## 演習問題2　最短経路のダイクストラ法

**問　次のプログラムの説明およびプログラムを読んで，設問に答えよ。**

〔プログラムの説明〕

出発地から目的地までの最短経路を求める副プログラムである。

(1) 副プログラムSPは，N個（N>1）の地点で構成される**図1**のような有向グラフで表される経路図において，地点1（出発地）から地点N（目的地）までの最短経路を求めて，出力するプログラムである。
なお，**図1**において，円は地点を，矢印の向きは進行方向を，矢印に付けた数字は地点間の距離を表す。

(2) この副プログラムでは，次の配列を用いる。要素番号i，jの値は1，2，…，Nであり，各地点と対応している。

C［i，j］：　地点iから地点jまでの距離を表す配列である。地点iから地点jまでの直接の経路がない場合，地点iと地点jが等しい場合，および進行方向と逆の場合は∞が格納されている。例えば**図1**では，C［1，2］=30，C［1，3］=∞，C［2，1］=∞，C［2，2］=∞である。

D［i］：　地点1から地点iまでの最短距離を格納するための配列である。初期設定で地点1から地点iまでの距離C［1，i］を設定する。例えば**図1**の場合の初期設定では，D［2］=C［1，2］=30，D［3］=C［1，3］=∞，D［4］=C［1，4］=20，D［5］=C［1，5］=120が設定される。

P［i］：　最短経路を求める過程で，処理済となった地点を識別するための配列である。初期設定ではP［1］だけに1を設定し，他のすべての要素には0を設定する。最短経路を求めた後はすべての要素が1となる。

S［i］：　地点1から地点iまでの最短経路を求める過程で，地点iの直前の地点を格納するための配列である。初期設定ではすべての要素に1を設定する。例えば**図1**の場合，最短経路を求めた後はS［2］=1，S［3］=4，S［4］=1，S［5］=3となり，地点5の直前は地点3，地点3の直前は地点4，地点4の直前は地点1とわかる。すなわち，地点1から地点5までの最短経路は，地点1→地点4→地点3→地点5となる。

W［i］：　S［i］を用いて最短経路を出力する際に使用する作業用の配列である。

(3) 最短経路を求める手順は，次のとおりである。

① 処理していないすべての地点（P［i］=0）のうちで，D［T］が最小である地点Tを選ぶ。

② 地点Tを処理済（P［T］=1）とする。

③ 処理していないすべての地点（P［i］=0）に対して，D［T］+C［T，i］の値がD［i］の値より小さければ，D［T］+C［T，i］の値でD［i］を置き換える。

④ ③でD［i］を置き換えた場合，S［i］にTを代入する。

⑤ 処理①～④を，全地点が処理済になるまで繰返す。

(4) 最短経路を求めた後，配列Sを用いて最短経路を出力する。出力には，数値Xを出力した後に改行する副プログラムOutput (X) を利用する。

(5) **図1**の経路図に対する出力結果を**図2**に，また最短経路を求める過程での配列と変数Tの値の変化を**表1**に示す。

(6) 出発地から目的地までの経路は，必ず存在するものとする。

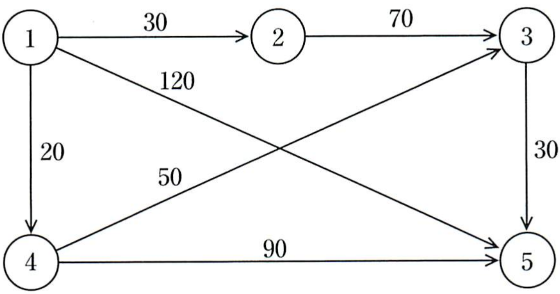



**図1　経路図の例（N=5の場合）**

```
1
4
3
5
```

**図2　図1の経路図に対する出力結果**

### 表1　図1の最短経路を求める過程の配列と変数Tの値の変化

| | | 配列 | [1] | [2] | [3] | [4] | [5] | T |
|---|---|---|---|---|---|---|---|---|
| 1回目 | ① | P | 1 | 0 | 0 | 0 | 0 | |
| | | D | ∞ | 30 | ∞ | $\boxed{20}$ | 120 | 4 |
| | ② | P | 1 | 0 | 0 | 1 | 0 | 4 |
| | ③ | D | ∞ | 30 | 70 | 20 | 110 | 4 |
| | ④ | S | 1 | 1 | 4 | 1 | 4 | 4 |
| 2回目 | ① | P | 1 | 0 | 0 | 1 | 0 | |
| | | D | ∞ | $\boxed{30}$ | 70 | 20 | 110 | 2 |
| | ② | P | 1 | 1 | 0 | 1 | 0 | 2 |
| | ③ | D | ∞ | 30 | 70 | 20 | 110 | 2 |
| | ④ | S | 1 | 1 | 4 | 1 | 4 | 2 |
| 3回目 | ① | P | 1 | 1 | 0 | 1 | 0 | |
| | | D | ∞ | 30 | $\boxed{70}$ | 20 | 110 | 3 |
| | ② | P | 1 | 1 | 1 | 1 | 0 | 3 |
| | ③ | D | ∞ | 30 | 70 | 20 | 100 | 3 |
| | ④ | S | 1 | 1 | 4 | 1 | 3 | 3 |
| 4回目 | ① | P | 1 | 1 | 1 | 1 | 0 | |
| | | D | ∞ | 30 | 70 | 20 | $\boxed{100}$ | 5 |
| | ② | P | 1 | 1 | 1 | 1 | 1 | 5 |
| | ③ | D | ∞ | 30 | 70 | 20 | 100 | 5 |
| | ④ | S | 1 | 1 | 4 | 1 | 3 | 5 |

**注**　表中の①～④は，プログラムの説明中の (3) ①～④に対応する。

$\boxed{\phantom{0}}$ はTを選んだときの配列を，　■ は値が置き換えられた要素を示す。

**(7)** 副プログラムSPの引数は，**表2**のとおりである。

### 表2　SPの引数

| 変数名 | 入力／出力 | 意　味 |
|---|---|---|
| N | 入力 | 地点数 |
| C [ , ] | 入力 | 地点間の距離を表す2次元の定数配列 |

**〔プログラム〕**

```
○プログラム名：SP (N, C [ , ])
○実数型：C [ , ], D [N], Z
○整数型：P [N], S [N], W [N], N, T, X, Y

{初期設定}
・X ← 1
■ X ≦ N
│   ・D [X] ← C [1, X]
│   ・P [X] ← 0
│   ・S [X] ← 1
│   ・X ← X + 1
■
・P [1] ← 1
```

{最短経路を求める処理}

```
・X ← 2
■ X ≦ N
|    ・Y ← 2
|    ・Z ← ∞
|    ■ Y ≦ N
|    |    ▲ (P [Y] = 0) and (D [Y] < Z)
|    |    |    ・T ← Y
|    |    |    ・[ a ]
|    |    ▼
|    |    ・Y ← Y + 1
|    ■
|    ・P [T] ← 1
|    ・Y ← 2
|    ■ Y ≦ N
|    |    ▲ (P [Y] = 0) and (D [Y] > (D [T] + C [T, Y]))
|    |    |    ・D [Y] ← D [T] + C [T, Y]
|    |    |    ・[ b ]
|    |    ▼
|    |    ・Y ← Y + 1
|    ■
|    ・X ← X + 1
■
```

{最短経路の出力処理}

```
・X ← 1
・Y ← N
■ Y ≠ 1
|    ・W [X] ← Y
|    ・[ c ]
|    ・X ← X + 1
■
・W [X] ← Y
■ [ d ]
|    ・Output (W [X])
|    ・X ← X − 1
■
```

## 設問

プログラム中の［　　　　］に入れる正しい答えを，解答群の中から選べ。

**a～cに関する解答群**

**ア** P［Y］← 0　　**イ** P［Y］← 1　　**ウ** S［T］← Y

**エ** S［Y］← T　　**オ** Y ← S［X］　　**カ** Y ← S［Y］

**キ** Z ← D［Y］

**dに関する解答群**

**ア** X ＞ 0　　**イ** X ≧ 0

**ウ** W［X］＞ 0　　**エ** W［X］≧ 0

### 共通に使用される擬似言語の記述形式

擬似言語を使用したこの問題では，問題文中に注記がない限り，この記述形式が適用されているものとする。

［擬似言語の記述形式の説明］

| 記述形式 | 説　明 |
|---|---|
| ○ | 手続，変数などの名前，型などを宣言する。 |
| ・変数 ← 式 | 変数に式の値を代入する。 |
| {文} | 文に注釈を記述する。 |
| ▲ 条件式<br>・処理1<br>―――<br>・処理2<br>▼ | 選択処理を示す。<br>条件式が真のときは処理1を実行し，<br>偽のときは処理2を実行する。 |
| ■ 条件式<br>・処理<br>■ | 前判定繰返し処理を示す。<br>条件式が真の間，処理を実行する。 |

## 解説

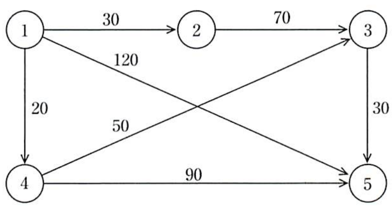
### 設問の考え方

最短経路のダイクストラ法です。

ダイクストラ法のアルゴリズムを知らないと，問題文を読んで理解するだけでも大変ですね。

問題文の**図1**の例で説明します。５つの地点があるのでN＝5です。

まず，重要なのが配列C［i，j］です。5×5の２次元配列で，次のような値が設定されています。




| C［i，j］ | j=1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| i=1 | ∞ | 30 | ∞ | 20 | 120 |
| 2 | ∞ | ∞ | 70 | ∞ | ∞ |
| 3 | ∞ | ∞ | ∞ | ∞ | 30 |
| 4 | ∞ | ∞ | 50 | ∞ | 90 |
| 5 | ∞ | ∞ | ∞ | ∞ | ∞ |

C［1，2］なら，地点①から地点②までの距離30が設定されています。

配列D［i］，P［i］，S［i］の初期値は次のとおりです。

| D[i] | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | ∞ | 30 | ∞ | 20 | 120 |

| P[i] | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | 1 | 0 | 0 | 0 | 0 |

| S[i] | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 |

では，擬似言語のプログラムを見てみましょう。初期設定では，Xを1～N（**図1**の例では，1～5）まで変化させて，上図のような初期値を設定しています。

最短経路を求めるのが重要です。Xを2～Nまで変化させる大きなループの中に，２つのループがあります。内側の１つ目のループは何をしているのでしょうか？

```
・Y ← 2
・Z ← ∞
■ Y ≦ N の間繰返す
│ ▲ (P［Y］ = 0) and (D［Y］ < Z) なら,
│ │   ・T ← Y
│ │   ・[    a    ]
│ ▼
│  ・Y ← Y + 1
■
```

Yに2を設定して，Yの値を1ずつ増やしながらY≦Nの間繰返すのですから，Yを2～Nまで繰返します。「(P［Y］ = 0) and (D［Y］ < Z)」の条件が真のときに，TにYを代入し，**空欄a**を実行します。

プログラムの説明（3）の最短経路を求める手順の①「処理していないすべての地点（P［i］=0）のうちで，D［T］が最小である地点Tを選ぶ」とあります。D［Y］の最小値をTに求めるようです。ZにYまでの最小値を設定しておくのでしょう。**キ**の「Z←D［Y］」を入れます。最小値を求める基本的なアルゴリズムです。

プログラムの説明（3）の②に「地点Tを処理済（P［T］＝1）とする」とあり，確かに1にしています。

```
    ・P［T］ ← 1
```

1回目はD［2］の30が最小値なので，T＝4，P［4］←1とします。

| D［i］ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | ∞ | 30 | ∞ | 20 | 120 |

| P［i］ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | 0 |

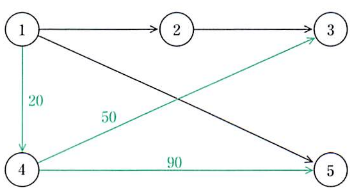
| S［i］ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 |

2つ目のループもYを2〜Nまで繰返します。

```
・Y ← 2
■Y ≦ N
│  ▲(P［Y］ ＝ 0) and (D［Y］ ＞ (D［T］ ＋ C［T, Y］))
│  │  ・D［Y］ ← D［T］ ＋ C［T, Y］
│  │  ・[        b        ]
│  ▼
│  ・Y ← Y ＋ 1
■
```

手順の③「処理していないすべての地点（P［i］＝0）に対して，D［T］＋C［T, i］の値がD［i］の値より小さければ，D［T］＋C［T, i］の値でD［i］を置き換える」とあり，さらに④「③でD［i］を置き換えた場合，S［i］にTを代入する」の部分が**空欄b**でしょう。したがって，**空欄b**は，**エ**の「S［Y］←T」になります。

どのような処理を行っているかを見ておきましょう。T＝4ですから，D［Y］と（D［4］＋C［4, Y］）で，（D［Y］と20＋C［4, Y］）を比較します。




| C［i，j］ | j＝1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| i＝1 | ∞ | 30 | ∞ | 20 | 120 |
| 2 | ∞ | ∞ | 70 | ∞ | ∞ |
| 3 | ∞ | ∞ | ∞ | ∞ | 30 |
| 4 | ∞ | ∞ | 50 | ∞ | 90 |
| 5 | ∞ | ∞ | ∞ | ∞ | ∞ |

Y＝2〜5でトレースしてみました。

| Y | D［Y］ | 比較 | 20＋C［4，Y］ | |
|---|---|---|---|---|
| 2 | 30 | | 20＋∞ | |
| 3 | ∞ | ＞ | 20＋50＝70 | D［3］を70に置き換え |
| 4 | 20 | | 20＋∞ | |
| 5 | 120 | ＞ | 20＋90＝110 | D［5］を110に置き換え |

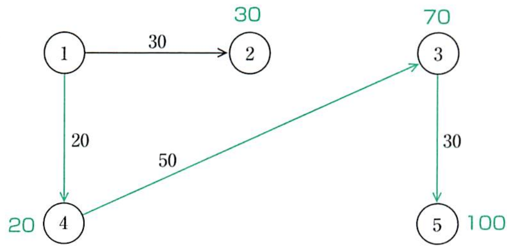
例えば，地点⑤には，①→⑤の他に，①→④→⑤の経路でも行くことができ，①→④→⑤の方が距離が短いわけです。S［3］とS［5］は，①から来るより④を経由したほうが近いという意味で，4を設定しているのです。

| D [i] | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | ∞ | 30 | 70 | 20 | 110 |

| P [i] | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | 1 | 0 | 0 | 1 | 0 |

| S [i] | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | 1 | 1 | 4 | 1 | 4 |

これが問題文の**表1**で，4回目までトレースされています。ぜひ，**表1**とつき合わせて自分でトレースしておいてください。




S［ i ］には経由地点が記録されているので，地点⑤から逆にたどると最短経路になっています。

| D [i] | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | ∞ | 30 | 70 | 20 | 100 |

| P [i] | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | 1 | 1 | 1 | 1 | 1 |

| S [i] | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| | 1 | 1 | 4 | 1 | 3 |

つまり，⑤←③←④←①が最短経路で，最短距離D［5］は100です。

さて，最短経路の出力処理を見ましょう。

```
・X ← 1
・Y ← N
■Y ≠ 1
│  ・W [X] ← Y
│  ・[ c ]
│  ・X ← X + 1
■
・W [X] ← Y
■[ d ]
│  ・Output (W [X])
│  ・X ← X − 1
■
```

YにNを設定しているということは，終点からスタートして，始点（Y＝1）に到達するまで繰返すのでしょう。S［ i ］を逆にたどりたいわけです。つまり，YをW［ X ］に保存した上で，S［ Y ］を新しいYにすればよいので，**空欄c**には，**カ**の「Y ← S［ Y ］」が入ります。

W［ X ］に最短経路が逆順で記憶されました。N＝5なら，W［ 5 ］～W［ 1 ］までを出力すればいいわけです。Xから1を引いていますから，Xを1まで変化させてループ内を実行するためのループ条件は，**ア**の「X　＞　0」になります。

**解答**

**設問　a キ　b エ　c カ　d ア**