Problem 1:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i/j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 5 | 17 | 19 | 36 | 48 | 64 | 70 | 81 |
| 2 |  | 7 | 9 | 25 | 33 | 49 | 55 | 66 |
| 3 |  |  | 1 | 10 | 18 | 34 | 40 | 49 |
| 4 |  |  |  | 8 | 16 | 32 | 38 | 46 |
| 5 |  |  |  |  | 4 | 14 | 18 | 26 |
| 6 |  |  |  |  |  | 6 | 10 | 18 |
| 7 |  |  |  |  |  |  | 2 | 7 |
| 8 |  |  |  |  |  |  |  | 3 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i/j | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 1 | 2 | 2 | 2 | 2 | 4 | 4 | 4 |
| 2 |  | 2 | 2 | 4 | 4 | 4 | 4 | 4 |
| 3 |  |  | 3 | 4 | 4 | 4 | 4 | 6 |
| 4 |  |  |  | 4 | 4 | 4 | 4 | 6 |
| 5 |  |  |  |  | 5 | 6 | 6 | 6 |
| 6 |  |  |  |  |  | 6 | 6 | 6 |
| 7 |  |  |  |  |  |  | 7 | 8 |
| 8 |  |  |  |  |  |  |  | 8 |

A close up of a mans face

Description automatically generated

Problem 2

Problem 3

* 1. e = (ui,j, vi,j+1) => j ≤ n - 2,   
     e = (ui,j, vi+1, j) => ( i = 2k , j = n-1 ) OR ( i = 2k+1, j = 0) k = 0…n .
  2. A screenshot of a cell phone

     Description automatically generated

1. 1. e = (ui,j, vi+1,j) => i ≤ n – 2  
      e = (ui,j, vi, j+1) => i = 0, j < n – 2
   2. A picture containing text

      Description automatically generated

A screenshot of a cell phone

Description automatically generated

1. 1. DFS:   
      e = (ui,j, vi+1,j) => i < n -2   
      e = (ui,j, vi,j+1) => ( i = 0, j = 2k+1 ) OR ( i = n-1, j = 2k ) k = 0…n .  
        
      BFS:  
      e = (ui,j, vi,j+1) => i ≤ n – 2, j = 0  
      e = (ui,j, vi+1, j) => j < n – 2
   2. A close up of text on a white background

      Description automatically generated

Problem 4

**CycleDetection(G, v):**

for each vertex u ∈ V do

u.color ⃪ white

u.p ⃪ nil

for each vertex u ∈ V do

if u.color = white then DFS-Visit(u, v)

return “No Cycle”

**DFS-Visit(u, vertex):**

u.color ⃪ gray

for each v ∈ Adj[u] do

if v.color = white then

v.p ⃪ u

DFS-Visit(v)

else if v ≠ u.p then

while u ≠ v do

Cycle[] ⃪ u

u ⃪ u.p

Cycle[] ⃪ v

u.color ⃪ black

if Cycle[] contain vertex

output “Cycle found:”

for each u ∈ Cycle[]

output u

return

The algorithm mentioned in class is O(|V|) since there is early termination, the algorithm will end when the first cycle has been found. If we are going to found all the cycle until the target vertex inside the cycle, the run time will just like DFS algorithm: O(|E| + |V|).

Problem 5

1. If Edge e = (u, v) does not belong to the MST, then there must be an Edge e’ = (u’, v) weight less than e = (u, v). In term of using Kruskal’s Algorithm, all the edges have been gone through according to their weight in ascending order. A cheaper edge to node v (i.e. Edge e’) will always been chosen earlier. And the more expense edge will never be choosing afterward (i.e. Edge e).  
   Or in other way, if Edge e = (u, v) is not in the MST, adding it to the MST must form a cycle. Since there must be another path travel from u to v. By removing the heaviest edge in the cycle can produce an MST. Let’s assume that the original MST is correct, if we remove any edge other than Edge e, the cost of the MST will increase. This mean all other edges in the cycle are cheaper than Edge e.
2. For each edge in E do

if weight(edge) > weight(e) then

remove edge from E

For each edge in E do

connect edge to Tree

if Tree is connected then

e is not in the MST

else

e is in the MST