Title: Lecture - Numerical Methods, PHYS 777

Speakers: Erik Schnetter, Dustin Lang

Collection/Series: Numerical Methods (Core), PHYS 777-, January 6 - February 5, 2025

Subject: Other

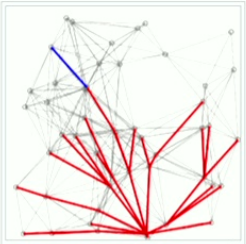
Date: January 17, 2025 - 11:30 AM

URL: https://pirsa.org/25010058

Pseudocode [edit]

In the following pseudocode, `dist` is an array that contains the current distances from the *source* to other vertices, i.e. `dist[u]` is the current distance from the source to the vertex *u*. The `prev` array contains pointers to previous-hop nodes on the shortest path from source to the given vertex (equivalently, it is the *next-hop* on the path *from* the given vertex *to* the source). The code `u ← vertex in Q with min dist[u]`, searches for the vertex *u* in the vertex set *Q* that has the least `dist[u]` value. `Graph.Edges(u, v)` returns the length of the edge joining (i.e. the distance between) the two neighbor-nodes *u* and *v*. The variable *alt* on line 14 is the length of the path from the *source* node to the neighbor node *v* if it were to go through *u*. If this path is shorter than the current shortest path recorded for *v*, then the distance of *v* is updated to *alt*.[7]

```
1  function Dijkstra(Graph, source):
2
3      for each vertex v in Graph.Vertices:
4          dist[v] ← INFINITY
5          prev[v] ← UNDEFINED
6          add v to Q
7      dist[source] ← 0
8
9      while Q is not empty:
10         u ← vertex in Q with minimum dist[u]
11         remove u from Q
12
13         for each neighbor v of u still in Q:
14             alt ← dist[u] + Graph.Edges(u, v)
15             if alt < dist[v]:
16                 dist[v] ← alt
17                 prev[v] ← u
18
19     return dist[], prev[]
```

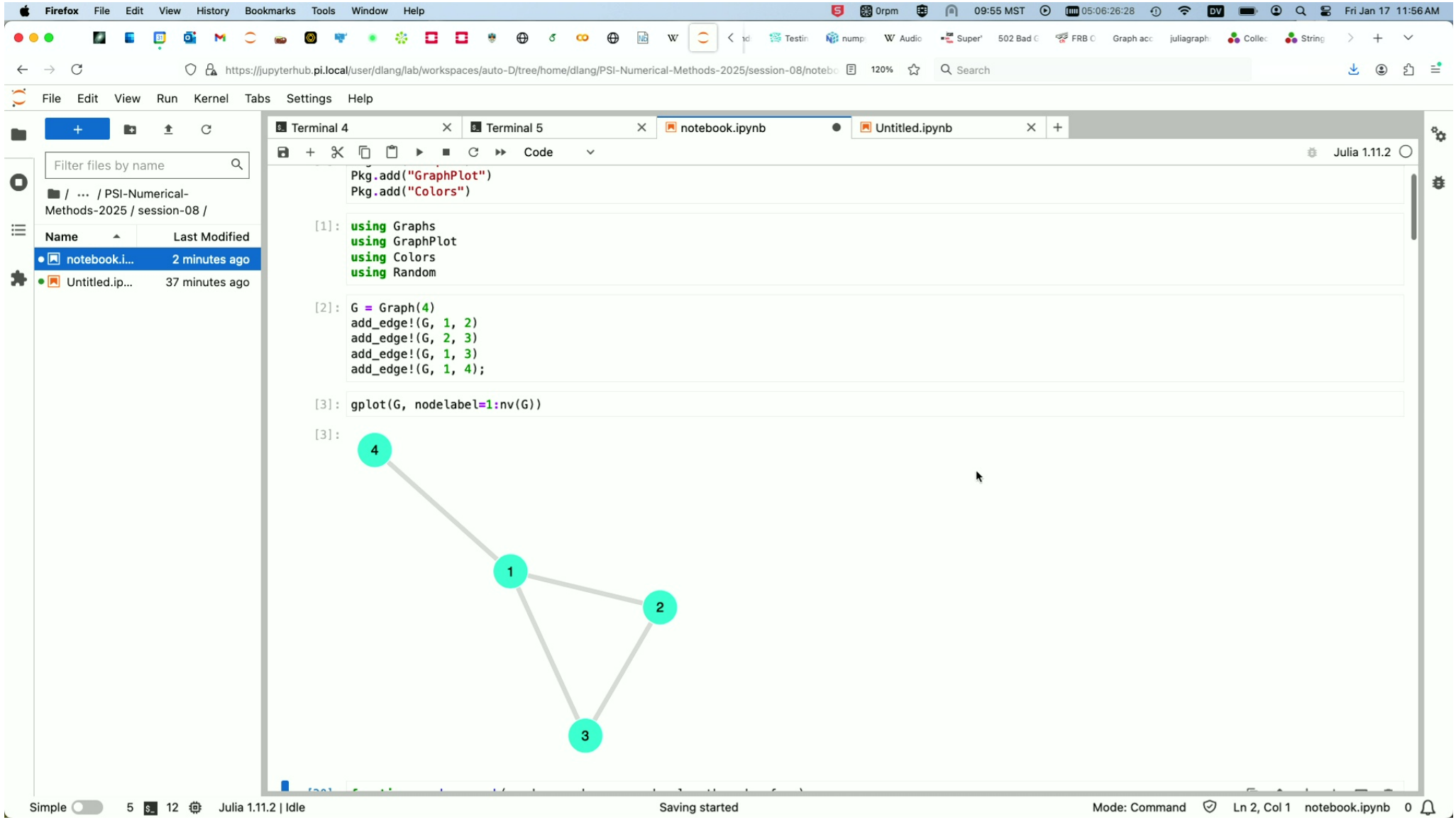


A demo of Dijkstra's algorithm based on Euclidean distance. Red lines are the shortest path covering, i.e., connecting *u* and prev[*u*]. Blue lines indicate where relaxing happens, i.e., connecting *v* with a node *u* in *Q*, which gives a shorter path from the source to *v*.

To find the shortest path between vertices *source* and *target*, the search terminates after line 10 if `u = target`. The shortest path from *source* to *target* can be obtained by reverse iteration:

```
1  S ← empty sequence
2  u ← target
3  if prev[u] is defined or u = source:          // Proceed if the vertex is reachable
4      while u is defined:                       // Construct the shortest path with a stack S
5          insert u at the beginning of S        // Push the vertex onto the stack
6          u ← prev[u]                           // Traverse from target to source
```

Now sequence *S* is the list of vertices constituting one of the shortest paths from *source* to *target*, or the empty sequence if no


Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
Pkg.add("GraphPlot")
Pkg.add("Colors")
[1]:
using Graphs
using GraphPlot
using Colors
using Random
[2]:
G = Graph(4)
add_edge!(G, 1, 2)
add_edge!(G, 2, 3)
add_edge!(G, 1, 3)
add_edge!(G, 1, 4);
[3]:
gplot(G, nodelabel=1:nv(G))
[3]:
4
1
2
3
Filter files by name
/ ... / PSI-Numerical-Methods-2025 / session-08 /
Name
Last Modified
notebook.i...
2 minutes ago
Untitled.ip...
37 minutes ago
Simple
Julia 1.11.2 | Idle
Saving started
Mode: Command
Ln 2, Col 1
notebook.ipynb

Firefox File Edit View History Bookmarks Tools Window Help

https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-D/tree/home/dlang/PSI-Numerical-Methods-2025/session-08/notebo

File Edit View Run Kernel Tabs Settings Help

Terminal 4 | Terminal 5 | notebook.ipynb | Untitled.ipynb

Filter files by name

/ ... / PSI-Numerical-Methods-2025 / session-08 /

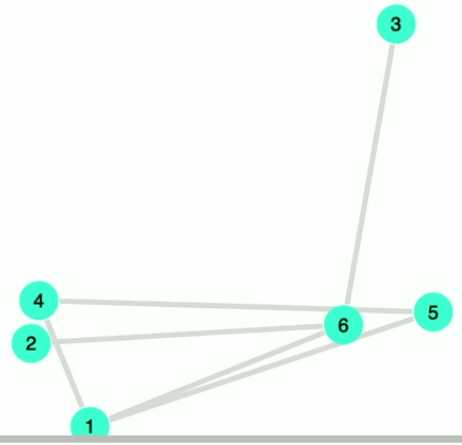
Name	Last Modified
notebook.i...	a minute ago
Untitled.ip...	38 minutes ago

Code — Julia 1.11.2

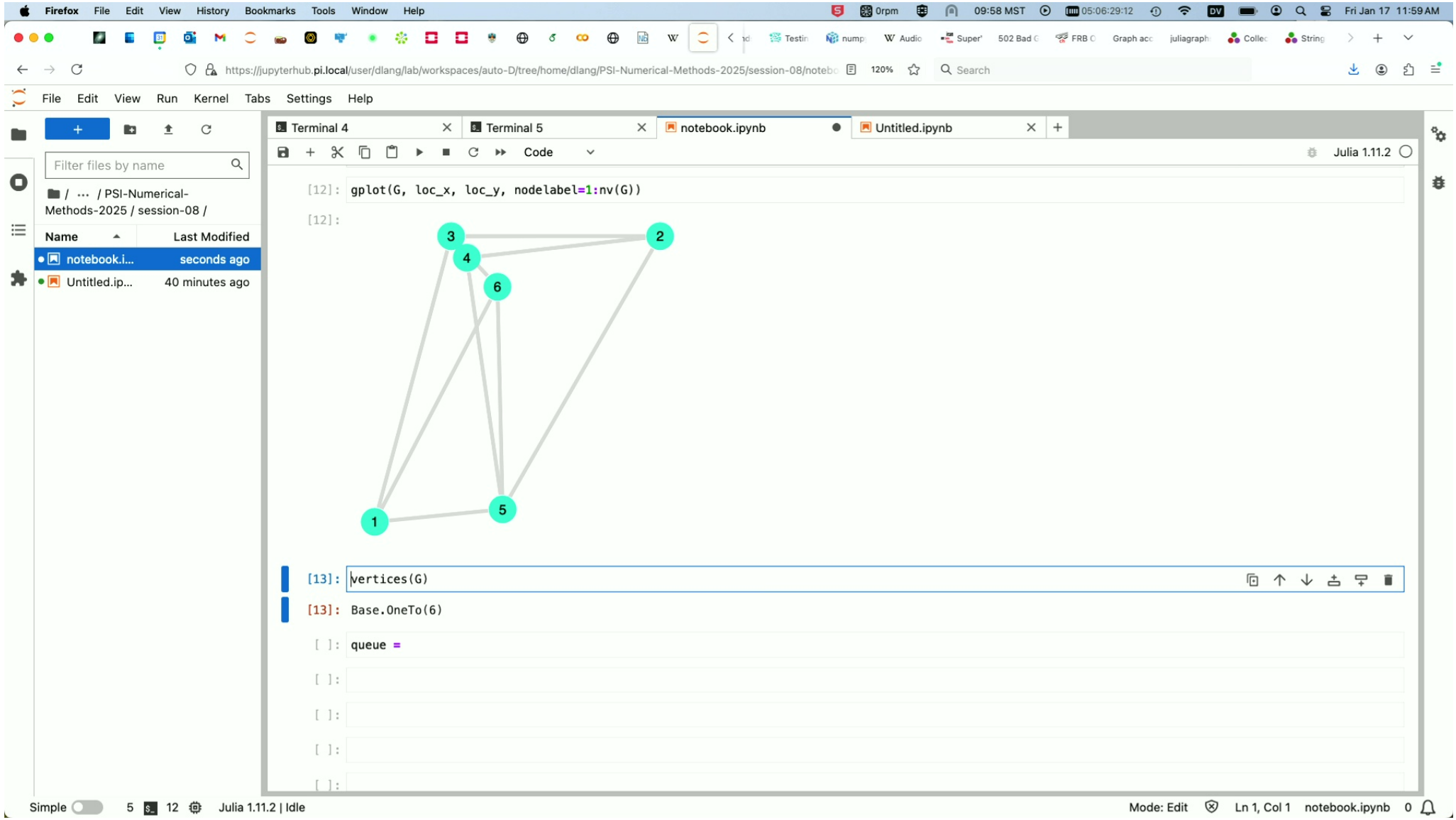
[5]:
```
seed = 45
n_nodes = 6
max_edge_length = 2
edge_frac = 0.5
source = 1
dest = 3

#seed = 100
#n_nodes = 15
#max_edge_length = 0.4
#edge_frac = 0.7
#source = 3
#dest = 12

Random.seed!(seed)
G, loc_x, loc_y, edge_labels, path_lengths = random_graph(seed, n_nodes, max_edge_length, edge_frac);
```

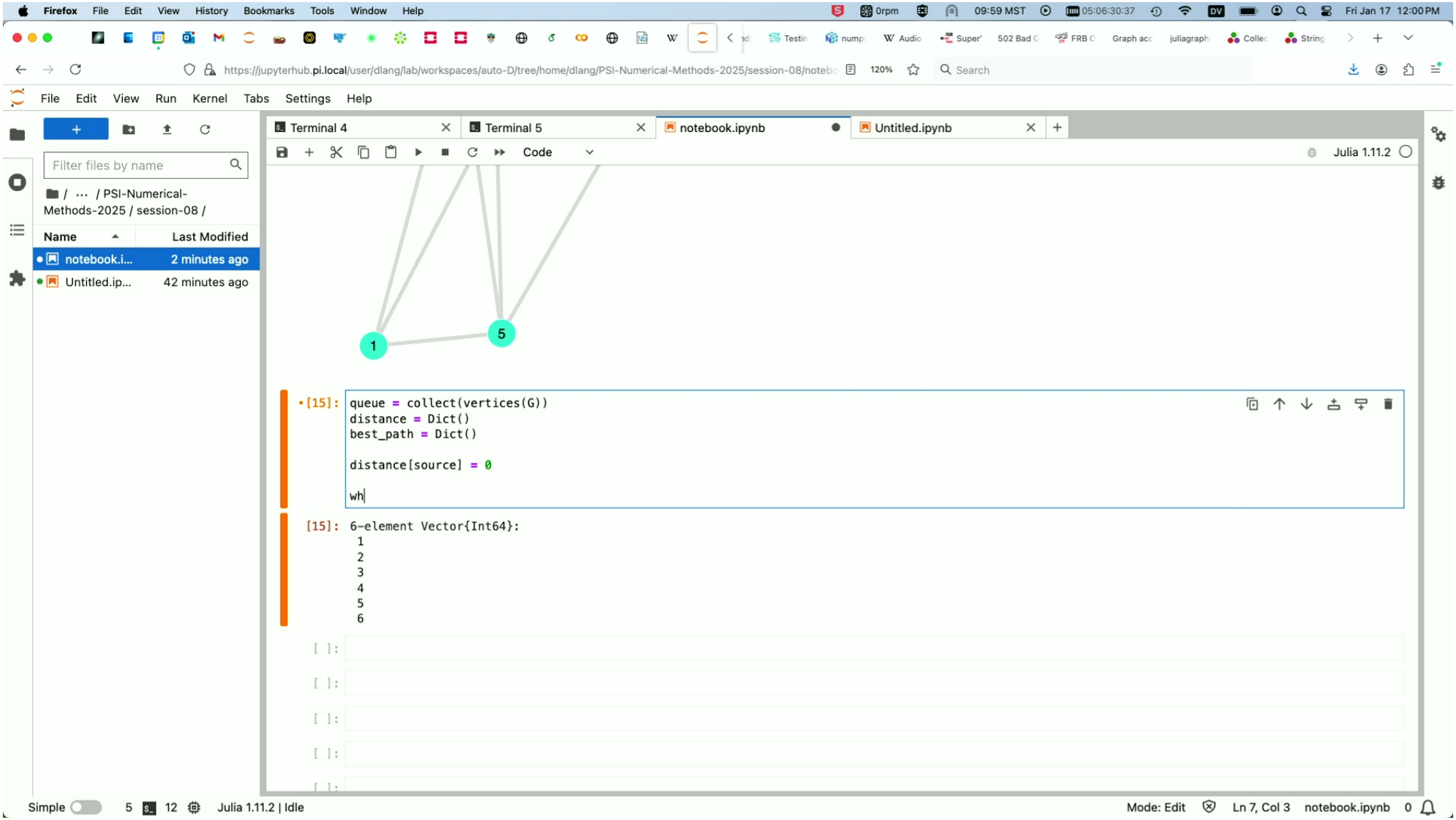
[6]:
```
gplot(G, loc_x, loc_y, nodelabel=1:nv(G))
```

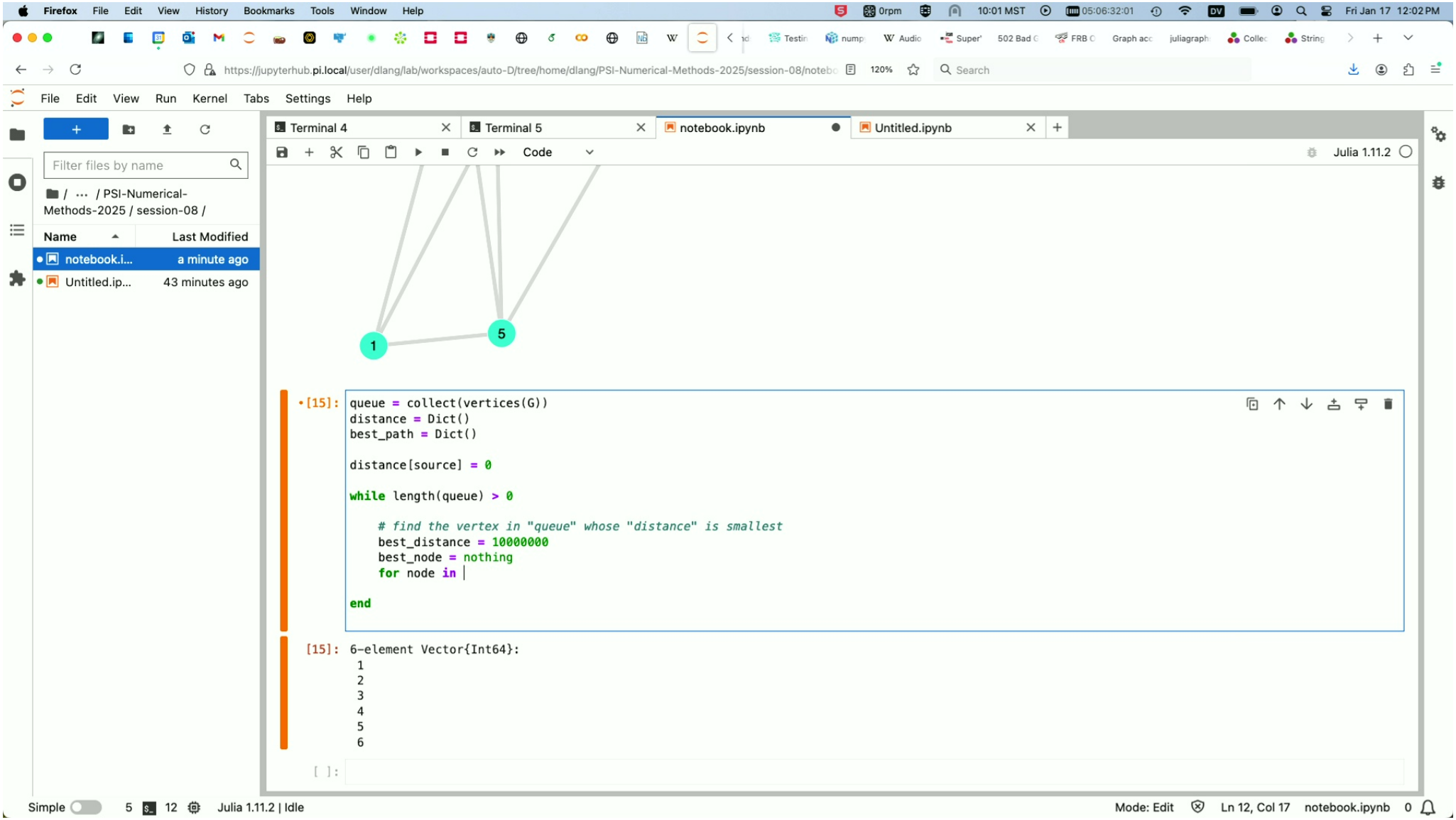
[6]:

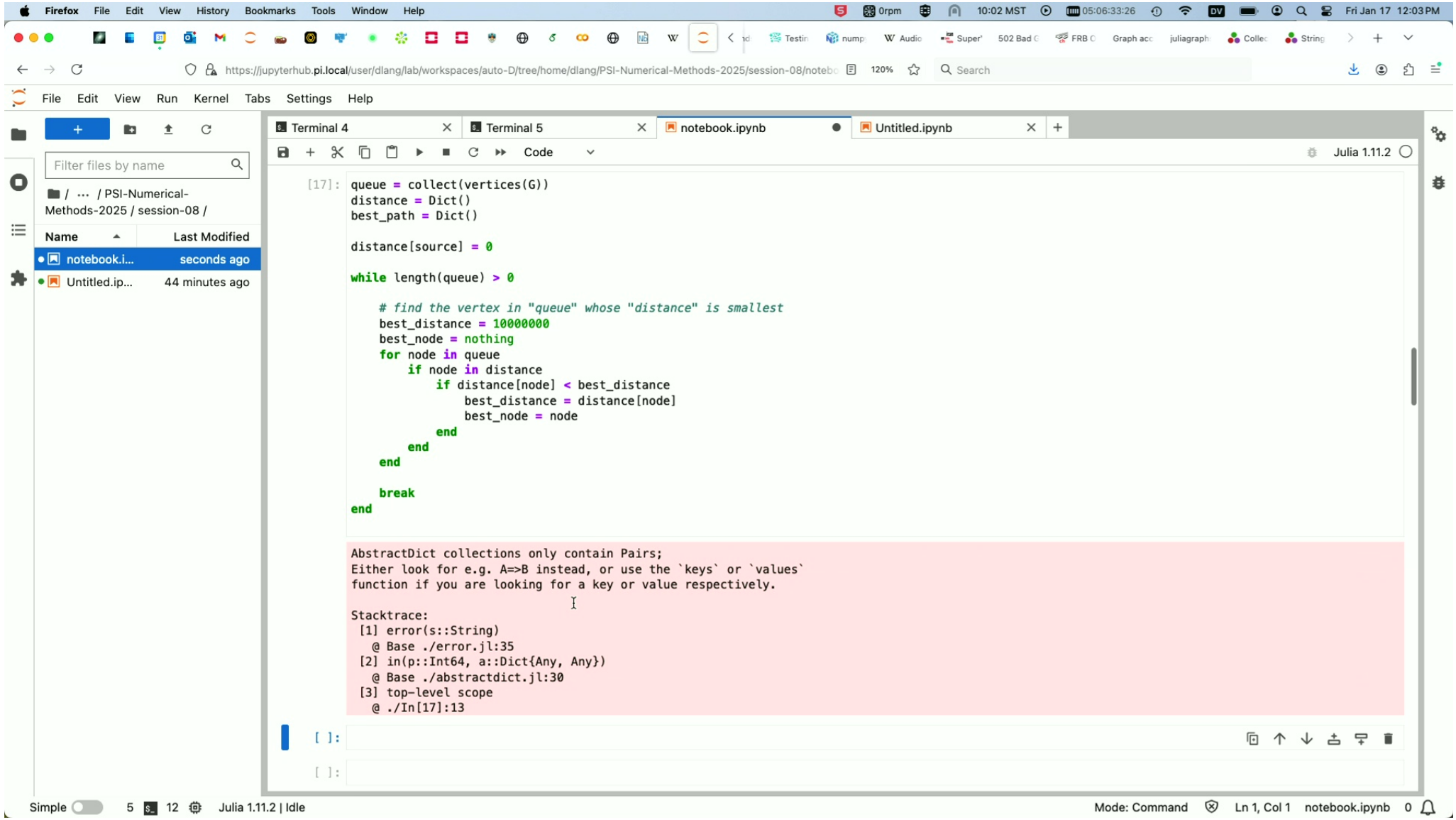


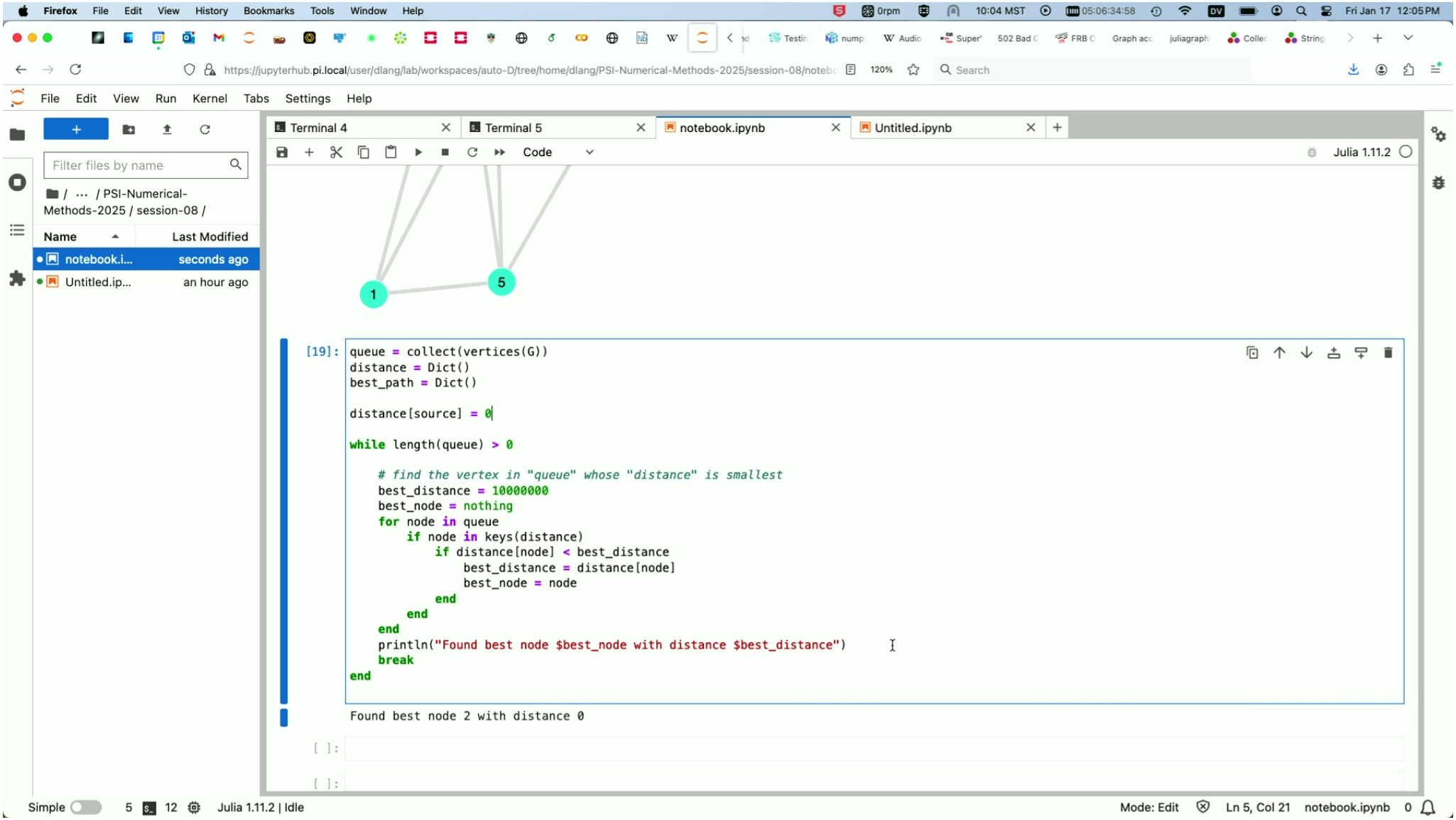
Simple 5 12 Julia 1.11.2 | Idle — Mode: Command — Ln 1, Col 1 — notebook.ipynb 0

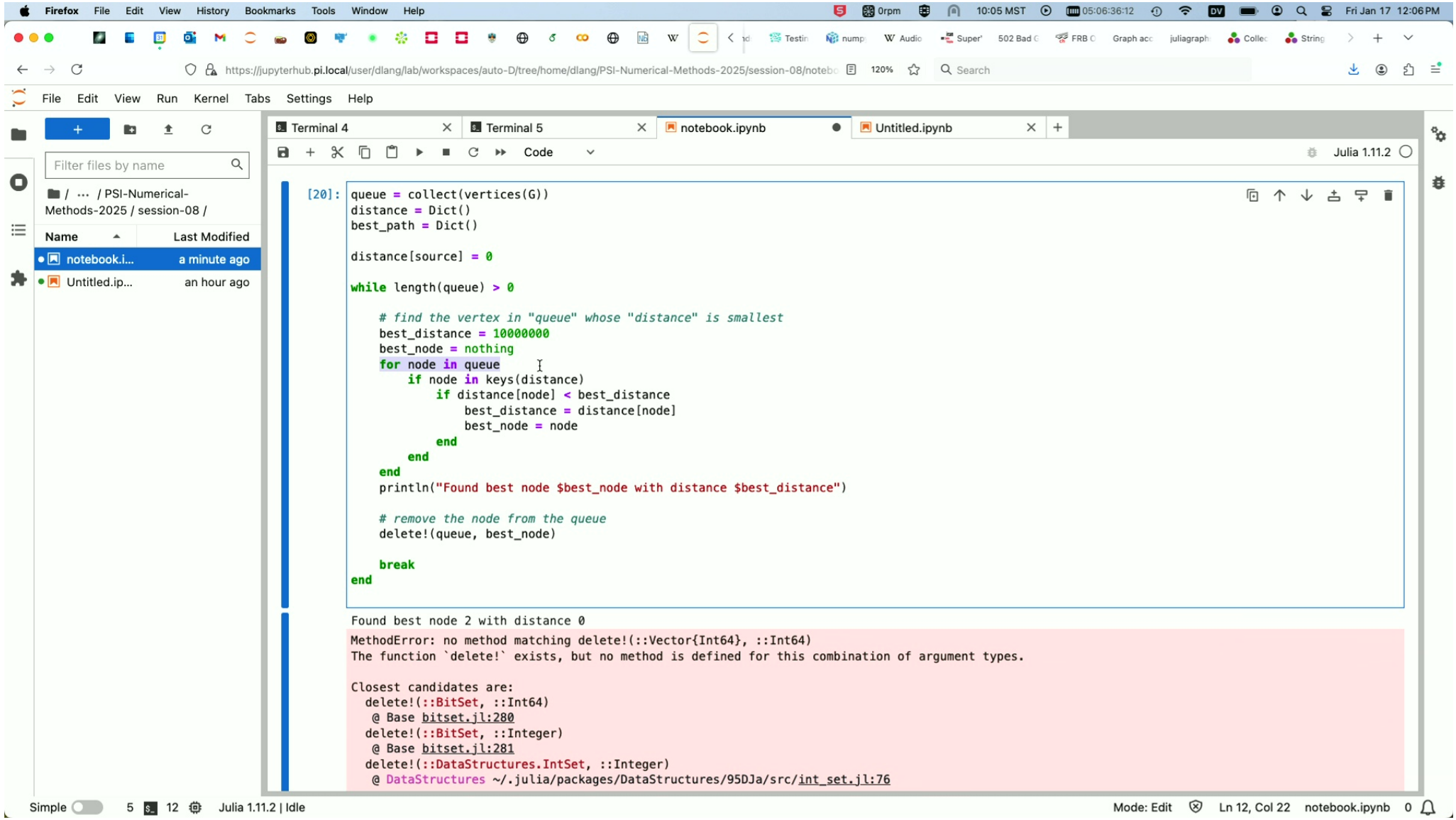

Firefox File Edit View History Bookmarks Tools Window Help
Fri Jan 17 11:59 AM
https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-D/tree/home/dlang/PSI-Numerical-Methods-2025/session-08/notebo
File Edit View Run Kernel Tabs Settings Help
Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
Filter files by name
/ ... / PSI-Numerical-Methods-2025 / session-08 /
Name
Last Modified
notebook.i... seconds ago
Untitled.ip... 40 minutes ago
[12]: gplot(G, loc_x, loc_y, nodelabel=1:nv(G))
[12]:
1
2
3
4
5
6
[13]: vertices(G)
[13]: Base.OneTo(6)
[ ]: queue =
Simple
Julia 1.11.2 | Idle
Mode: Edit
Ln 1, Col 1
notebook.ipynb

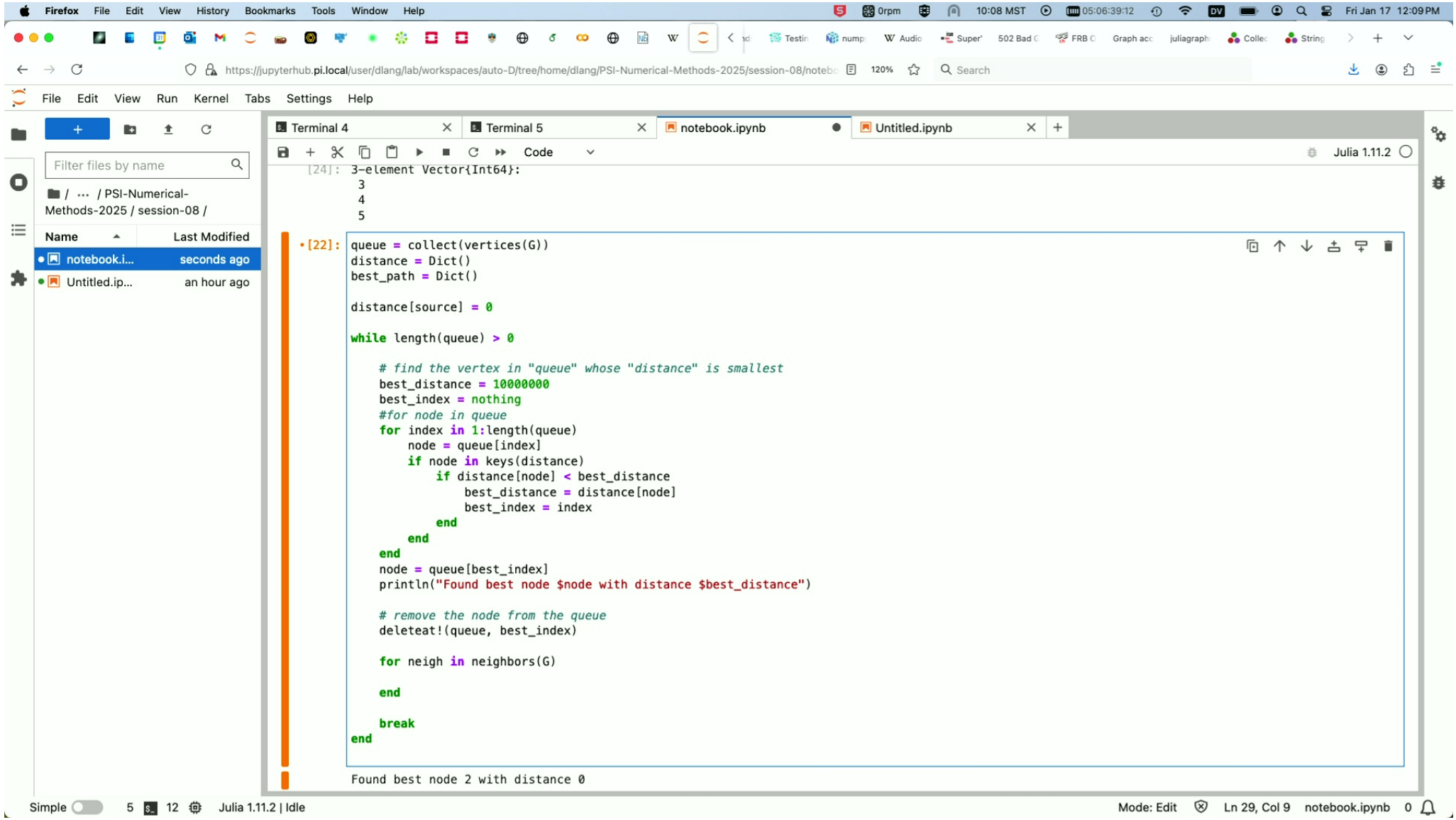

Firefox File Edit View History Bookmarks Tools Window Help
Fri Jan 17 12:00 PM
https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-D/tree/home/dlang/PSI-Numerical-Methods-2025/session-08/notebo
File Edit View Run Kernel Tabs Settings Help
Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
Filter files by name
/ ... / PSI-Numerical-Methods-2025 / session-08 /
Name
Last Modified
notebook.i...
2 minutes ago
Untitled.ip...
42 minutes ago
1
5
[15]:
queue = collect(vertices(G))
distance = Dict()
best_path = Dict()
distance[source] = 0
wh
[15]:
6-element Vector{Int64}:
1
2
3
4
5
6
Simple
5
12
Julia 1.11.2 | Idle
Mode: Edit
Ln 7, Col 3
notebook.ipynb
0

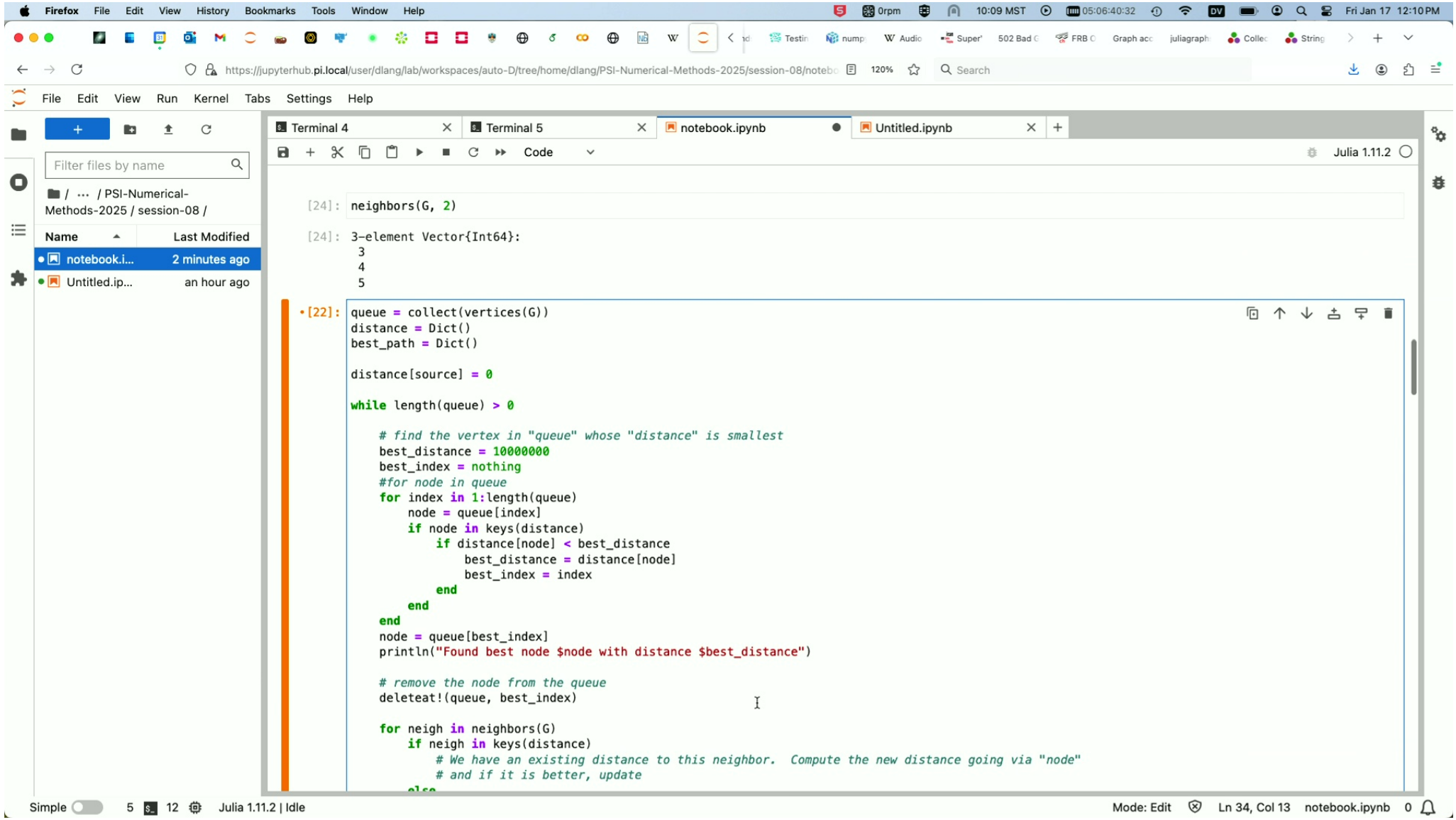

Firefox File Edit View History Bookmarks Tools Window Help
Fri Jan 17 12:02 PM
https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-D/tree/home/dlang/PSI-Numerical-Methods-2025/session-08/notebo
File Edit View Run Kernel Tabs Settings Help
Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
Filter files by name
/ ... / PSI-Numerical-Methods-2025 / session-08 /
Name
Last Modified
notebook.i... a minute ago
Untitled.ip... 43 minutes ago
1
5
[15]:
queue = collect(vertices(G))
distance = Dict()
best_path = Dict()
distance[source] = 0
while length(queue) > 0
# find the vertex in "queue" whose "distance" is smallest
best_distance = 10000000
best_node = nothing
for node in
end
[15]: 6-element Vector{Int64}:
1
2
3
4
5
6
[ ]:
Simple
5
12
Julia 1.11.2 | Idle
Mode: Edit
Ln 12, Col 17
notebook.ipynb
0

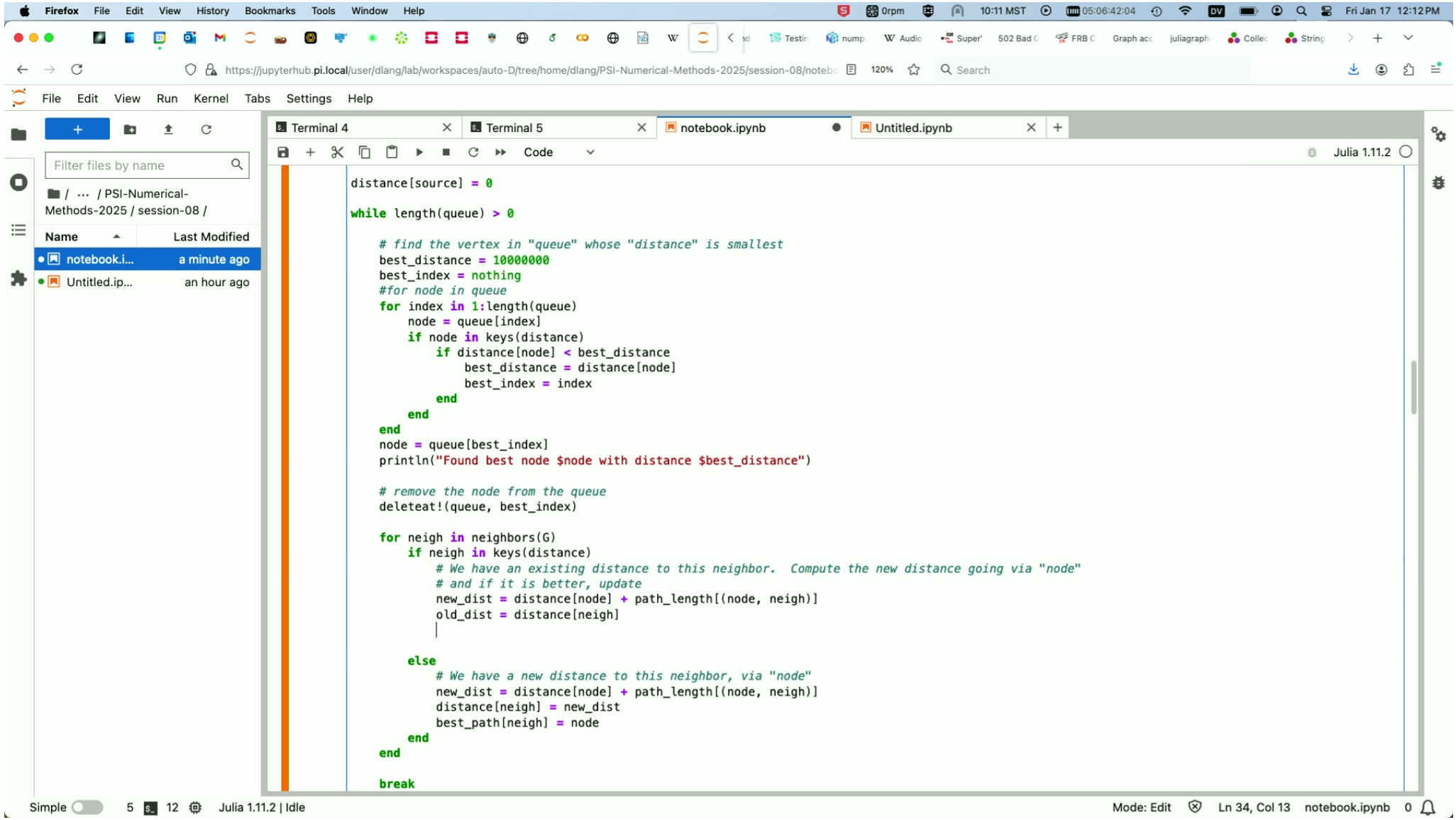

Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
[17]:
queue = collect(vertices(G))
distance = Dict()
best_path = Dict()
distance[source] = 0
while length(queue) > 0
# find the vertex in "queue" whose "distance" is smallest
best_distance = 10000000
best_node = nothing
for node in queue
if node in distance
if distance[node] < best_distance
best_distance = distance[node]
best_node = node
end
end
end
break
end
AbstractDict collections only contain Pairs;
Either look for e.g. A=>B instead, or use the `keys` or `values`
function if you are looking for a key or value respectively.
Stacktrace:
[1] error(s::String)
@ Base ./error.jl:35
[2] in(p::Int64, a::Dict{Any, Any})
@ Base ./abstractdict.jl:30
[3] top-level scope
@ ./In[17]:13
[ ]:
[ ]:
Filter files by name
/ ... / PSI-Numerical-Methods-2025 / session-08 /
Name
Last Modified
notebook.i...
seconds ago
Untitled.ip...
44 minutes ago
Simple
Julia 1.11.2 | Idle
Mode: Command
Ln 1, Col 1
notebook.ipynb

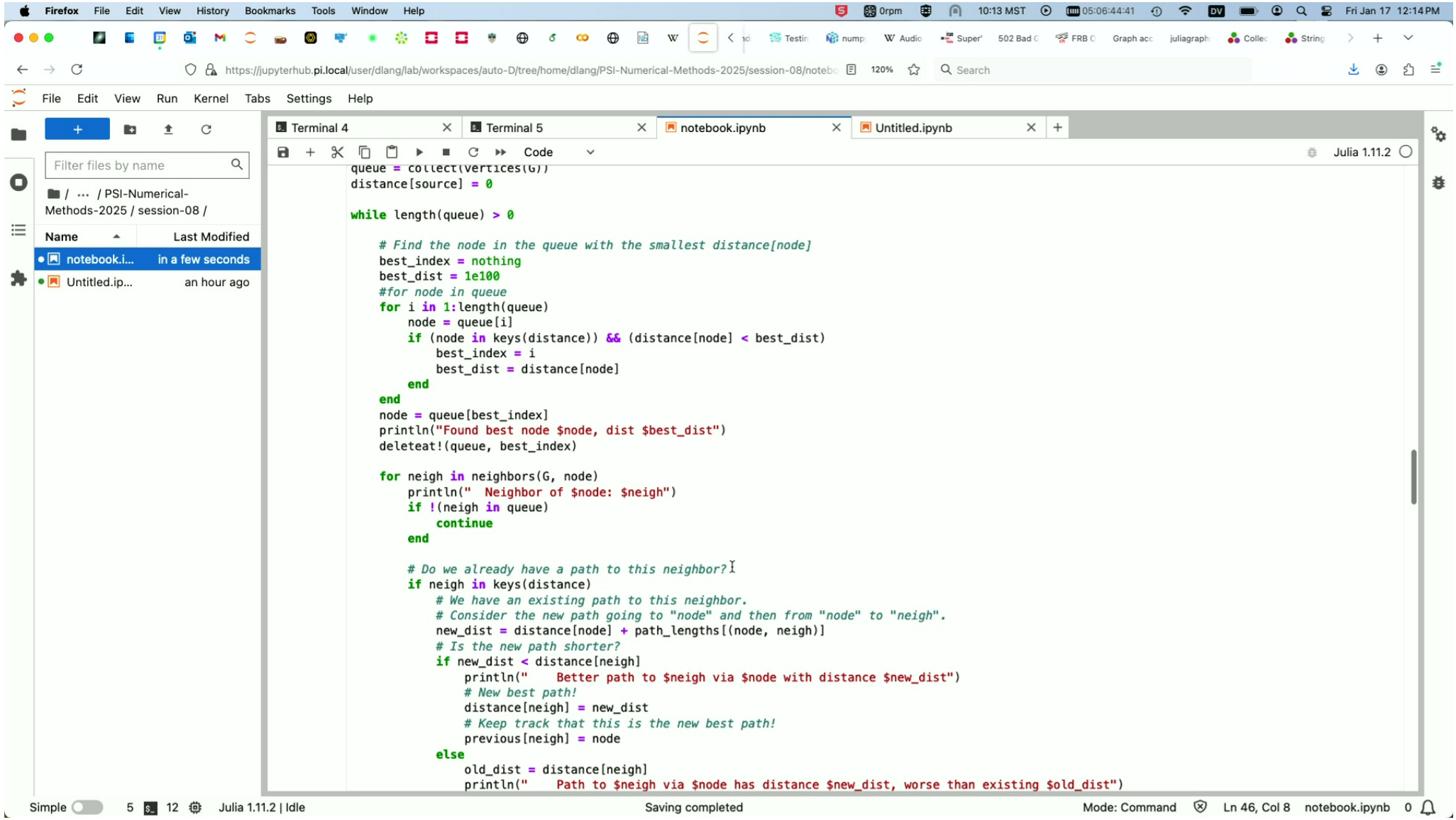

Firefox File Edit View History Bookmarks Tools Window Help
Fri Jan 17 12:05 PM
https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-D/tree/home/dlang/PSI-Numerical-Methods-2025/session-08/notebo
File Edit View Run Kernel Tabs Settings Help
Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
Filter files by name
/ ... / PSI-Numerical-Methods-2025 / session-08 /
Name
Last Modified
notebook.i... seconds ago
Untitled.ip... an hour ago
1
5
[19]:
queue = collect(vertices(G))
distance = Dict()
best_path = Dict()
distance[source] = 0
while length(queue) > 0
# find the vertex in "queue" whose "distance" is smallest
best_distance = 10000000
best_node = nothing
for node in queue
if node in keys(distance)
if distance[node] < best_distance
best_distance = distance[node]
best_node = node
end
end
end
println("Found best node $best_node with distance $best_distance")
break
end
Found best node 2 with distance 0
[ ]:
[ ]:
Simple
5
12
Julia 1.11.2 | Idle
Mode: Edit
Ln 5, Col 21
notebook.ipynb
0

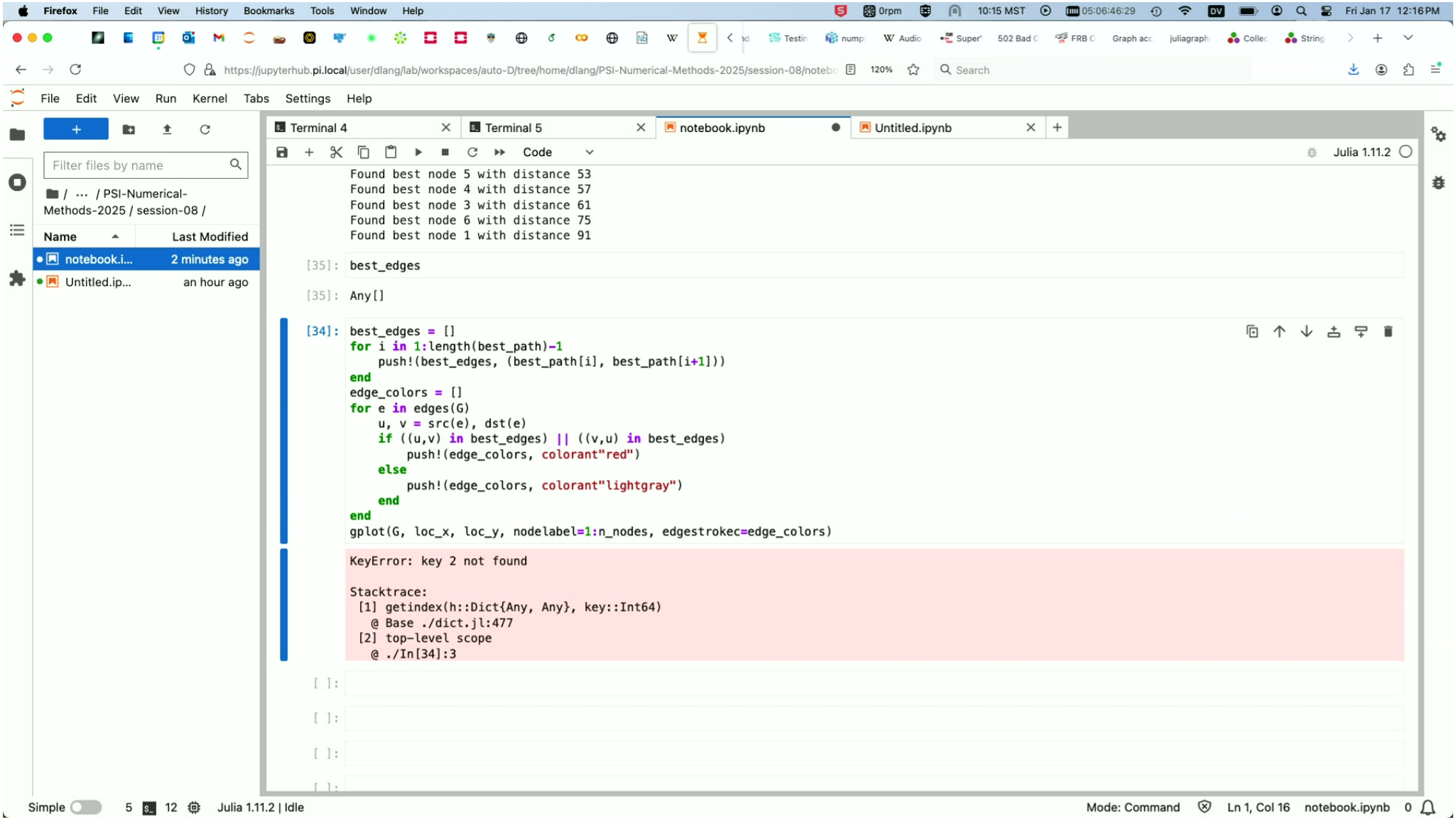

Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
[20]:
queue = collect(vertices(G))
distance = Dict()
best_path = Dict()
distance[source] = 0
while length(queue) > 0
# find the vertex in "queue" whose "distance" is smallest
best_distance = 10000000
best_node = nothing
for node in queue
if node in keys(distance)
if distance[node] < best_distance
best_distance = distance[node]
best_node = node
end
end
end
println("Found best node $best_node with distance $best_distance")
# remove the node from the queue
delete!(queue, best_node)
break
end
Found best node 2 with distance 0
MethodError: no method matching delete!(::Vector{Int64}, ::Int64)
The function `delete!` exists, but no method is defined for this combination of argument types.
Closest candidates are:
delete!(::BitSet, ::Int64)
@ Base bitset.jl:280
delete!(::BitSet, ::Integer)
@ Base bitset.jl:281
delete!(::DataStructures.IntSet, ::Integer)
@ DataStructures ~/.julia/packages/DataStructures/95DJa/src/int_set.jl:76
Simple
Julia 1.11.2 | Idle
Mode: Edit
Ln 12, Col 22
notebook.ipynb


Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
[24]: 3-element Vector{Int64}:
3
4
5
[22]:
queue = collect(vertices(G))
distance = Dict()
best_path = Dict()
distance[source] = 0
while length(queue) > 0
# find the vertex in "queue" whose "distance" is smallest
best_distance = 10000000
best_index = nothing
#for node in queue
for index in 1:length(queue)
node = queue[index]
if node in keys(distance)
if distance[node] < best_distance
best_distance = distance[node]
best_index = index
end
end
end
node = queue[best_index]
println("Found best node $node with distance $best_distance")
# remove the node from the queue
deleteat!(queue, best_index)
for neigh in neighbors(G)
end
break
end
Found best node 2 with distance 0
Julia 1.11.2 | Idle
Mode: Edit
Ln 29, Col 9
notebook.ipynb


Firefox File Edit View History Bookmarks Tools Window Help
Fri Jan 17 12:10 PM
https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-D/tree/home/dlang/PSI-Numerical-Methods-2025/session-08/notebo
File Edit View Run Kernel Tabs Settings Help
Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
Filter files by name
/ ... / PSI-Numerical-Methods-2025 / session-08 /
Name
Last Modified
notebook.i... 2 minutes ago
Untitled.ip... an hour ago
[24]: neighbors(G, 2)
[24]: 3-element Vector{Int64}:
3
4
5
[22]:
queue = collect(vertices(G))
distance = Dict()
best_path = Dict()
distance[source] = 0
while length(queue) > 0
# find the vertex in "queue" whose "distance" is smallest
best_distance = 10000000
best_index = nothing
#for node in queue
for index in 1:length(queue)
node = queue[index]
if node in keys(distance)
if distance[node] < best_distance
best_distance = distance[node]
best_index = index
end
end
end
node = queue[best_index]
println("Found best node $node with distance $best_distance")
# remove the node from the queue
deleteat!(queue, best_index)
for neigh in neighbors(G)
if neigh in keys(distance)
We have an existing distance to this neighbor. Compute the new distance going via "node"
and if it is better, update
Simple 5 12 Julia 1.11.2 | Idle
Mode: Edit Ln 34, Col 13 notebook.ipynb 0


Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
distance[source] = 0
while length(queue) > 0
# find the vertex in "queue" whose "distance" is smallest
best_distance = 10000000
best_index = nothing
#for node in queue
for index in 1:length(queue)
node = queue[index]
if node in keys(distance)
if distance[node] < best_distance
best_distance = distance[node]
best_index = index
end
end
end
node = queue[best_index]
println("Found best node $node with distance $best_distance")
# remove the node from the queue
deleteat!(queue, best_index)
for neigh in neighbors(G)
if neigh in keys(distance)
# We have an existing distance to this neighbor. Compute the new distance going via "node"
# and if it is better, update
new_dist = distance[node] + path_length[(node, neigh)]
old_dist = distance[neigh]
else
# We have a new distance to this neighbor, via "node"
new_dist = distance[node] + path_length[(node, neigh)]
distance[neigh] = new_dist
best_path[neigh] = node
end
end
break
Simple
Julia 1.11.2 | Idle
Mode: Edit
Ln 34, Col 13
notebook.ipynb

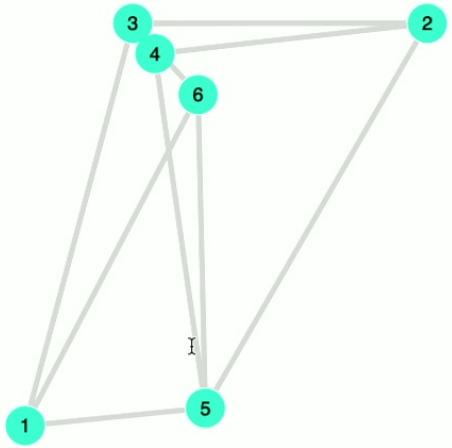

Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
queue = collect(vertices(G))
distance[source] = 0
while length(queue) > 0
# Find the node in the queue with the smallest distance[node]
best_index = nothing
best_dist = 1e100
#for node in queue
for i in 1:length(queue)
node = queue[i]
if (node in keys(distance)) && (distance[node] < best_dist)
best_index = i
best_dist = distance[node]
end
end
node = queue[best_index]
println("Found best node $node, dist $best_dist")
deleteat!(queue, best_index)
for neigh in neighbors(G, node)
println(" Neighbor of $node: $neigh")
if !(neigh in queue)
continue
end
Do we already have a path to this neighbor?
if neigh in keys(distance)
# We have an existing path to this neighbor.
# Consider the new path going to "node" and then from "node" to "neigh".
new_dist = distance[node] + path_lengths[(node, neigh)]
# Is the new path shorter?
if new_dist < distance[neigh]
println("   Better path to $neigh via $node with distance $new_dist")
# New best path!
distance[neigh] = new_dist
# Keep track that this is the new best path!
previous[neigh] = node
else
old_dist = distance[neigh]
println("   Path to $neigh via $node has distance $new_dist, worse than existing $old_dist")
Filter files by name
/ ... / PSI-Numerical-Methods-2025 / session-08 /
Name
Last Modified
notebook.i...
in a few seconds
Untitled.ip...
an hour ago
Simple
Julia 1.11.2 | Idle
Saving completed
Mode: Command
Ln 46, Col 8
notebook.ipynb


Found best node 5 with distance 53
Found best node 4 with distance 57
Found best node 3 with distance 61
Found best node 6 with distance 75
Found best node 1 with distance 91
[35]: best_edges
[35]: Any[]
[34]:
best_edges = []
for i in 1:length(best_path)-1
push!(best_edges, (best_path[i], best_path[i+1]))
end
edge_colors = []
for e in edges(G)
u, v = src(e), dst(e)
if ((u,v) in best_edges) || ((v,u) in best_edges)
push!(edge_colors, colorant"red")
else
push!(edge_colors, colorant"lightgray")
end
end
gplot(G, loc_x, loc_y, nodelabel=1:n_nodes, edgestrokec=edge_colors)
KeyError: key 2 not found
Stacktrace:
[1] getindex(h::Dict{Any, Any}, key::Int64)
@ Base ./dict.jl:477
[2] top-level scope
@ ./In[34]:3

```
Random.seed!(seed)
G, loc_x, loc_y, edge_labels, path_lengths = random_graph(seed, n_nodes, max_edge_length, edge_frac);
```

[32]:
```
gplot(G, loc_x, loc_y, nodelabel=1:nv(G))
```

[32]:

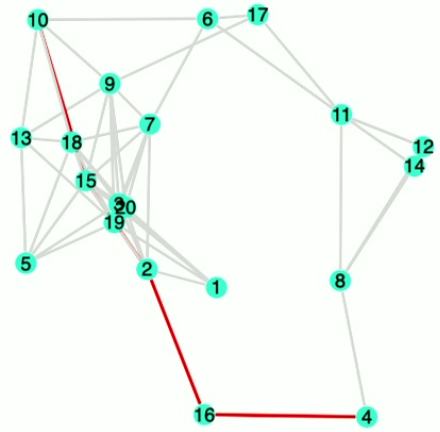


[33]:
```
queue = collect(vertices(G))
distance = Dict()
best_path = Dict()

distance[source] = 0

while length(queue) > 0

    # find the vertex in "queue" whose "distance" is smallest
    best_distance = 10000000
    best_index = nothing
    #for node in queue
```

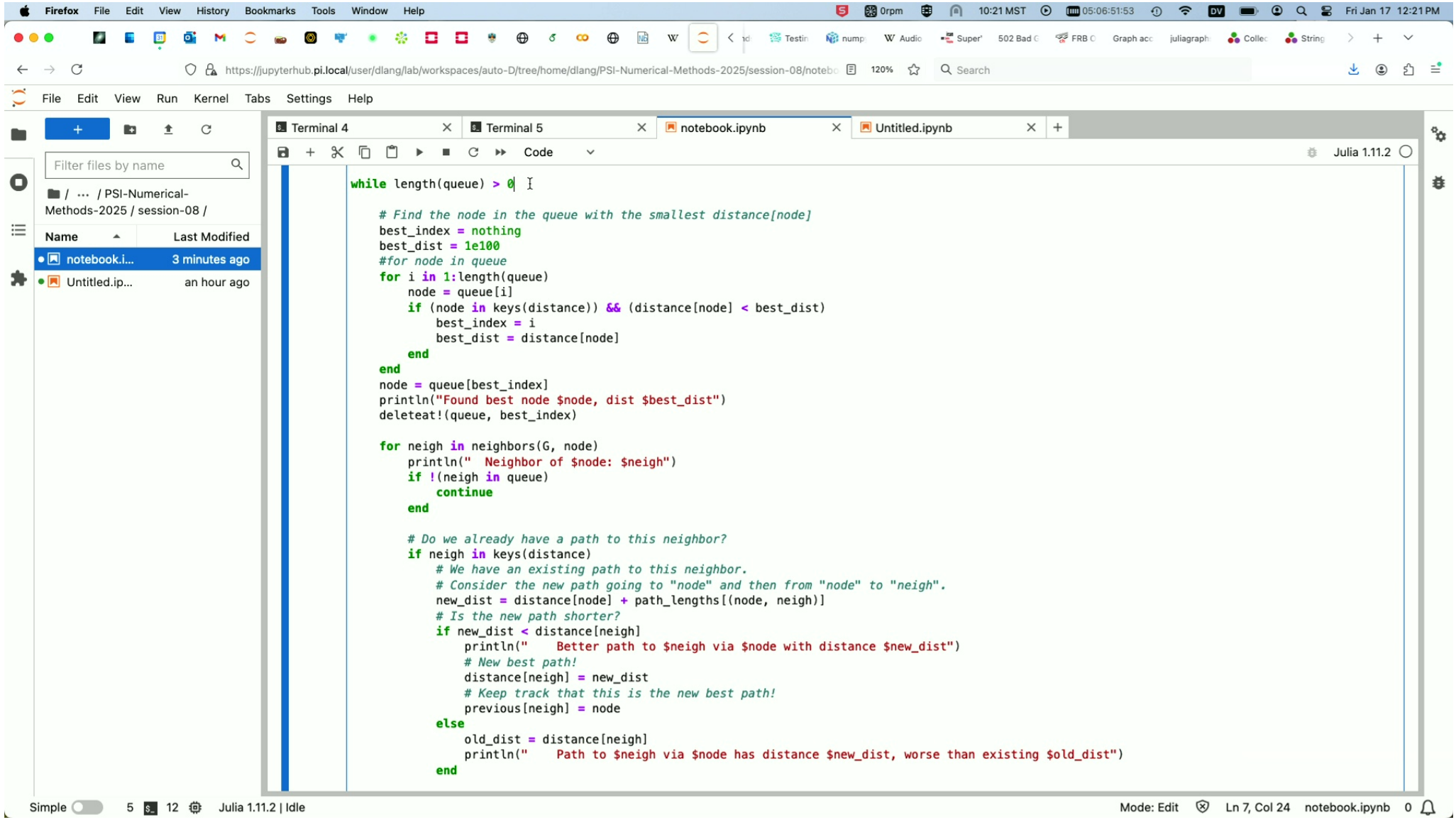
```
    push!(best_edges, (best_path[i], best_path[i+1]))
end
edge_colors = []
for e in edges(G)
    u, v = src(e), dst(e)
    if ((u,v) in best_edges) || ((v,u) in best_edges)
        push!(edge_colors, colorant"red")
    else
        push!(edge_colors, colorant"lightgray")
    end
end
```

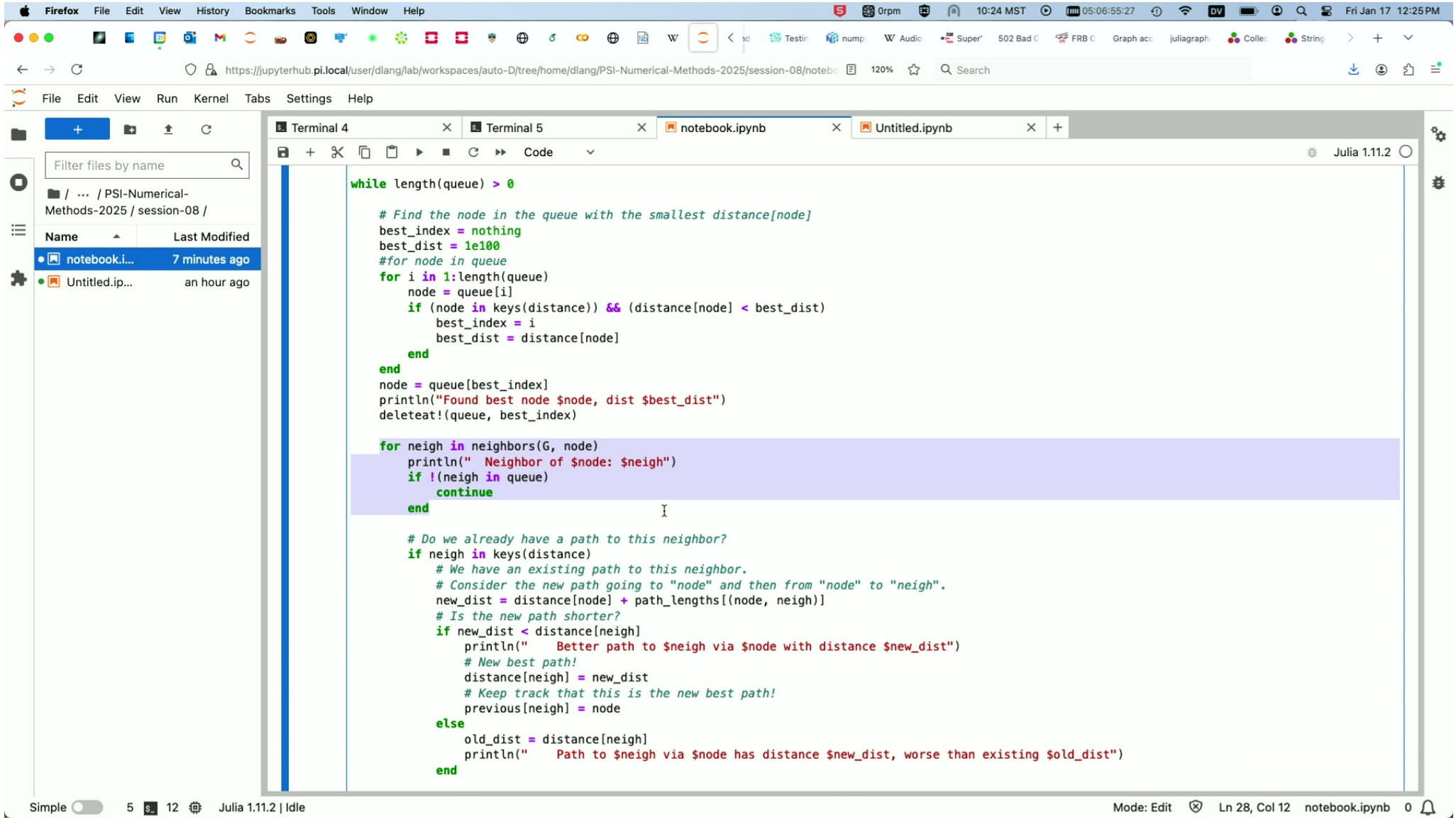
[50]:
```
gplot(G, loc_x, loc_y, nodelabel=1:n_nodes, edgestrokec=edge_colors)
```

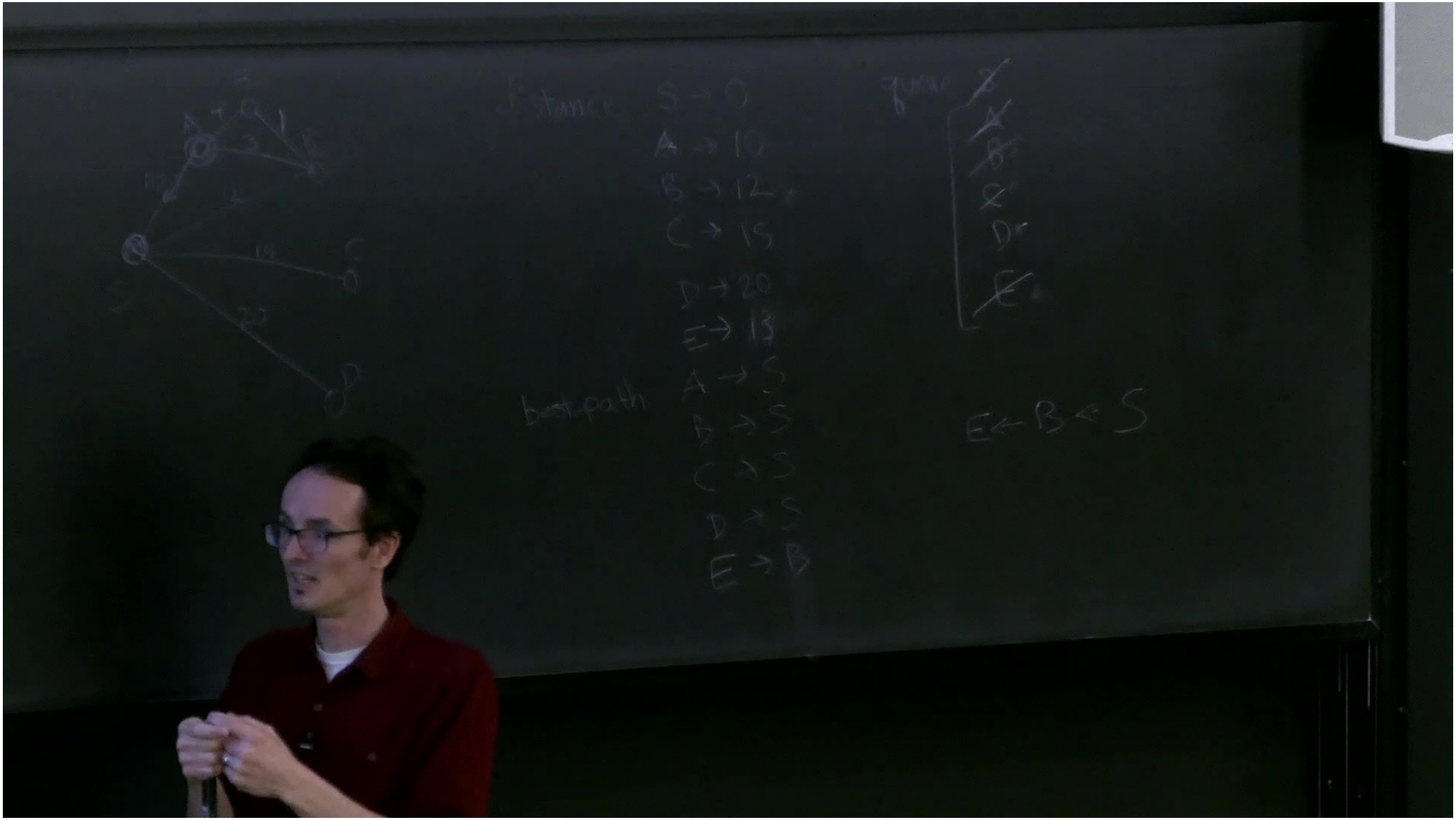
[50]:



[41]:
```
best_edges = []
for k in keys(previous)
    push!(best_edges, (k, previous[k]))
```


Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
while length(queue) > 0
# Find the node in the queue with the smallest distance[node]
best_index = nothing
best_dist = 1e100
#for node in queue
for i in 1:length(queue)
node = queue[i]
if (node in keys(distance)) && (distance[node] < best_dist)
best_index = i
best_dist = distance[node]
end
end
node = queue[best_index]
println("Found best node $node, dist $best_dist")
deleteat!(queue, best_index)
for neigh in neighbors(G, node)
println(" Neighbor of $node: $neigh")
if !(neigh in queue)
continue
end
Do we already have a path to this neighbor?
if neigh in keys(distance)
# We have an existing path to this neighbor.
# Consider the new path going to "node" and then from "node" to "neigh".
new_dist = distance[node] + path_lengths[(node, neigh)]
# Is the new path shorter?
if new_dist < distance[neigh]
println(" Better path to $neigh via $node with distance $new_dist")
# New best path!
distance[neigh] = new_dist
# Keep track that this is the new best path!
previous[neigh] = node
else
old_dist = distance[neigh]
println(" Path to $neigh via $node has distance $new_dist, worse than existing $old_dist")
end
Filter files by name
Name
Last Modified
notebook.i...
3 minutes ago
Untitled.ip...
an hour ago
Simple
Julia 1.11.2 | Idle
Mode: Edit
Ln 7, Col 24
notebook.ipynb


Firefox File Edit View History Bookmarks Tools Window Help
Fri Jan 17 12:25 PM
https://jupyterhub.pi.local/user/dlang/lab/workspaces/auto-D/tree/home/dlang/PSI-Numerical-Methods-2025/session-08/notebo
File Edit View Run Kernel Tabs Settings Help
Terminal 4
Terminal 5
notebook.ipynb
Untitled.ipynb
Code
Julia 1.11.2
Filter files by name
/ ... / PSI-Numerical-Methods-2025 / session-08 /
Name
Last Modified
notebook.i... 7 minutes ago
Untitled.ip... an hour ago
while length(queue) > 0
# Find the node in the queue with the smallest distance[node]
best_index = nothing
best_dist = 1e100
#for node in queue
for i in 1:length(queue)
node = queue[i]
if (node in keys(distance)) && (distance[node] < best_dist)
best_index = i
best_dist = distance[node]
end
end
node = queue[best_index]
println("Found best node $node, dist $best_dist")
deleteat!(queue, best_index)
for neigh in neighbors(G, node)
println(" Neighbor of $node: $neigh")
if !(neigh in queue)
continue
end
Do we already have a path to this neighbor?
if neigh in keys(distance)
# We have an existing path to this neighbor.
# Consider the new path going to "node" and then from "node" to "neigh".
new_dist = distance[node] + path_lengths[(node, neigh)]
# Is the new path shorter?
if new_dist < distance[neigh]
println(" Better path to $neigh via $node with distance $new_dist")
# New best path!
distance[neigh] = new_dist
# Keep track that this is the new best path!
previous[neigh] = node
else
old_dist = distance[neigh]
println(" Path to $neigh via $node has distance $new_dist, worse than existing $old_dist")
end
Simple
5 12
Julia 1.11.2 | Idle
Mode: Edit
Ln 28, Col 12
notebook.ipynb
0


distance
S → 0
A → 10
B → 12
C → 15
D → 20
E → 13
best path
A → S
B → S
C → S
D → S
E → B
queue
E ← B ← S