CS 313
- Kangmei Yang
Shortest Path

For Unweighted graph, the shortest path from s to t, is the minimum edges from s to t.

- Use breadth-first search from s.
- It stores the weight from s, the first time we visit t, is the shortest path from s to t.

SP from v1 to v3

SP(v1): enqueue v1
dequeue, adj[v1]:v2
wt[v2]=wt[v1]+1, enqueue v4
adj[v1]:v4
wt[v4]=wt[v1]+1, enqueue
dequeue, adj[v2]:v3
For Weighted graph, the shortest path from s to t, is the minimum total sum of weights on the path from s to t.
- Use similar idea as breadth-first search from s.

Greedy Algorithm
- Solve a task in stages
- Do what seems to be the best thing at each stage
- Not always optimal

Sample:
Changes:
- Find quarters, dimes, nickels, pennies
Always give minimum number of coins

What if there’s a 7 cents coin?
- For 14 cents, using a dime and 4 pennies are not optimal
Dijkstra’s algorithm

- It’s an example of greedy algorithm
- May use this method to find shortest-path for weighted graph.
- It proceeds in stages like unweighted shortest-path algorithm
- Keeps all information as unweighted shortest-path and as breadth-first search.

-Find SP from s to every other vertex

Sample: Shortest Path v1 to v2

SP(v1):
visit[v1]=T, adj[v1]: wt[v2], wt[v4]
wt[v3]
visit[v3]=T
Dijkstra’s Algorithm

States

Parent

<table>
<thead>
<tr>
<th>States</th>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>0</td>
</tr>
<tr>
<td>c21</td>
<td>c1</td>
</tr>
<tr>
<td>c22</td>
<td>c3</td>
</tr>
<tr>
<td>c23</td>
<td>c3</td>
</tr>
<tr>
<td>c31</td>
<td>c3</td>
</tr>
<tr>
<td>c32</td>
<td>c3</td>
</tr>
<tr>
<td>c33</td>
<td>c3</td>
</tr>
<tr>
<td>c43</td>
<td>c3</td>
</tr>
</tbody>
</table>