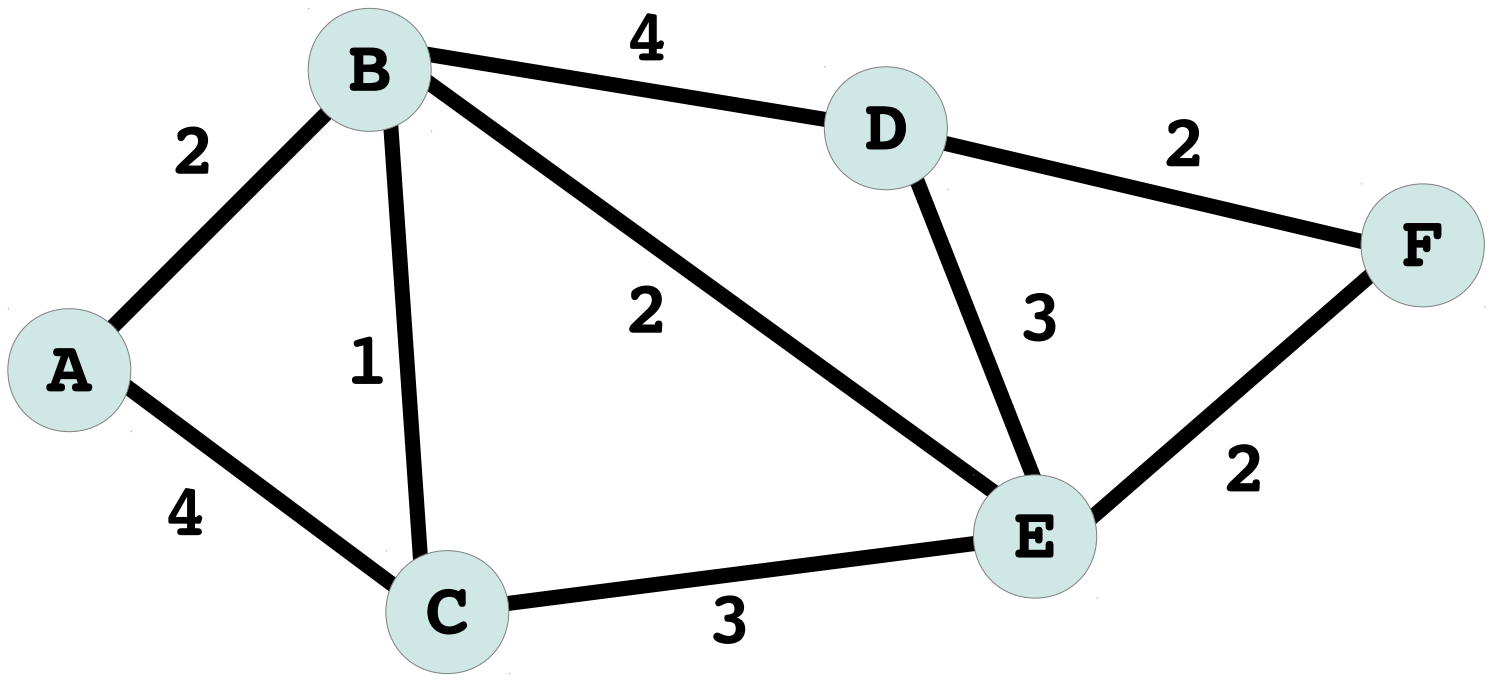
Algoritmo di Dijkstra

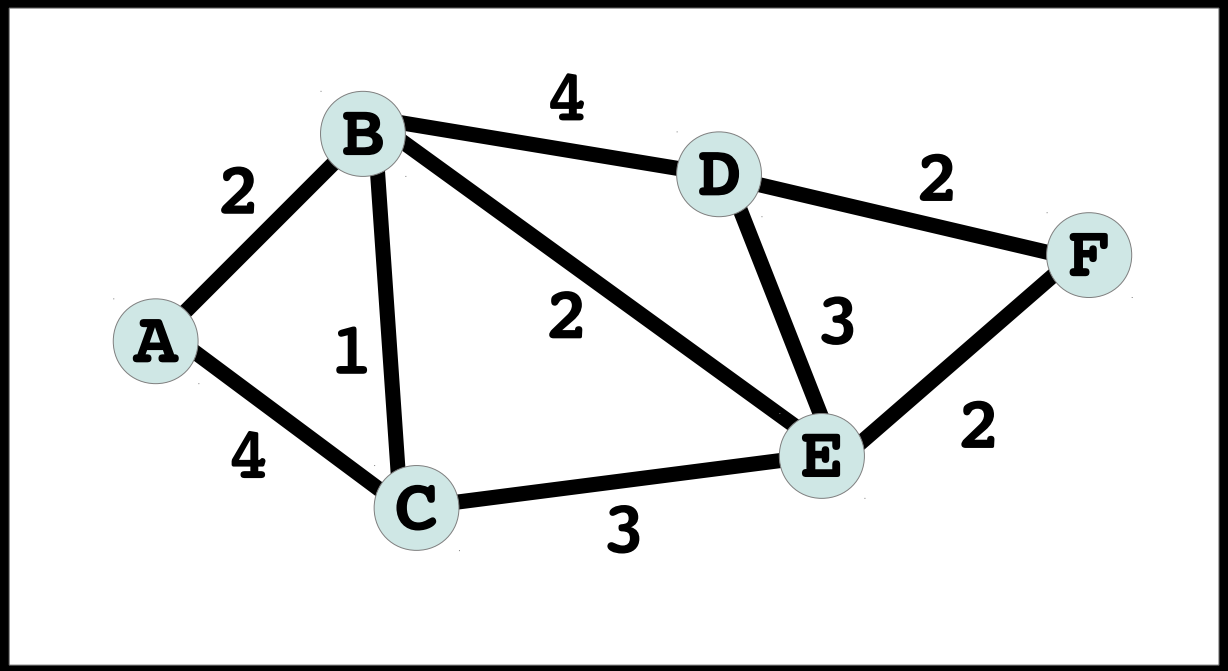
Problem Solving
1959: *Dijkstra's Algorithm*



Trovare il percorso a *energia minima* da A a F

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

Problem Solving
1959: *Dijkstra's Algorithm*



Quali variabili?
Quali contenitori?
Quale struttura dati?
MATRICE NxN!

	A	B	C	D	E	F
A	0	2	4	∞	∞	∞
B	2	0	1	4	2	∞
C	4	1	0	∞	3	∞
D	∞	4	∞	0	3	2
E	∞	2	3	3	0	2
F	∞	∞	∞	2	2	0

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

Problem Solving
1959: *Dijkstra's Algorithm*

1. Ogni **nodo memorizza** l'energia minima necessaria per raggiungerlo.
2. Da **ogni nodo** si calcola l'energia minima per arrivare a tutti **i suoi nodi adiacenti.**
L'energia per arrivare da un nodo I a un nodo J e' data dalla **somma** di
- energia per arrivare al nodo I di partenza (memorizzata nel nodo I)
- energia per arrivare al nodo J di arrivo (arco di connessione)

3. Se la somma e' inferiore a quella gia' memorizzata nel nodo J di arrivo allora si memorizza la nuova energia minima e il nuovo **percorso**

Di quali **strutture dati** abbiamo bisogno? Quali **variabili**? Di che **tipo**?
Numero di nodi = **N** (0,1,2,...,N-1)
EMR[I], I=0,1,2,...,N-1 (Una lista!)
Energia minima necessaria per raggiungere il **nodo I**

NODO_PRECEDENTE[I], I=0,1,2,...,N-1 (Una lista!)
Il **percorso** viene descritto memorizzando per ogni nodo I il nodo precedente (primo nodo 0: non significativo)

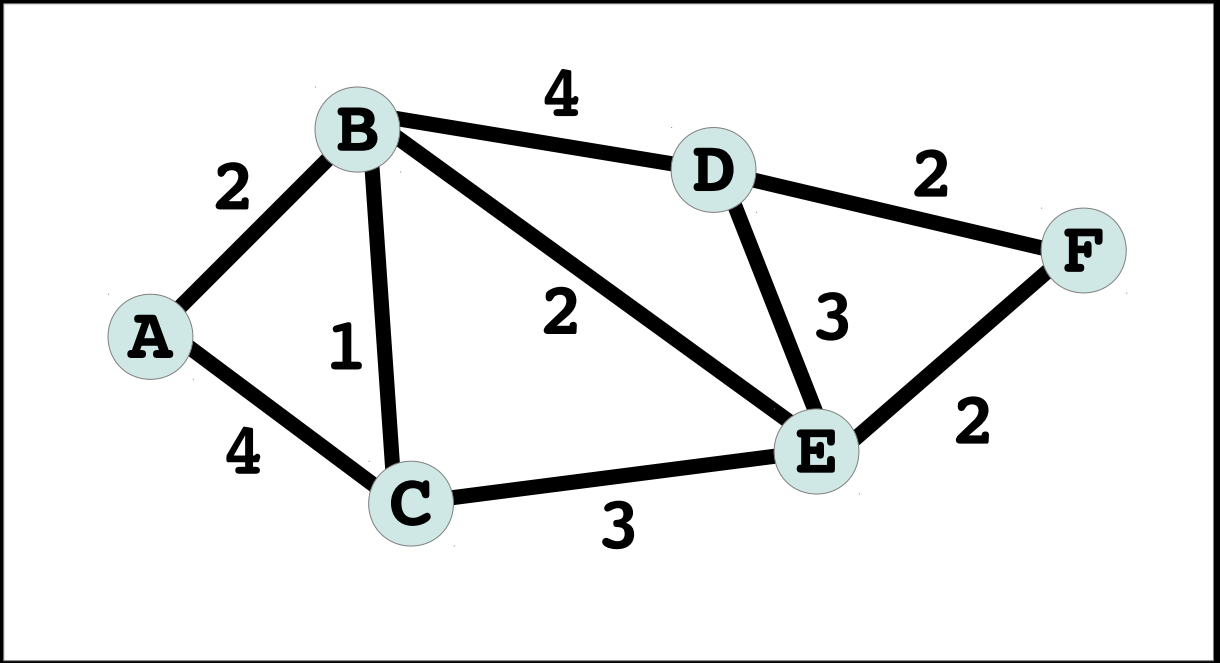
E[I,J],I=0,1,2,...,N-1 ; J=0,1,2,...,N-1
La **matrice** che contiene l'energia per andare dal nodo I al nodo J.

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

Problem Solving
1959: *Dijkstra's Algorithm*

E[I,J] matrice

	A	B	C	D	E	F
A	0	2	4	∞	∞	∞
B	2	0	1	4	2	∞
C	4	1	0	∞	3	∞
D	∞	4	∞	0	3	2
E	∞	2	3	3	0	2
F	∞	∞	∞	2	2	0



	A	B	C	D	E	F
EMR	0	∞	∞	∞	∞	∞
lista						

	A	B	C	D	E	F
NODO_PRECEDENTE	0	?	?	?	?	?
lista						

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

Dijkstra

In un grafo di N nodi, trova il percorso a energia minima per andare da un nodo iniziale N=0 a qualsiasi altro nodo.

inizio

```
I=0,1,2,...,N-1 ; J=0,1,2,...,N-1
E[I,J] la matrice contiene l'energia per andare dal nodo I al nodo J
```

```
EMR[0] = 0, Energia minima per raggiungere il nodo iniziale = 0
EMR[I] = INF, I=1,2,...,N.1
Energia minima per raggiungere tutti gli altri nodi = INFINITO
```

```
NODO_PRECEDENTE[0] = 0
il nodo iniziale non ha nodi precedenti
NODO_PRECEDENTE[I] = ?
Per tutti i nodi I=1,2,...,N-1
il nodo precedente e' incognito
```

```
Per tutti i nodi I = 0,1,2,...,N-1 calcola la nuova energia
per raggiungere tutti i nodi J partendo dal nodo I
NERJ = EMR[I] + E[I,J]
Nuova energia per raggiungere il nodo J (dal nodo I) =
Energia minima per raggiungere il nodo I +
Energia per andare dal nodo I al nodo J
```

NERJ < EMR[J] ?

NO → *fine*

SI →

C'e' un percorso migliore!
Aggiorna energia minima e
aggiorna percorso!

```
EMR[J] = NERJ
Aggiorna energia minima
per raggiungere il nodo J
NODO_PRECEDENTE[J] = I
Il nodo J si raggiunge dal NODO I
```

→ *fine*

Laboratorio di programmazione: Python
MATRICI (liste di liste!)

```
# inizializza la matrice
MATRICE=[]
RIGHE=2
COLONNE=3
# inserisci nella matrice le liste vuote corrispondenti alle righe
for I in xrange(RIGHE):
    MATRICE.append([])
# inserisci gli elementi nella matrice, riga per riga
for I in xrange(RIGHE):
    for J in xrange(COLONNE):
        ELEMENTO= raw_input("Inserisci elemento ")
        MATRICE[I].append(ELEMENTO)
# visualizza la matrice
for I in xrange(RIGHE):
# vai a capo dopo la stampa di una riga
    print
    for J in xrange(COLONNE):
        print MATRICE[I][J] ,
```

Laboratorio di programmazione: Python
MATRICI (liste di liste!)

Visualize Python, Java, JavaScript, TypeScript, and Ruby code execution - Mozilla Firefox

File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto

Visualize Python, Java...

www.pythontutor.com/visualize.html#mode=display

Python 2.7

```
1  # inizializza la matrice
2  MATRICE=[]
3  RIGHE=2
4  COLONNE=3
5  # inserisci nella matrice le liste vuote corrispondenti
6  for I in xrange(RIGHE):
7      MATRICE.append([])
8  # inserisci gli elementi nella matrice, riga per riga
9  for I in xrange(RIGHE):
10     for J in xrange(COLONNE):
11         ELEMENTO= raw_input("Inserisci elemento ")
12         MATRICE[I].append(ELEMENTO)
13 # visualizza la matrice
14 for I in xrange(RIGHE):
15 # vai a capo dopo la stampa di una riga
16     print
17     for J in xrange(COLONNE):
18         print MATRICE[I][J] ,
```

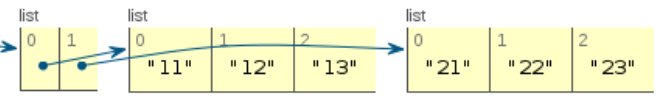
Edit code

Frames

Global frame

MATRICE	
RIGHE	2
COLONNE	3
I	1
J	2
ELEMENTO	"23"

Objects



Program output:

```
Inserisci elemento 11
Inserisci elemento 12
Inserisci elemento 13
Inserisci elemento 21
Inserisci elemento 22
Inserisci elemento 23

11 12 13
21 22 23
```

Laboratorio di programmazione: Python
MATRICI (liste di liste!)

```
# inizializza la matrice E[I,J]
E=[]
N=6
# inserisci nella matrice le liste vuote corrispondenti alle righe
for I in xrange(N):
   E.append([])
# inserisci gli elementi nella matrice, riga per riga
for I in xrange(N):
   for J in xrange(N):
      print "inserisci distanza", I, J
      DISTANZA = raw_input("Inserisci distanza")
      E[I].append(DISTANZA)
# visualizza la matrice
for I in xrange(N):
# vai a capo dopo la stampa di una riga
   print
   for J in xrange(N):
      print E[I][J] ,
```

```
E[I,J]
0       2    4 1000 1000 1000
2       0    1    4    2 1000
4       1    0 1000    3 1000
1000    4 1000    0    3    2
1000    2    3    3    0    2
1000 1000 1000    2    2    0
```

Laboratorio di programmazione: Python
MATRICI (liste di liste!)

```
# inizializza la matrice E[I,J]
E=[]
N=6
# inserisci nella matrice le liste vuote corrispondenti alle righe
for I in xrange(N):
   E.append([])
# inserisci gli elementi nella matrice, riga per riga
# riga 0
E[0]=[   0,    2,    4, 1000, 1000, 1000]
E[1]=[   2,    0,    1,    4,    2, 1000]
E[2]=[   4,    1,    0, 1000,    3, 1000]
E[3]=[1000,    4, 1000,    0,    3,    2]
E[4]=[1000,    2,    3,    3,    0,    2]
E[5]=[1000, 1000, 1000,    2,    2,    0]
# visualizza la matrice
for I in xrange(N):
# vai a capo dopo la stampa di una riga
   print
   for J in xrange(N):
      print E[I][J] ,
```

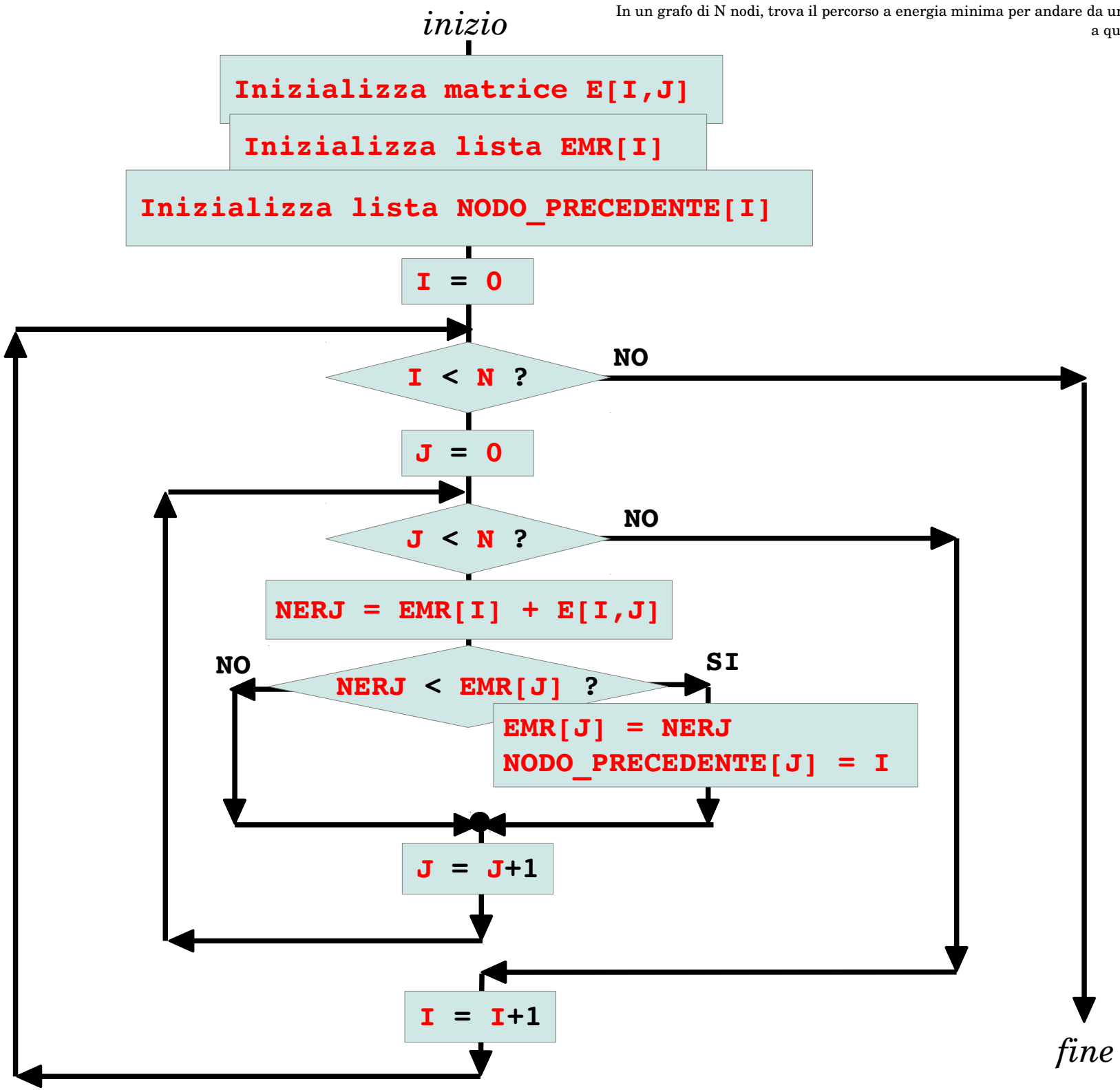
```
E[I,J]
0       2    4 1000 1000 1000
2       0    1    4    2 1000
4       1    0 1000    3 1000
1000    4 1000    0    3    2
1000    2    3    3    0    2
1000 1000 1000    2    2    0
```

```
N=6

# inizializza la lista EMR[I]
EMR=[]
EMR.append(0) # energia per raggiungere nodo iniziale
for I in xrange(1,N,1):
   EMR.append(1000)
print "EMR "
print EMR

# inizializza la lista NODO_PRECEDENTE
NODO_PRECEDENTE=[]
NODO_PRECEDENTE.append(0) # nodo iniziale
for I in xrange(1,N,1):
   NODO_PRECEDENTE.append(1000)
print "NODO_PRECEDENTE "
print NODO_PRECEDENTE
```

```
EMR
[0, 1000, 1000, 1000, 1000, 1000]
NODO_PRECEDENTE
[0, 1000, 1000, 1000, 1000, 1000]
```

Dijkstra

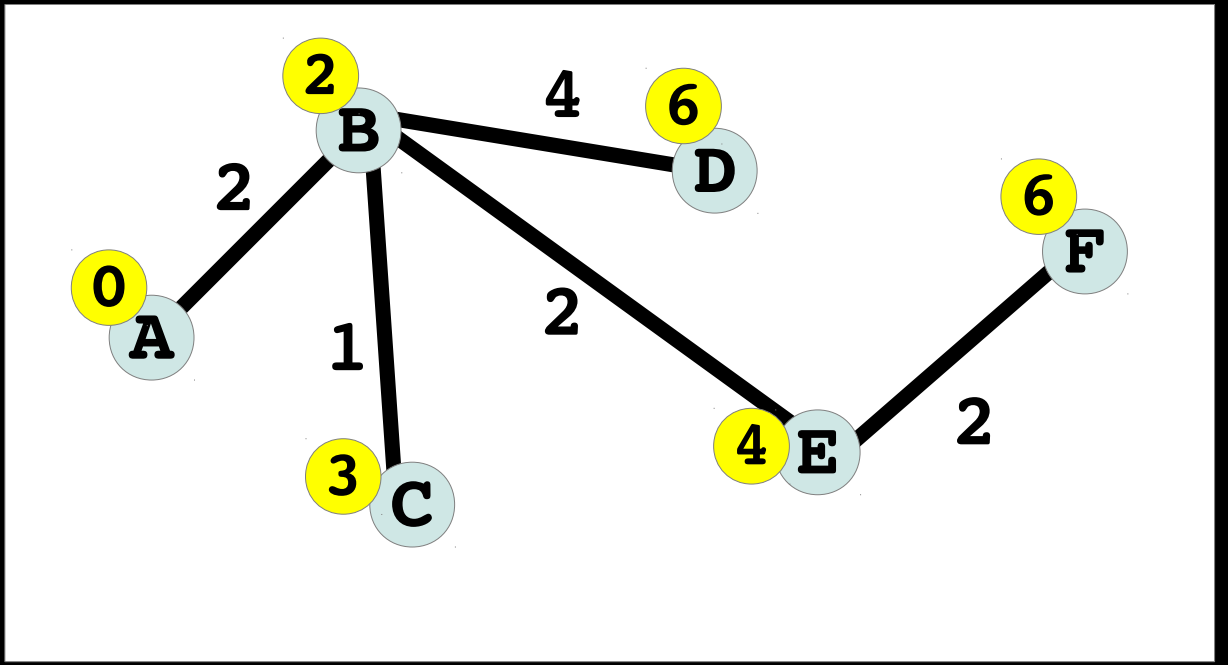
In un grafo di N nodi, trova il percorso a energia minima per andare da un nodo iniziale N=0 a qualsiasi altro nodo.



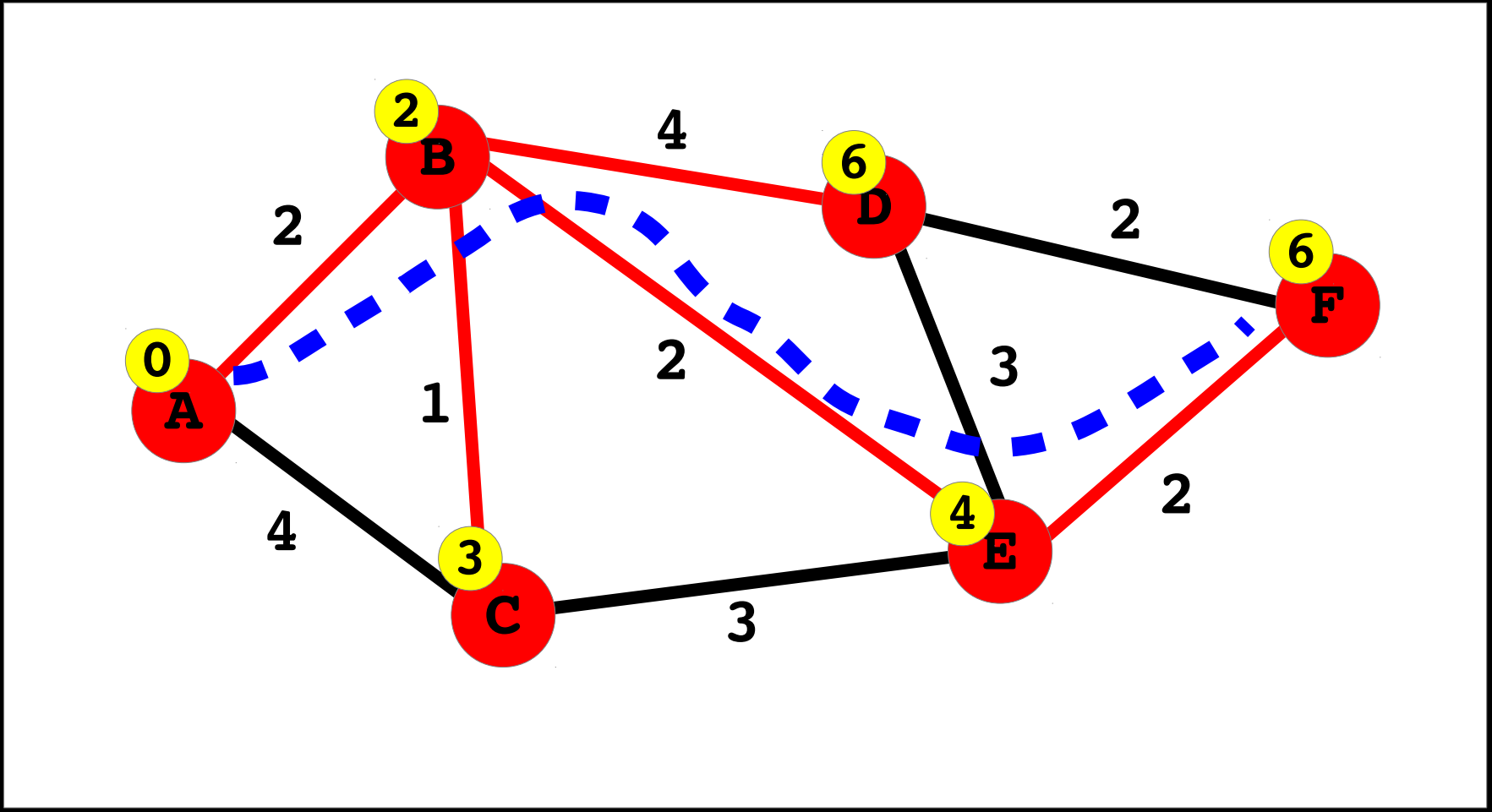
Laboratorio di programmazione: Python

```
N=6
# inizializza la matrice E[I,J]
# inizializza la lista EMR[I]
# inizializza la lista NODO_PRECEDENTE

# visita tutta la matrice
for I in xrange(N):
    for J in xrange(N):
       NERJ = EMR[I] + E[I][J]
       if (NERJ < EMR[J]):
           EMR[J]=NERJ
           NODO_PRECEDENTE[J]=I
print "EMR"
print EMR
print "NODO_PRECEDENTE"
print NODO_PRECEDENTE
```

```
EMR
[0, 2, 3, 6, 4, 6]
 A  B  C  D  E  F
NODO_PRECEDENTE
[0, 0, 1, 1, 1, 4]
 0  A  B  B  B  E
 A  B  C  D  E  F
```



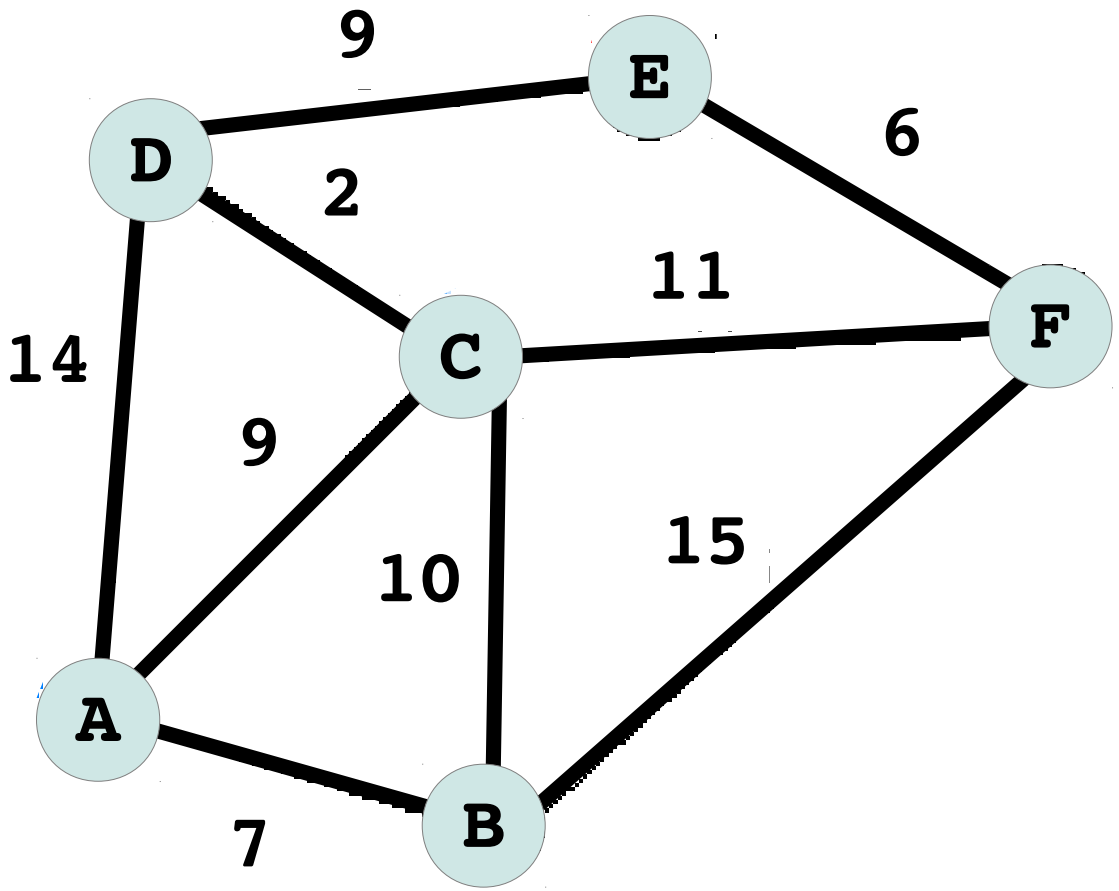
Problem Solving
1959: *Dijkstra's Algorithm*



Problema: come descrivere la soluzione al computer?
Quali strutture dati? Quali variabili?
Il pensiero computazionale ci aiuta ad affrontare questo problema ...

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

Problem Solving
1959: *Dijkstra's Algorithm*



Trovare il percorso a *energia minima* da A a F

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

Laboratorio di programmazione: Python
MATRICI (liste di liste!)

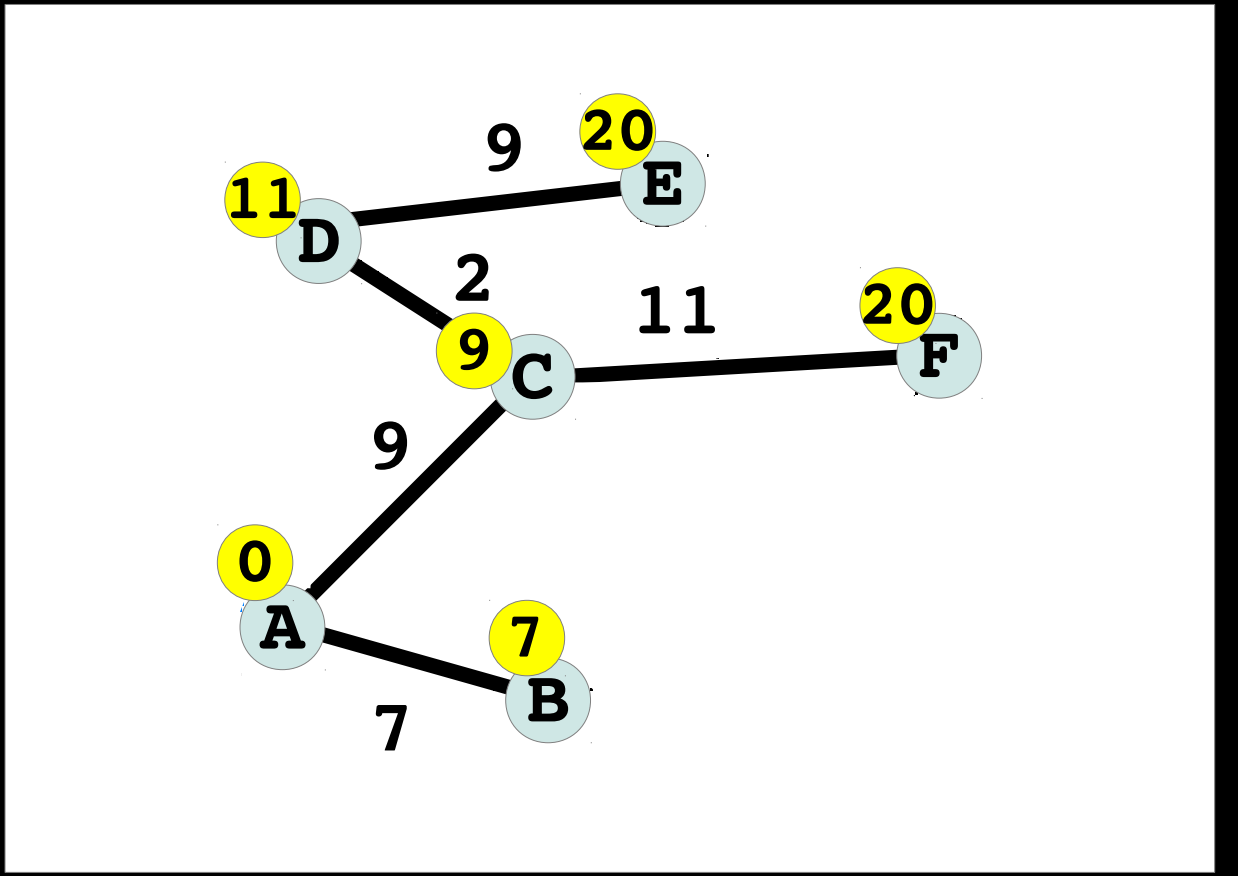
```
# inizializza la matrice E[I,J]
E=[]
N=6
# inserisci nella matrice le liste vuote corrispondenti alle righe
for I in xrange(N):
   E.append([])
# inserisci gli elementi nella matrice, riga per riga
# riga 0
E[0]=[   0,    7,    9,   14, 1000, 1000]
E[1]=[   7,    0,   10, 1000, 1000,   15]
E[2]=[   9,   10,    0,    2, 1000,   11]
E[3]=[  14, 1000,    2,    0,    9, 1000]
E[4]=[1000, 1000, 1000,    9,    0,    6]
E[5]=[1000,   15,   11, 1000,    6,    0]
# visualizza la matrice
for I in xrange(N):
# vai a capo dopo la stampa di una riga
   print
   for J in xrange(N):
      print E[I][J] ,
```

```
E[I,J]
0       7    9   14 1000 1000
7       0   10 1000 1000   15
9      10    0    2 1000   11
14   1000    2    0    9 1000
1000 1000 1000    9    0    6
1000   15   11 1000    6    0
```

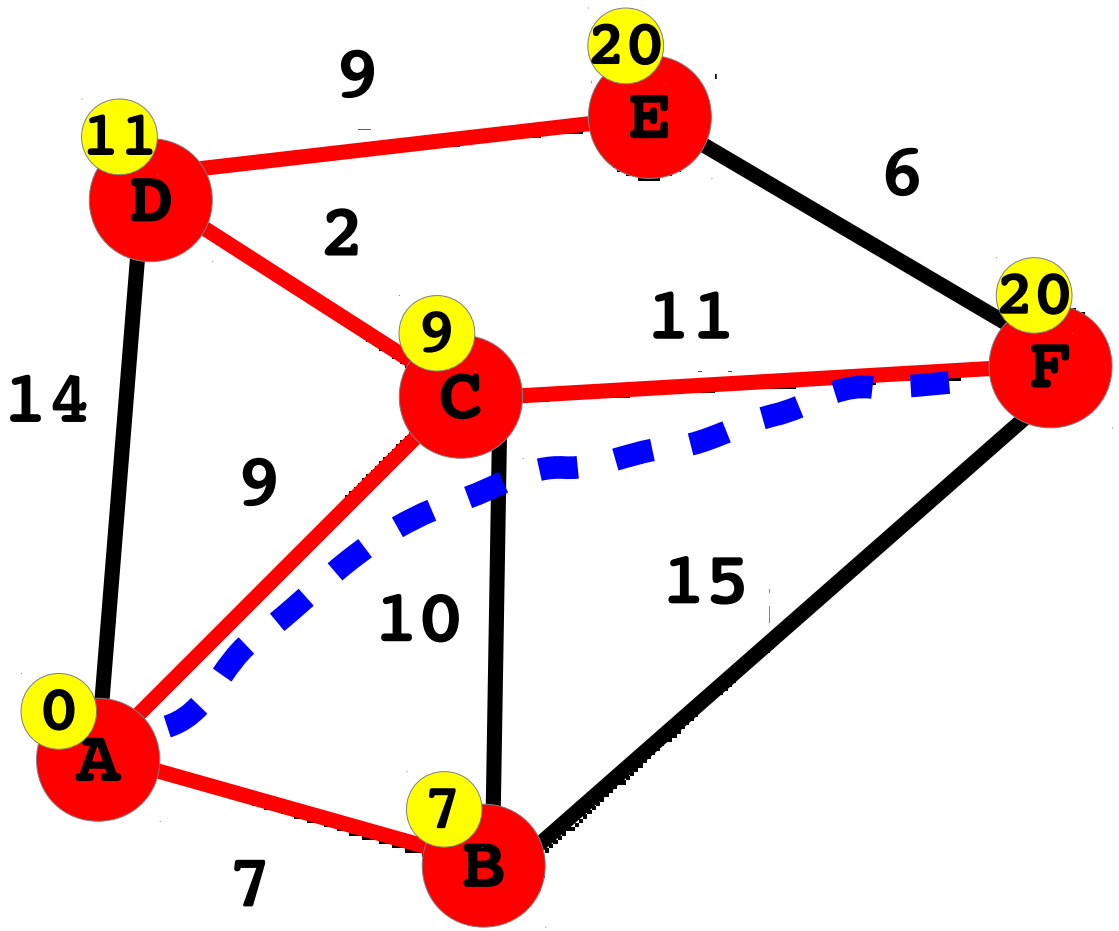
Laboratorio di programmazione: Python

```
N=6
# inizializza la matrice E[I,J]
# inizializza la lista EMR[I]
# inizializza la lista NODO_PRECEDENTE

# visita tutta la matrice
for I in xrange(N):
    for J in xrange(N):
       NERJ = EMR[I] + E[I][J]
       if (NERJ < EMR[J]):
           EMR[J]=NERJ
           NODO_PRECEDENTE[J]=I
print "EMR"
print EMR
print "NODO_PRECEDENTE"
print NODO_PRECEDENTE
```

```
EMR
[0, 7, 9,11,20,20]
 A  B  C  D  E  F
NODO_PRECEDENTE
[0, 0, 0, 2, 3, 2]
 0  A  A  C  D  C
 A  B  C  D  E  F
```



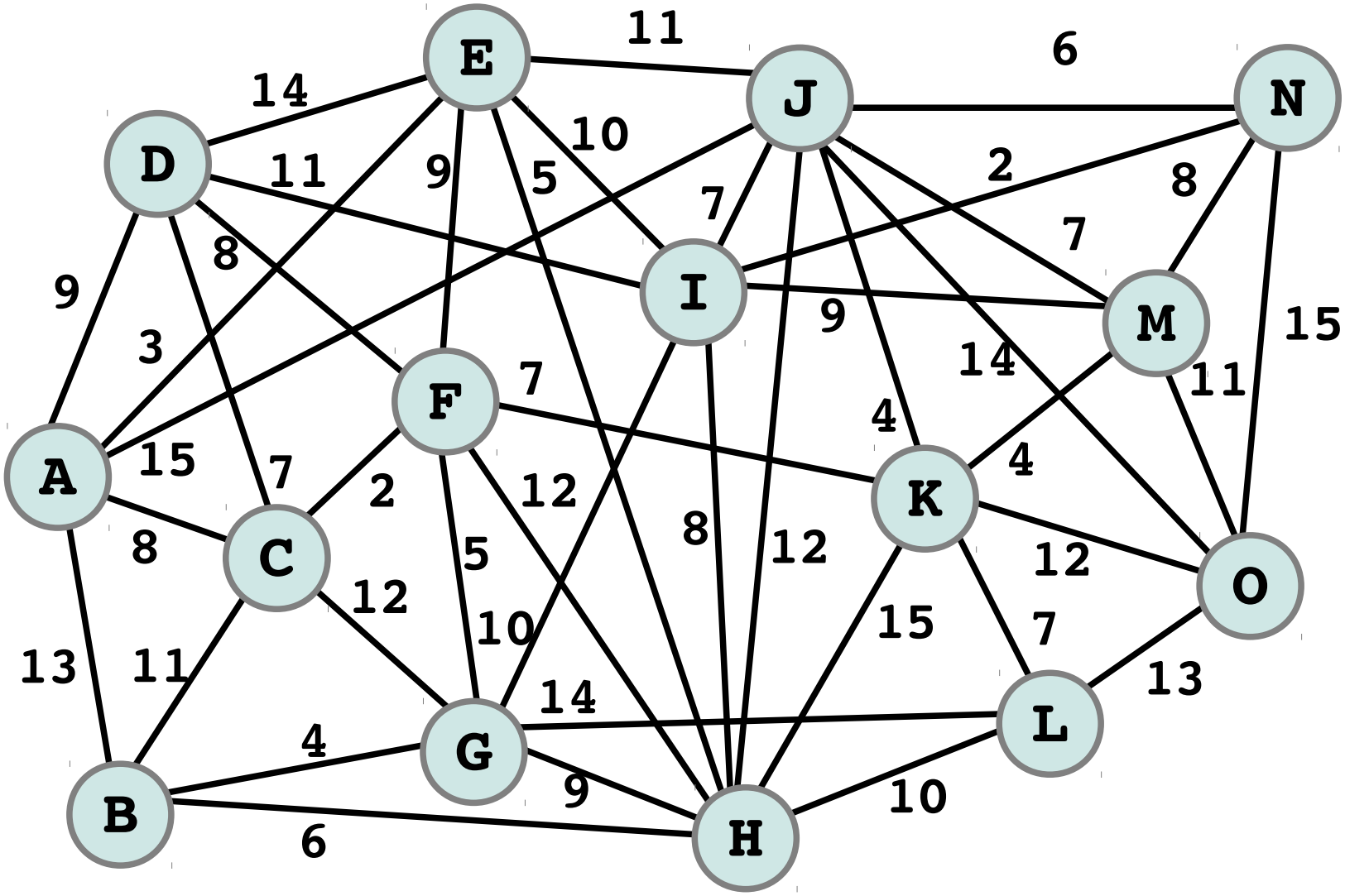
Problem Solving
1959: *Dijkstra's Algorithm*



Come descrivere la soluzione al computer?
Quali strutture dati? Quali variabili?
Pensiero computazionale!

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

Problem Solving
1959: *Dijkstra's Algorithm*



Trovare il percorso a *energia minima* da A a O

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

```
N=15

# inizializza la lista EMR[I]
EMR=[]
EMR.append(0) # energia per raggiungere nodo iniziale
for I in xrange(1,N,1):
   EMR.append(100)
print "EMR "
print EMR

# inizializza la lista NODO_PRECEDENTE
NODO_PRECEDENTE=[]
NODO_PRECEDENTE.append(0) # nodo iniziale
for I in xrange(1,N,1):
   NODO_PRECEDENTE.append(100)
print "NODO_PRECEDENTE "
print NODO_PRECEDENTE
```

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

```
# inizializza la matrice E[I,J]
E=[]
# inserisci nella matrice le liste vuote corrispondenti alle righe
for I in xrange(N):
    E.append([])
# inserisci gli elementi nella matrice, riga per riga
#        A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
E[0]=[  0, 13,  8,  9,  3,100,100,100,100, 15,100,100,100,100,100]
E[1]=[ 13,  0, 11,100,100,100,  4,  6,100,100,100,100,100,100,100]
E[2]=[  8, 11,  0,  7,100,  2, 12,100,100,100,100,100,100,100,100]
E[3]=[  9,100,  7,  0, 14,  8,100,100, 11,100,100,100,100,100,100]
E[4]=[  3,100,100, 14,  0,  9,100,  5, 10, 11,100,100,100,100,100]
E[5]=[100,100,  2,  8,  9,  0,  5, 12,100,100,  7,100,100,100,100]
E[6]=[100,  4, 12,100,100,  5,  0,  9, 10,100,100, 14,100,100,100]
E[7]=[100,  6,100,100,  5, 12,  9,  0,  8, 12, 15, 10,100,100,100]
E[8]=[100,100,100, 11, 10,100, 10,  8,  0,  7,100,100,  9,  2,100]
E[9]=[ 15,100,100,100, 11,100,100, 12,  7,  0,  4,100,  7,  6, 14]
E[10]=[100,100,100,100,100,  7,100, 15,100,  4,  0,  7,  4,100, 12]
E[11]=[100,100,100,100,100,100, 14, 10,100,100,  7,  0,100,100, 13]
E[12]=[100,100,100,100,100,100,100,100,  9,  7,  4,100,  0,  8, 11]
E[13]=[100,100,100,100,100,100,100,100,  2,  6,100,100,  8,  0, 15]
E[14]=[100,100,100,100,100,100,100,100,100, 14, 12, 13,  11,15,  0]
# visualizza la matrice
for I in xrange(N):
# vai a capo dopo la stampa di una riga
    print
    for J in xrange(N):
        print E[I][J] ,
```

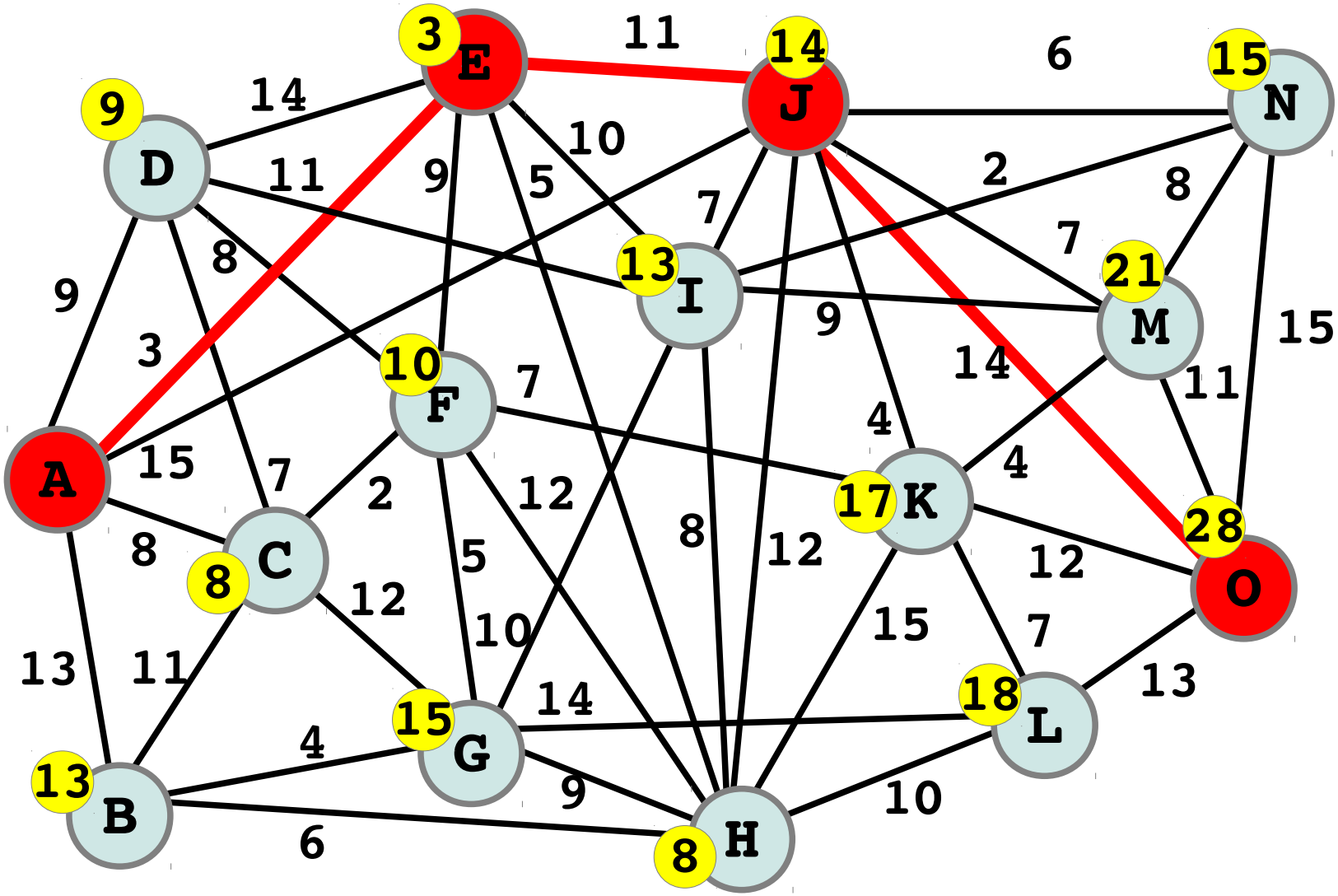
Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

```
# visita tutta la matrice
for I in xrange(N):
    for J in xrange(N):
       NERJ = EMR[I] + E[I][J]
       if (NERJ < EMR[J]):
           EMR[J]=NERJ
           NODO_PRECEDENTE[J]=I
print "EMR"
print EMR
print "NODO_PRECEDENTE"
print NODO_PRECEDENTE
```

```
EMR
[0, 13,  8,  9,  3, 10, 15,  8, 13, 14, 17, 18, 21, 15, 28]
 A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
 0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
NODO_PRECEDENTE
 A   B   C   D   E   F   G   H   I   J   K   L   M   N   O
[0,  0,  0,  0,  0,  2,  5,  4,  4,  4,  5,  7,  9,  8,  9]
 A   A   A   A   A   C   F   E   E   E   F   H   J   I   J
```

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.

Problem Solving
1959: *Dijkstra's Algorithm*



Trovare il percorso a *energia minima* da A a O

Source: Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs", Numerische Mathematik 1: 269–271.