All Pairs Shortest Path

```
procedure Floyd (var A:array[1..n,1..n] of real; C:array[1..n,1..n] of real);
{Floyd computes shortest path matrix A given arc cost matrix C }
var
        i,j,k : integer;
begin
        for i:= 1 to n do           {copy C into A }
                for j:= 1 to n do
                        A[i,j] := C[i,j];

        for i:= 1 to n do           {set distance to self as 0 }
                A[i,i] := 0;

        for k:= 1 to n do                   {compute shortest paths }
                for i:= 1 to n do
                        for j:= 1 to n do
                                if  A[i,k] + A[k,j] < A[i,j]  then
                                        A[i,j] := A[i,k] + A[k,j];
end;  {Floyd}
```

```
procedure shortest (var A:array[1..n,1..n] of real; C:array[1..n,1..n] of real; P:array[1..n,1..n] of integer);
{shortest takes an n X n matrix C of arc costs and produces an n X n matrix A of lengths of shortest paths and
 an n X n matrix P giving a point in the "middle" of each shortest path }
var
        i,j,k : integer;
begin
        for i:= 1 to n do           {copy C into A }
                for j:= 1 to n do
                begin
                        A[i,j] := C[i,j];
                        P[i,j] := 0;      { set P to zero }
                end

        for i:= 1 to n do           {set distance to self as 0 }
                A[i,i] := 0;

        for k:= 1 to n do                   {compute shortest paths }
                for i:= 1 to n do
                        for j:= 1 to n do
                                if  A[i,k] + A[k,j] < A[i,j]  then
                                begin
                                        A[i,j] := A[i,k] + A[k,j];
                                        P[i,j] := k                 {record the middle point }
                                end
end;  {Floyd}
```

---

path algorithm on next page ☺

```
procedure path ( i, j : integer);
var
        k : integer;
begin
        k := P[i,j];            {set k to midpoint of  i → j }

        if k = 0 then           {if there is a direct path then ground the recursion }
                return;

        path(i,k);              {get the midpoint between  i → k }

        writeln(k);             {process the midpoint }

        path(k,j);              {get the midpoint between k → j }

end;
```