  60 <-- 30  <-- 30  <-- 70  <-- 30  <-- 40  <-- 50  <--  0
310：  60 <== 30  <== 30  <== 70  <== 30  <== 40  <== 50  <==  0  dist[j][k] 区
間時間
sum：310 <==250  <==220  <==190  <==120  <== 90  <== 50


340：  52 <-- 51  <-- 50  <-- 54  <-- 55  <-- 49  <-- 58  <--  2  <--  1
     :足摺 <--土佐 <--中村 <--宿毛 <--御荘 <--宇和 <--大江 <--八幡 <--三崎
340：  30 <-- 60  <-- 30  <-- 30  <-- 70  <-- 30  <-- 40  <-- 50  <--  0
340：  30 <== 60  <== 30  <== 30  <== 70  <== 30  <== 40  <== 50  <==  0  dist[
j][k] 区間時間
sum：340 <==310  <==250  <==220  <==190  <==120  <== 90  <== 50


300：  53 <-- 54  <-- 55  <-- 49  <-- 58  <--  2  <--  1
     :竜串 <--宿毛 <--御荘 <--宇和 <--大江 <--八幡 <--三崎
300：  80 <-- 30  <-- 70  <-- 30  <-- 40  <-- 50  <--  0
300：  80 <== 30  <== 70  <== 30  <== 40  <== 50  <==  0  dist[j][k] 区間時間
sum：300 <==220  <==190  <==120  <== 90  <== 50


220：  54 <-- 55  <-- 49  <-- 58  <--  2  <--  1
     :宿毛 <--御荘 <--宇和 <--大江 <--八幡 <--三崎
220：  30 <-- 70  <-- 30  <-- 40  <-- 50  <--  0
220：  30 <== 70  <== 30  <== 40  <== 50  <==  0  dist[j][k] 区間時間
sum：220 <==190  <==120  <== 90  <== 50


190：  55 <-- 49  <-- 58  <--  2  <--  1
     :御荘 <--宇和 <--大江 <--八幡 <--三崎
190：  70 <-- 30  <-- 40  <-- 50  <--  0
190：  70 <== 30  <== 40  <== 50  <==  0  dist[j][k] 区間時間
sum：190 <==120  <== 90  <== 50


200：  56 <-- 55  <-- 49  <-- 58  <--  2  <--  1
     :船越 <--御荘 <--宇和 <--大江 <--八幡 <--三崎
200：  10 <-- 70  <-- 30  <-- 40  <-- 50  <--  0
```

```
200：  10 <== 70   <== 30   <== 40   <== 50   <==   0   dist[j][k] 区間時間
sum：200 <==190   <==120   <== 90   <== 50


200：  57 <-- 49   <-- 58   <--   2   <--   1
     :滑床 <--宇和 <--大江 <--八幡 <--三崎
200：  80 <-- 30   <-- 40   <-- 50   <--   0
200：  80 <== 30   <== 40   <== 50   <==   0   dist[j][k] 区間時間
sum：200 <==120   <== 90   <== 50


 90：  58 <--   2   <--   1
     :大江 <--八幡 <--三崎
 90：  40 <-- 50   <--   0
 90：  40 <== 50   <==   0   dist[j][k] 区間時間
sum：  90 <== 50



 分：到達点 <-- 経由点

 1=j     0   三崎
 2=j     1   八幡浜 三崎
 3=j     2   大洲 八幡浜 三崎
 4=j     3   内子 大洲 八幡浜 三崎
 5=j     4   伊予市 内子 大洲 八幡浜 三崎
 6=j     4   御三戸 内子 大洲 八幡浜 三崎
 7=j     5   松山 伊予市 内子 大洲 八幡浜 三崎
 8=j     6   北条 松山 伊予市 内子 大洲 八幡浜 三崎
 9=j     6   小松 松山 伊予市 内子 大洲 八幡浜 三崎
10=j     7   大西 北条 松山 伊予市 内子 大洲 八幡浜 三崎
11=j     7   鈍川温泉 北条 松山 伊予市 内子 大洲 八幡浜 三崎
12=j     8   今治 大西 北条 松山 伊予市 内子 大洲 八幡浜 三崎
13=j     8   糸山 大西 北条 松山 伊予市 内子 大洲 八幡浜 三崎
14=j     7   西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
15=j     8   新浜 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
16=j     8   川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
17=j     9   阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
18=j     9   観音寺 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
19=j    10   琴平 観音寺 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
20=j    10   金蔵寺 観音寺 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
21=j    11   坂出 金蔵寺 観音寺 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三
崎
22=j    12   五色台 坂出 金蔵寺 観音寺 川之江 西条 小松 松山 伊予市 内子 大洲 八
幡浜 三崎
23=j    12   高松 坂出 金蔵寺 観音寺 川之江 西条 小松 松山 伊予市 内子 大洲 八幡
浜 三崎
24=j    13   湯元 高松 坂出 金蔵寺 観音寺 川之江 西条 小松 松山 伊予市 内子 大洲
八幡浜 三崎
25=j    14   屋島 湯元 高松 坂出 金蔵寺 観音寺 川之江 西条 小松 松山 伊予市 内子
大洲 八幡浜 三崎
26=j    13   鳴門 徳島 穴吹 貞光 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲
八幡浜 三崎
27=j    12   徳島 穴吹 貞光 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡
```

浜 三崎
28 =j 13 眉山 徳島 穴吹 貞光 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
29 =j 11 穴吹 貞光 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
30 =j 13 小松島 徳島 穴吹 貞光 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
31 =j 14 見能林 小松島 徳島 穴吹 貞光 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
32 =j 15 清峰 見能林 小松島 徳島 穴吹 貞光 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
33 =j 9 室戸岬 野市 御免 高知 佐川 御三戸 内子 大洲 八幡浜 三崎
34 =j 8 野市 御免 高知 佐川 御三戸 内子 大洲 八幡浜 三崎
35 =j 9 龍河洞 土佐山田 領石 高知 佐川 御三戸 内子 大洲 八幡浜 三崎
36 =j 7 御免 高知 佐川 御三戸 内子 大洲 八幡浜 三崎
37 =j 8 土佐山田 領石 高知 佐川 御三戸 内子 大洲 八幡浜 三崎
38 =j 6 高知 佐川 御三戸 内子 大洲 八幡浜 三崎
39 =j 7 領石 高知 佐川 御三戸 内子 大洲 八幡浜 三崎
40 =j 10 大豊 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
41 =j 11 栃の瀬 大豊 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
42 =j 11 見ノ越 貞光 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
43 =j 10 貞光 阿波池田 川之江 西条 小松 松山 伊予市 内子 大洲 八幡浜 三崎
44 =j 5 佐川 御三戸 内子 大洲 八幡浜 三崎
45 =j 6 須崎 佐川 御三戸 内子 大洲 八幡浜 三崎
46 =j 5 面河渓 御三戸 内子 大洲 八幡浜 三崎
47 =j 6 石鎚山 面河渓 御三戸 内子 大洲 八幡浜 三崎
48 =j 7 窪川 中村 宿毛 御荘 宇和島 大江 八幡浜 三崎
49 =j 3 宇和島 大江 八幡浜 三崎
50 =j 6 中村 宿毛 御荘 宇和島 大江 八幡浜 三崎
51 =j 7 土佐清水 中村 宿毛 御荘 宇和島 大江 八幡浜 三崎
52 =j 8 足摺岬 土佐清水 中村 宿毛 御荘 宇和島 大江 八幡浜 三崎
53 =j 6 竜串 宿毛 御荘 宇和島 大江 八幡浜 三崎
54 =j 5 宿毛 御荘 宇和島 大江 八幡浜 三崎
55 =j 4 御荘 宇和島 大江 八幡浜 三崎
56 =j 5 船越 御荘 宇和島 大江 八幡浜 三崎
57 =j 4 清床 宇和島 大江 八幡浜 三崎
58 =j 2 大江 八幡浜 三崎

dist[j][k]

```
 0 =j :   0
 1 =j :   0
 2 =j :  50   0
 3 =j :  30  50   0
 4 =j :  15  30  50   0
 5 =j :  35  15  30  50   0
 6 =j :  90  15  30  50   0
 7 =j :  25  35  15  30  50   0
 8 =j :  30  25  35  15  30  50   0
 9 =j :  80  25  35  15  30  50   0
10 =j :  30  30  25  35  15  30  50   0
11 =j :  50  30  25  35  15  30  50   0
12 =j :  20  30  30  25  35  15  30  50   0
```

```
13 := j:  30  30  30  25  35  15  30  50   0
14 := j:  10  80  25  35  15  30  50   0
15 := j:  15  10  80  25  35  15  30  50   0
16 := j:  55  10  80  25  35  15  30  50   0
17 := j:  50  55  10  80  25  35  15  30  50   0
18 := j:  30  55  10  80  25  35  15  30  50   0
19 := j:  30  30  55  10  80  25  35  15  30  50   0
20 := j:  20  30  55  10  80  25  35  15  30  50   0
21 := j:  20  20  30  55  10  80  25  35  15  30  50   0
22 := j:  40  20  20  30  55  10  80  25  35  15  30  50   0
23 := j:  40  20  20  30  55  10  80  25  35  15  30  50   0
24 := j:  15  40  20  20  30  55  10  80  25  35  15  30  50   0

25 := j:  10  15  40  20  20  30  55  10  80  25  35  15  30  50
   0
26 := j:  25  50  15  30  50  55  10  80  25  35  15  30  50   0

27 := j:  50  15  30  50  55  10  80  25  35  15  30  50   0
28 := j:  15  50  15  30  50  55  10  80  25  35  15  30  50   0

29 := j:  15  30  50  55  10  80  25  35  15  30  50   0
30 := j:  30  50  15  30  50  55  10  80  25  35  15  30  50   0

31 := j:  40  30  50  15  30  50  55  10  80  25  35  15  30  50
   0
32 := j: 138  40  30  50  15  30  50  55  10  80  25  35  15  30
  50    0
33 := j: 120  10  30  40  90  90  15  30  50   0
34 := j:  10  30  40  90  90  15  30  50   0
35 := j:  15  10  20  40  90  90  15  30  50   0
36 := j:  30  40  90  90  15  30  50   0
37 := j:  10  20  40  90  90  15  30  50   0
38 := j:  40  90  90  15  30  50   0
39 := j:  20  40  90  90  15  30  50   0
40 := j:  75  50  55  10  80  25  35  15  30  50   0
41 := j:  45  75  50  55  10  80  25  35  15  30  50   0
42 := j:  90  30  50  55  10  80  25  35  15  30  50   0
43 := j:  30  50  55  10  80  25  35  15  30  50   0
44 := j:  90  90  15  30  50   0
45 := j:  40  90  90  15  30  50   0
46 := j:  50  90  15  30  50   0
47 := j:  30  50  90  15  30  50   0
48 := j:  40  30  30  70  30  40  50   0
49 := j:  30  40  50   0
50 := j:  30  30  70  30  40  50   0
51 := j:  60  30  30  70  30  40  50   0
52 := j:  30  60  30  30  70  30  40  50   0
53 := j:  80  30  70  30  40  50   0
54 := j:  30  70  30  40  50   0
55 := j:  70  30  40  50   0
56 := j:  10  70  30  40  50   0
57 := j:  80  30  40  50   0
58 := j:  40  50   0
```

```
sumd[j][k]
 0 := j :   0
 1 := j :   0
 2 := j :  50   0
 3 := j :  80  50   0
 4 := j :  95  80  50   0
 5 := j : 130  95  80  50   0
 6 := j : 185  95  80  50   0
 7 := j : 155 130  95  80  50   0
 8 := j : 185 155 130  95  80  50   0
 9 := j : 235 155 130  95  80  50   0
10 := j : 215 185 155 130  95  80  50   0
11 := j : 235 185 155 130  95  80  50   0
12 := j : 235 215 185 155 130  95  80  50   0
13 := j : 245 215 185 155 130  95  80  50   0
14 := j : 245 235 155 130  95  80  50   0
15 := j : 260 245 235 155 130  95  80  50   0
16 := j : 300 245 235 155 130  95  80  50   0
17 := j : 350 300 245 235 155 130  95  80  50   0
18 := j : 330 300 245 235 155 130  95  80  50   0
19 := j : 360 330 300 245 235 155 130  95  80  50   0
20 := j : 350 330 300 245 235 155 130  95  80  50   0
21 := j : 370 350 330 300 245 235 155 130  95  80  50   0
22 := j : 410 370 350 330 300 245 235 155 130  95  80  50   0
23 := j : 410 370 350 330 300 245 235 155 130  95  80  50   0
24 := j : 425 410 370 350 330 300 245 235 155 130  95  80  50   0

25 := j : 435 425 410 370 350 330 300 245 235 155 130  95  80  50
   0
26 := j : 470 445 395 380 350 300 245 235 155 130  95  80  50   0

27 := j : 445 395 380 350 300 245 235 155 130  95  80  50   0
28 := j : 460 445 395 380 350 300 245 235 155 130  95  80  50   0

29 := j : 395 380 350 300 245 235 155 130  95  80  50   0
30 := j : 475 445 395 380 350 300 245 235 155 130  95  80  50   0

31 := j : 515 475 445 395 380 350 300 245 235 155 130  95  80  50
   0
32 := j : 653 515 475 445 395 380 350 300 245 235 155 130  95  80
  50   0
33 := j : 475 355 345 315 275 185  95  80  50   0
34 := j : 355 345 315 275 185  95  80  50   0
35 := j : 360 345 335 315 275 185  95  80  50   0
36 := j : 345 315 275 185  95  80  50   0
37 := j : 345 335 315 275 185  95  80  50   0
38 := j : 315 275 185  95  80  50   0
39 := j : 335 315 275 185  95  80  50   0
40 := j : 425 350 300 245 235 155 130  95  80  50   0
41 := j : 470 425 350 300 245 235 155 130  95  80  50   0
42 := j : 470 380 350 300 245 235 155 130  95  80  50   0
43 := j : 380 350 300 245 235 155 130  95  80  50   0
44 := j : 275 185  95  80  50   0
45 := j : 315 275 185  95  80  50   0
```

```
 46 :=j:  235  185   95   80   50    0
 47 :=j:  265  235  185   95   80   50    0
 48 :=j:  290  250  220  190  120   90   50    0
 49 :=j:  120   90   50    0
 50 :=j:  250  220  190  120   90   50    0
 51 :=j:  310  250  220  190  120   90   50    0
 52 :=j:  340  310  250  220  190  120   90   50    0
 53 :=j:  300  220  190  120   90   50    0
 54 :=j:  220  190  120   90   50    0
 55 :=j:  190  120   90   50    0
 56 :=j:  200  190  120   90   50    0
 57 :=j:  200  120   90   50    0
 58 :=j:   90   50    0
Dijkstra Algorithm   program Normal End

別の始点からの経路の検索をしますか (y/n)=n

ダイクストラ･アルゴリズム  完了
Program Normal End ...  正常終了
Press any key to continue
```

［完］