MINIMUM LENGTH PATHS (Dynamic Programming)

Given: a graph with vertices and labeled edges. Suppose the labels are positive integers which could be the edge’s length, cost, weight, time, ... . Number the graph vertices 1, 2, ..., n. The edge labels determine entries in an $n \times n$ matrix $g$. If the edge-label between vertices $i$ and $j$ is 5, then $g(i,j)=5$. There are 0’s on the diagonal. If there is no edge between vertices $i$ and $j$, $g(i,j)=\infty$, infinity, indicating impossibility. For this graph, the vertices are 1, 2, 3, 4.

The matrix for this graph is the $4 \times 4$ matrix $g$ such that $g(1,2)=5$, $g(1,3)=2$, $g(2,3)=2$, ...

$$
\begin{pmatrix}
1 & 2 & \text{inf} & \text{inf} \\
2 & 0 & 2 & \text{inf} \\
3 & 2 & 0 & 4 \\
4 & \text{inf} & 1 & 4
\end{pmatrix}
$$

Min lengths $\min_g = 
\begin{pmatrix}
1 & 2 & 3 & 4 \\
1 & 0 & 4 & 2 & 5 \\
2 & 4 & 0 & 2 & 1 \\
3 & 2 & 0 & 3 \\
4 & 5 & 1 & 3 & 0
\end{pmatrix}.

Given the matrix $g$, we want to find the matrix $\min_g$ such that $\min_g(i,j)$ = minimum distance between $i$ and $j$ = the total length (not number of steps) of a path from $i$ to $j$ of minimum total length. We start with calculating the minimum lengths $\min_g$ of paths of whose number of steps is $nsteps = 1, 2, 4, 8, ...$ or less until $nsteps >= \maxsteps$. When $nsteps=1$, $\min_g = g$. A path of length $2*nsteps$ has an intermediate point $k$ with $nsteps$ before $k$ and $nsteps$ after $k$. If there are $n$ vertices, $\maxsteps$, the maximum steps needed for a min-length path, is $n-1$. If 3-2-5 is a min-length path, so are the paths 3-2-5-5, 3-2-5-5-5. Hence more steps doesn’t mean longer length.

CLASSWORK 23.1(6) minpath.f95 Suppose $g$ is the matrix for an undirected (you can go either way on an edge) graph with $n$ points with lengths on its edges. Write a subroutine $\text{min_length}(g, \min_g, n)$ which calculates the matrix $\min_g$ such that $\min_g(i,j)$ = the minimum length of a path between points $i$ and $j$ of $g$.

$\text{find_path}(g, \min_g, n, \text{start}, \text{end}, \text{path}, m)$ finds a min-length path $\text{path}=[\text{path}(1) ... \text{path}(m)]$ from $\text{start} = \text{path}(1)$ to $\text{path}(m)=\text{end}$ (find_path is not on quiz).
Subroutine min_length(g, min_g, n)
!
*implicit none*

integer:: g(n,n), min_g(n,n), double(n)
integer:: i, j, k, nsteps, maxsteps

nsteps=1; min_g=g; maxsteps=n-1

!finish code to find min-length paths of >= 1 step
enddo

endsubroutine

Subroutine find_path(g, min_g, n, start, end, path, m)
!
$m = \text{path length, } n = \text{dim of } g, a$

integer:: g(n,n), min_g(n,n)
integer:: path(n), start, end, mm(1)

m=1; path(1)=start;

do
if(path(m)==end) exit;

mm=minloc(max(g(path(m),1:n),1)+min_g(1:n,end))

m=m+1; path(m)=mm(1);
enddo

10 format(i1,a,i1,a,i1,a,i1,i2,a,200(i2))
print 10, start, "-", end, " length: ", & & min_g(1,end), " path: ", path(1:m)
endsubroutine

Program test_min_length

integer:: g(4,4)=1, min_g(4,4), path(4)

integer:: inf=10**6, n=4

g(1,1:n)=(/0,5,2,inf/)
g(2,1:n)=(/5,0,2,1/)
g(3,1:n)=(/2,2,0,4/)
g(4,1:n)=(/inf,1,4,0/)
call min_length(g, min_g, n)
call print_graph(g, n)
call print_graph(min_g, n)
do j=1,4
   call find_path(g, min_g, n, 1, j, path, m)
enddo
endprogram

Subroutine print_graph(a, n)

integer:: a(n,n)

11 format(4x,20(2x,a,i1,a)
print 11, ("\(, i, \)"), i=1,n)

12 format(3x,"\((, i, \)\)), 200(i2,3x)
do i=1,n; print 12, i, a(i,1:n); enddo; print*
endsubroutine

Masks

Lecture 21 reading assignment.

A mask is a vector or array of truth values.

If $a = [-1, 0, 1]$, $a > 0$ is the mask of truth values $[F, F, T]$.

If $b = [0, 0, 0]$, then $a == b$ is $[F, T, F]$. For any mask $m$, $\text{all}(m)$ is true iff all values are true.

any($m$) is true iff some value is true.

count($m$) is the number of values which are true.

sum($a, m$) is the sum of all values in $a$ where $m$ is true.

where($m$) does ... at positions where $m$ is true, and *** otherwise.
TRAVELING SALESMAN PROBLEM

Suppose the nodes of a graph are cities (assume there are two or more) and the edges are distances between the cities. For this graph, the vertices (nodes, cities) are 1, 2, 3, 4, 5.

The matrix for this graph is the 5 × 5 matrix \( g \) such that

\[
g = \begin{bmatrix}
0 & 5 & 2 & \text{inf} & 1 \\
5 & 0 & 2 & 1 & \text{inf} \\
2 & 2 & 0 & 4 & \text{inf} \\
\text{inf} & 1 & 4 & 0 & 1 \\
1 & \text{inf} & \text{inf} & \text{inf} & 1
\end{bmatrix}
\]

A circuit is a path which starts and ends at the same city and visits all other cities exactly once. It need not cross all edges. The length of the circuit is the sum of the lengths of the edges along the path. The Traveling Salesman Problem is finding (if possible) a circuit with the shortest length. To enable recursion, we generalize this problem as follows: Given a partial circuit, i.e., a path which does not visit any city more than once, extend it (if possible) to a circuit which has the shortest length of any such extension. \( g \) will be the graph’s matrix, \( n \) the number of nodes (vertices). \( \text{path} \) will be a partial circuit, \( \text{npath} \) the number of its nodes. \( \text{circuit} \) will range over circuits, \( \text{cir_len} \) being the circuit’s length.

CLASSWORK 23.2(6) mincircuit.f95 Write a subroutine \( \text{min_circuit} \) which given the matrix \( g \) of an \( n \)-node graph, finds (if possible) a min-length \( \text{circuit} \) which extends a path of \( \text{npath} \) nodes. Set \( \text{cir_len} \) to \( \text{inf} \) if not possible.

-- continued

After completing mincircuit next page.

CLASSWORK 23.1(6) minpath 23.2(6) mincircuit

email: dale@math.hawaii.edu subject line: 190 c23(12)

Quizzed on the subroutine \( \text{max_length} \) (same for A and B). Not quizzed on mindrillpath.
recursive subroutine min_circuit (g, n, path, npath, circuit, cir_len)
! implied none! g = graph of n nodes. npath = # cities in path!
! assume path has no repeated cities.
! must extend path to complete circuit and or set cir_len to inf.
integer :: g(n,n), path(n+1), path_next(n+1)
integer :: n, next, npath, cir_len, cir_lenx, i
integer, parameter :: inf=10**6
if (npath==n) then ! base case, no unvisited cities
    cir_len = 0
    do i = 1, npath
        cir_len = cir_len + g(circuit(i), circuit(i+1))
    enddo
    return
endif
cir_len = inf ! recursively reduce to fewer unvisited cities
do next = 1, n " next = next possible city
    if (any(path(1:npath)==next)) cycle
    if (g(path(npath), next)==inf) cycle
! make path_next = path+next
    if (cir_lenx < cir_len) then
        circuit = circuitx; cir_len = cir_lenx
        enddo
endsubroutine
endprogram
In a **directed** graph, the edges have arrows indicating the allowed direction. A **cycle** is a path which loops back to its starting point. A graph with no cycles is **acyclic**. Directed acyclic graphs always have longest possible paths between any two given points. Graphs are represented with matrices as for min-length paths with 0’s on the diagonal. If an arrow from vertex \( i \) to vertex \( j \) has length 5, \( g(i,j) = 5 \). However, if there is no arrow from \( i \) to \( j \), set \( g(i,j) = -\infty \) rather than to \( \infty \). Let \( \infty = 10^6 \).

Given the matrix \( g \), we want to find \( \max_g(i,j) \), maximum distance between \( i \) and \( j \) = the total length (not number of steps) of a path from \( i \) to \( j \) of maximum total length.

Recall that when successively finding a minimum (classwork above), we start with \( \infty \) as the default. When finding a maximum (homework), we start with \(-\infty \) as the default.

Do not do both 23.1A(5) and 23.1B(8).

**HOMEWORK 23.1A(5)** maxpath.f95 See **classwork** minpath.f95 email: dale@math.hawaii.edu subject line: 190 h23.1a(5)

Suppose \( g \) is the matrix for a directed acyclic graph with \( n \) vertices and integer lengths on its edges. Write a subroutine \( \text{max_length}(g,\max_g,n) \) which calculates the matrix \( \max_g \) such that \( \max_g(i,j) = \) the maximum length of a path between vertices \( i \) and \( j \) of the graph \( g \).

!max_length.f95 subject line: 190 h23.1a(5)

```
subroutine max_length(g,max_g,n)
~implicit none
integer::g(n,n),max_g(n,n),double(n)
integer::i,j,n,k,nsteps,maxsteps
nsteps=1; max_g=g; maxsteps=n+1
do
!delete this line, complete the subroutine, quiz on this
enddo
endsubroutine
```

```
program test_max_length
integer::g(4,4)=1,max_g(4,4),n=4
integer, parameter::inf=10**6

!max_length.f95 subject line: 190 h23.1a(5)

subroutine max_length(a,n)
integer::a(n,n)
11 format(4x,20(2x,"(",i1,")"))
print 11, (i,i=1,n)
12 format(3x,"(",i1,"),")",200(i2,3x))
do i=1,n; print 12,i,a(i,1:n); enddo;print*
endsubroutine !Your answer's first row should be: 0 1 2 3
```

```
Job scheduling

A project requires completing several jobs, some are independent, some must precede others. Find the minimum time required to complete the project. Strangely, this is a longest-path problem. Since all jobs must be completed (not just those on the shortest path), the minimum time required to complete all jobs is the time required to complete all jobs on the path with the path longest total time. The nodes of the graph are labeled with the jobs. There is an edge $A \rightarrow B$ between jobs $A$ and $B$ iff $A$ immediately precedes $B$. This edge is labeled with the time required to do $A$ which is the minimum time between the start of $A$ and the start of $B$.

- Job 1 must be done first; then jobs 2, 3, 4 can be done concurrently; job 5 must be done after 2, 3, 4. We add a final node 6 to represent the finish of the project.

Job times: 1→2hrs, 2→4hrs, 3→3hrs, 4→1hr, 5→5hrs.

The number on all edges from a job node = the time required for the job = the time between the job’s start the next job.

Answer: The minimum project time
= the time required to complete all jobs
= the time to complete the longest path = 11.

Minimum drill press paths

A drill press is used to drill holes in the 8 marked locations on a circuit board with coordinates marked in centimeters. You must find a circuit which drills all holes with the least total distance (measured in millimeters) traveled by the drill head. In Traveling Salesman terminology, drilling a hole is visiting a city. Just having the drill head pass over a hole doesn’t count as a visit.
The 8 columns of matrix \( b = \begin{bmatrix} 0 & 2 & 2 & 3 & 1 & 2 & 0 & 3 \\ 3 & 3 & 2 & 2 & 1 & 1 & 0 & 0 \end{bmatrix} \) give the coordinates of the 8 holes. The \( x \) coordinates are the top row, the \( y \) coordinates are the second. Hole 1, (top left) has coordinates \([0,3]=b(:,1)\). Hole 8 has coordinates \([3,0]=b(:,8)\). The graph \( g \) for this problem has 8 nodes (one for each hole). The length of an edge between two holes is the geometric distance between the holes. For the edge between holes 1 and 8, \( g(1,8) = \sqrt{(0-3)^2 + (3-0)^2} \). The Fortran code for the distance in millimeters first converts to reals (in order to take the square root) then back to integers in millimeters \( \text{distance}(u,v) = \text{int}(10*\text{sqrt}(\text{real}(\text{sum}((u-v)^2)))) \). Hence, the length of the edge from hole 1 to 8 is \( g(1,8) = \text{distance}(b(:,1),b(:,8)) \).

**Homework 23.2(4)** mindrillpath.f95 Not on quiz.

email: dale@math.hawaii.edu subject line: 190 h23.2(4)

Use the `min_circuit` subroutine to find a circuit for the drill head with the shortest total geometric distance. The `min_circuit` subroutine finds the circuit, you have to define the graph \( g \) and the `distance` function (see above). Fill the pink ____.

```fortran
integer function distance(u,v)
integer :: u(2), v(2)
distance = _______________
endfunction

program test_mindrillpath
integer :: g(8,8)=1, path(9), circuit(9), b(2,8)
distance := inf=10**6, n=8, cir_len, distance
path(1)=1; npath=1
b(1,:)= (/0,2,2,3,1,2,0,3/)
b(2,:)= _______________
do i=1,8; do j=1,8
  g(i,j)= _______________
endo; enddo
call min_circuit(g,n,path,npath, circuit,cir_len)
call print_circuit
contains
subroutine print_circuit
If(cir_len==inf) then;
  print*,"No circuit possible."
else;
  print'(a,f5.1,a)',"Min length = ",real(cir_len)/10
  12 format(a,200(i2,3x))
  print 12," Circuit:",circuit(1:n+1)
  print*
endif
endsubroutine
endprogram ! Ans: min length = 12.8
```