7. Shortest Path Case Study (ch. 5)

Problem definition

a. Definitions

- $G = (V, E)$ — directed graph
- V = vertices in the range 0...(N-1)
- E = edges of the form (i,j)
- path(i,j) = sequence of edges starting with i and ending with j
- cycle(i) = a path starting and ending with i
- $w : E \rightarrow \mathbb{N}$ — positive weights associated with each edge, edge weight function
- path length = sum of weights along the path
- $W : V \times V \rightarrow \mathbb{N}$ — the <u>edge weight matrix</u> is a useful extension

```
W(i,j) = w(i,j)  if (i,j) ∈ E
       ~ ∞       if (i,j) ∉ E
       ~ 0       if i=j
```

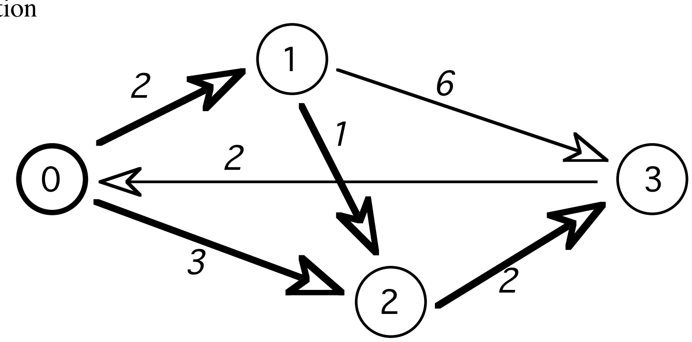
- $D : V \times V \rightarrow \mathbb{N}$ — the <u>shortest length</u> over all paths from i to j

b. Problem specification

- write a terminating program which computes D in some matrix d

```
true leads-to FP
FP ⇒ (d = D)
```

c. Illustration



d. Methodology

1. refine the specification by examining alternatives
2. seek to discover a program that meets the specification
3. establish which architecture is compatible with the program

Refinement 1

Refined specification

a. Find alternative formulation of the relation between d and D

```
1. FP ≡ ⟨ ∀ i,j :: d[i,j] = ⟨ min k :: d[i,k] + d[k,j] ⟩ ⟩
2. inv. d[i,j] ≤ W(i,j) ∧
              (d[i,j] is the length of some path from i to j)
3. ¬FP ∧ (num,sum) = (m,n) → (num,sum) < (m,n)

       num = ⟨ Σ i,j : d[i,j] = ∞ :: 1 ⟩
       sum = ⟨ Σ i,j : d[i,j] ≠ ∞ :: d[i,j] ⟩
```

Refinement correctness

```
a. (1) and (2) ⇒ d=D
   — by induction on the number m of edges in the shortest path

   Base case:  m=1 and d[i,j] = W(i,j) since there is only one edge
   Inductive step:
       -   consider (x,y) having the shortest path p of length (m+1)
       -   assume p = p'.(z,y) where the number of edges in p' is m
       -   d[x,y] ≤ d[x,z] + d[z,y] due to (1)
       -   d[x,y] is the length of some path due to (2)
       -   d[x,y] must equal D(x,y)

b. (3) guarantees termination due to well-foundness and induction
```

Program

a. Replace the "=" by ":=" in the FP expression

b. Select an appropriate initialization

```
Program SP1 (P5 in the text)
    initial ⟨ ‖ i,j :: d[i,j] = W(i,j) ⟩
    assign ⟨ ‖ i,j :: d[i,j] := ⟨ min k :: d[i,k] + d[k,j] ⟩ ⟩
end
```

c. Illustration

```
    d(0,3) = min over k := 0..3   0+∞  2+6  3+2  ∞+0
```

d. Correctness follows trivially from the proof above

Architecture

a. Synchronous parallel architecture

b. N^3 processors require $O(\log^2 N)$ steps

-- N^3 processors compute d[i,k] + d[k,j] in constant time

-- min can be computed in $O(\log N)$ steps (tree shaped computation)

-- FP is achieved in $O(\log N)$ assignments

let m be the number of assignments executed so far

```
inv. d[i,j] is the shortest distance along a path
       using at most 2^m-1 intermediate vertices

          m=0 no intermediate vertices (initilization)
          m=1 one intermediate vertex (i,k).(k,j)
          m=2 three intermediate vertices (i,a).(a,k) . (k,b).(b,j)
          d[i,k] + d[k,j] involves (2^m-1 + 2^m-1 +1) = 2^(m+1)-1
                 intermediate vertices (vertex k needs to be added)
```

Refinement 2

Refined specification

a. Revisit the FP definition and replace the minimization with comparisons against d[i,j]

```
   1. FP ≡ ⟨ ∀ i,j :: d[i,j] = ⟨ min k :: d[i,k] + d[k,j] ⟩ ⟩
is replced by
   1'. FP ≡ ⟨ ∀ i,j,k :: d[i,j] = min(d[i,j], d[i,k] + d[k,j]) ⟩
```

Refinement correctness

a. Preserved.

Program

a. Adjust the form of the assignment

b. Replace the "‖" by "[]" to avoid multiple values being assigned to d[i,j]

```
Program SP2 (P1 in the text)
   initial ⟨ ‖ i,j :: d[i,j] = W(i,j) ⟩
   assign ⟨ [] i,j,k :: d[i,j] := min(d[i,j], d[i,k] + d[k,j]) ⟩
end
```

c. Illustration

```
   d(0,3) = ∞ for k := 0..3
   is replaced (if possible) by  0+∞  2+6 (yes)  3+2 (yes)  ∞+0
```

Architecture

a. Sequential architecture

b. $O(N^3)$ assignments if we allow i and j to run faster than k

-- (k i j) := (k i j) **+** (0 0 1)

treating the triple as a 3-digit number in base N

let m be the number of assignments executed so far

Refinement 3

Program

a. Make the sequencing explicit -- Floyd-Warshall

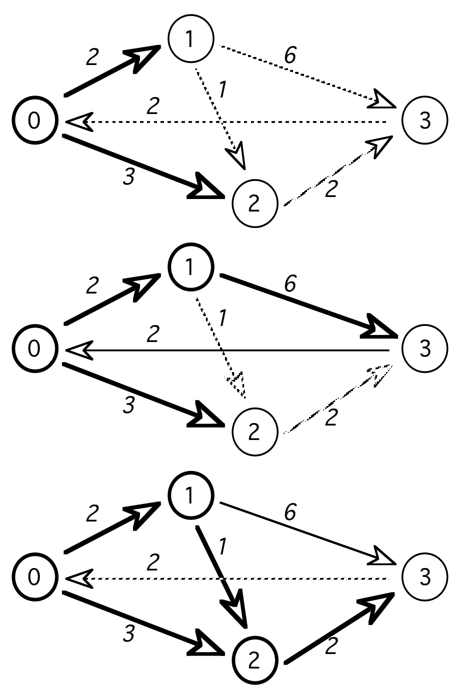
```
Program SP3 (P3 in the text)
initial   〈 ‖ i,j :: d[i,j] = W(i,j) 〉
        ‖  x,u,v = 0,0,0
assign     d[u,v] := min(d[u,v], d[u,x] + d[x,v])
        ‖  (x u v) := (x u v) + (0 0 1) if (x,u,v) ≠ (N-1,N-1,N-1)
end
```

Proof

a. Define H(i,j,k) to be the minimum length over all paths from i to j and using only intermediate vertices in the range 0 to (k-1)

```
H(i,j,0) = W(i,j)  — since the range is empty
H(i,j,k+1) = min(H(i,j,k), H(i,k,k)+H(k,j,k))
```

b. Illustration



c. Show that the definition is correct

```
Base case:  k=0 and H(i,j,0) = W(i,j)
— since the range is empty
Inductive step:
-   k is not on the shortest path from i to j using vetices 0 to k
        -- H(i,j,k+1) = H(i,j,k)
-   k is on the shortest path from i to j using vetices 0 to k
        -- H(i,j,k+1) = H(i,k,k)+H(k,j,k)
            -   k is on the shortest path
            -   the distances from j and i are minimal
            -   if the equality does not hold,
                there must be a shorter path through k (?)
```

d. Proof of progress is trivial

e. The following invariant is needed

```
inv.   (d[i,j] is the length of some path from i to j) ∧
       ⟨ ∀ i,j : (i,j)<(u,v) :: d[i,j] ≤ H(i,j,x+1) ⟩ ∧
       -- given the current value of x
          some (i,j) distances have been updated
       ⟨ ∀ i,j : (i,j)≥(u,v) :: d[i,j] ≤ H(i,j,x) ⟩
       -- while the rest of (i,j) distances have not been updated
          (one step lag)
```

Refinement 4

Program

a. The updates could be carried out in parallel

b. The proof is not affected

```
Program SP4 (P4 in the text)
initial   ⟨ ‖ i,j :: d[i,j] = W(i,j) ⟩
          ‖   k = 0
assign    ⟨ ‖ i,j :: d[i,j] := min(d[i,j], d[i,k] + d[k,j]) if k<N ⟩
          ‖   k := k+1 if k<N
end
```

Refinement 5

Program

a. Try to use H in the program

b. Employ the equational schema

```
Program SP5 (P2 in the text)
always
   ⟨ [] i,j :: H(i,j,0) = W(i,j) ⟩
[] ⟨ [] k :: ⟨ ‖ i,j ::
              H(i,j,k+1) = min(H(i,j,k), H(i,k,k)+H(k,j,k)) ⟩ ⟩
[] ⟨ [] i,j :: d(i,j) = H(i,j,N) ⟩
end
```

Refinement 6

Program

a. In SP4 the following invariant holds

```
inv. d[i,j] = H(i,j,k) ∧
   (d[i,j] is the length of some path from i to j) ∧ k≤N
```

b. We note that $H(i,j,k+m) \le H(i,j,k)$ with $(k < k+m \le N)$
— the metric can only be reduced as we consider more vertices

c. Let each processor have its own local variable k[i,j] and access distances on other processors only if their values of k are at least as far along

```
Program SP6 (P6 in the text)
initial ⟨ ‖ i,j :: d[i,j], k[i,j] = W(i,j), 0 ⟩
assign ⟨ [] i,j ::
          d[i,j], k[i,j] :=
              min(d[i,j], d[i,k[i,j]] + d[k[i,j],j]), k[i,j]+1
              if k[i,j]<N ∧ k[i, k[i,j]]≥k[i,j] ∧ k[k[i,j],j]≥k[i,j] ⟩
end
```

d. Read-only schema