# COMP90038 Assignment Project Exam Help Algorithms, and Complexity

Lecture 20: Greedy Algorithms – Prim and Dijkstra Add WeChat powcoder (with thanks to Harald Søndergaard & Michael Kirley)

Andres Munoz-Acosta

munoz.m@unimelb.edu.au

Peter Hall Building G.83

#### Recap

- We have talked a lot about dynamic programming:

  - DP is bottom-up problem solving technique.
     Assignment Project Exam Help
     Similar to divide-and-conquer; however, problems are overlapping, making tabulation a requirement tps://powcoder.com
  - Solutions often involve recursión.

#### Add WeChat powcoder

- We applied this idea to two graph problems:
  - Computing the transitive closure of a directed graph; and
  - Finding shortest distances in weighted directed graphs.

#### A practice challenge

Can you solve the problem in the figure?

Assignment Project Exam He

• W = 15

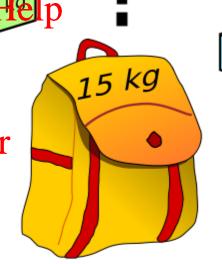
• w = [1 1 2 4 12]

• v = [1 2 2 10 4]

https://powcoder.com

Add WeChat powcoder

- Because it is a larger instance, memoing is preferable.
  - How many states do we need to evaluate?
- FYI the answer is \$15 {1,2,3,4}







#### The table

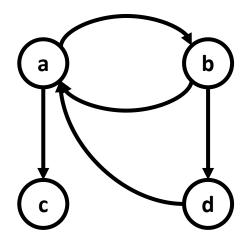
			j	(	) (	1	2	3	4	_	5 6	•	8	_		11	12	13	14	15
W	١	/	i			A	As	sigi	nme	nt P	roje	ect E	xam	i He	lp					
			0																	
	1	1	1			1	1	ht	tps1	//po	WCO	oden	con	1	. 1	1	1	1	1	1
	1	2	2			2	-1	3	-1	-1	1 -1								-1	3
	2	2	3		-1	1	-1	A	dd-1	Wed	Chat	pov	vcot	ler-1	-1	5	-1	-1	-1	5
	4	10	4		-1	1	-1	4	-1	-1	1 -1	լ -1		-1		-1	-1	-1	-1	15
	12	4	5		-1	1	-1	-1	-1	-1	l -1	L -1	1	-1	-1	-1	-1	-1	-1	15

• We know that we include all the elements up to 4 because the last column (15) is the cumulative sum of the values.

- Warshall's algorithm computes the transitive closure of a directed graph.
  - An edge (a,d) is in the saignitive nto Pure jorgar Laborator in G from a to d.

https://powcoder.com

• Is there a path from node i to node i using nodes [1 ... k] as "stepping stones"?



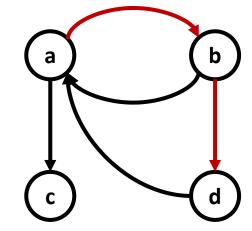
- Such path will exist if and only if we can:
  - step from *i* to *j* using only nodes [1 ... *k*-1], or
  - step from *i* to *k* using only nodes [1 ... *k*-1], and then step from *k* to *j* using only nodes [1 ... *k*-1].

$$\left[\begin{array}{cccc} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array}\right]$$

- Warshall's algorithm computes the transitive closure of a directed graph.
  - An edge (a,d) is in the saignitive nto Pure jorgar Laborator in G from a to d.

https://powcoder.com

• Is there a path from node i to node i using nodes [1 ... k] as "stepping stones"?



- Such path will exist if and only if we can:
  - step from *i* to *j* using only nodes [1 ... *k*-1], or
  - step from *i* to *k* using only nodes [1 ... *k*-1], and then step from *k* to *j* using only nodes [1 ... *k*-1].

$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

• If G's adjacency matrix is A then we can express the recurrence relation as:

Assignment Project Exam Help  $Assignment_{[i,j,0]} = A[i,j]$ 

 $R[i,j,k] = R[i \text{ hit has://powkiokler.chand} \ R[k,j,k-1])$ 

• We examined the simpledtive solohoof the calgorithm.

```
for k \leftarrow 1 to n do

for i \leftarrow 1 to n do

if A[i,k] then

for j \leftarrow 1 to n do

if A[k,j] then

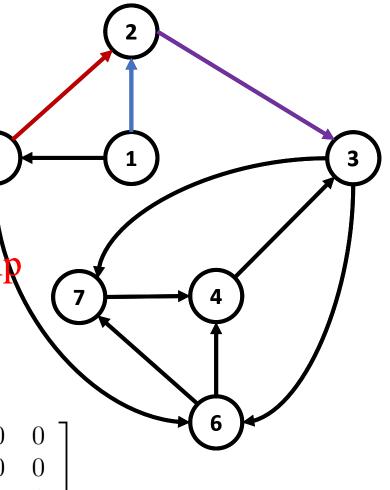
A[i,j] \leftarrow 1
```

Let's visualize the steps.

Assignment Project Exam Help

Using node 2 (k=2), we can reach node 3 from nodes 1 and 2 and 5.
 Add WeChat powcoder

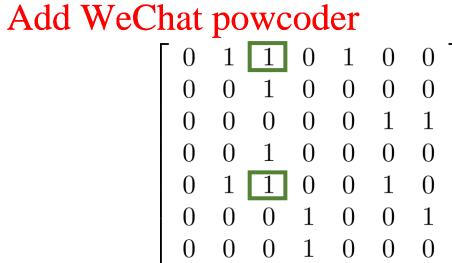
 $\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \end{bmatrix}$ 



Let's visualize the steps.

Assignment Project Exam Help

Using node 2 (k=2), we can reach node 3 from nodes 1 and powcoder.com
 5.

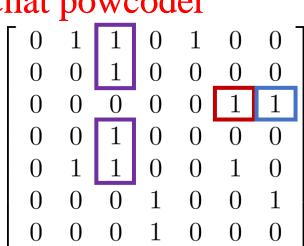


• Let's visualize the steps.

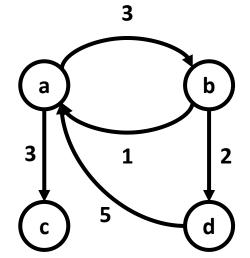
Assignment Project Exam Hel

Using node 2 (k=2), we can reach node 3 from nodes 1 and 2 and 5.
 Add WeChat powcoder

 Using node 3 (k=3) we can reach: Nodes [6 7] from nodes [1,2,5]

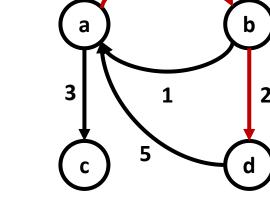


- Floyd's algorithm solves the **all-pairs shortest-path** problem for weighted graphs with **positive weights**.
  - It works for directed assign as embleroged grapham Help
- https://powcoder.com
   What is the shortest path from node *i* to node *j* using nodes [1 ... *k*] as "stepping stopes" Chat powcoder



- Such path will exist if and only if we can:
  - step from *i* to *j* using only nodes [1 ... *k*-1], or
  - step from *i* to *k* using only nodes [1 ... *k*-1], and then step from *k* to *j* using only nodes [1 ... *k*-1].

- Floyd's algorithm solves the **all-pairs shortest-path** problem for weighted graphs with **positive weights**.
  - It works for directed assign as embleroged of Epham Help
- https://powcoder.com
   What is the shortest path from node *i* to node *j* using nodes [1 ... *k*] as "stepping stopes" Chat powcoder



3

- Such path will exist if and only if we can:
  - step from *i* to *j* using only nodes [1 ... *k*-1], or
  - step from *i* to *k* using only nodes [1 ... *k*-1], and then step from *k* to *j* using only nodes [1 ... *k*-1].

 If G's weight matrix is W then we can express the recurrence relation as:

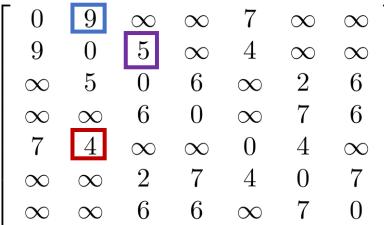
Assignment Project Exam Help 
$$D[i,j,k] = \min \{ D[i,j,k] = \min \{ D[i,j,k] \} \} D[i,j,k] = \min \{ D[i,j,k] \} D[i,j,k] D[$$

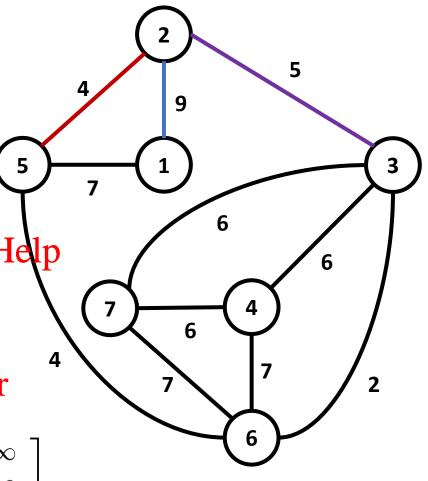
• A simpler version updathold WeChat powcoder

```
\begin{aligned} & \textbf{function} \ \text{FLOYD}(W[\cdot, \cdot], n) \\ & D \leftarrow W \\ & \textbf{for} \ k \leftarrow 1 \ \text{to} \ n \ \textbf{do} \\ & \textbf{for} \ i \leftarrow 1 \ \text{to} \ n \ \textbf{do} \\ & \textbf{for} \ j \leftarrow 1 \ \text{to} \ n \ \textbf{do} \\ & D[i, j] \leftarrow \min \left(D[i, j], D[i, k] + D[k, j]\right) \\ & \textbf{return} \ D \end{aligned}
```

- For *k*=2
  - We can go  $1 \rightarrow 2 \rightarrow 3$ , the distance  $1 \rightarrow 3$  is 9 + 5 = 14 Assignment Project Exam Help
  - We can go  $5 \rightarrow 2 \rightarrow 3$ , the distance of  $5 \rightarrow 3$  is 4 + 5 = 9

Add WeChat powcoder





• For *k*=2

• We can go  $1 \rightarrow 2 \rightarrow 3$ , the distance  $1 \rightarrow 3$  is 9 + 5 = 14 Assignment Project Exam Help

• We can go  $5 \rightarrow 2 \rightarrow 3$ , the distance of  $5 \rightarrow 3$  is 4 + 5 = 9

• The distance matrix get dd WeChat powcoder updated to:

 $\begin{bmatrix} 0 & 9 & 14 & \infty & 7 & \infty & \infty \\ 9 & 0 & 5 & \infty & 4 & \infty & \infty \\ 14 & 5 & 0 & 6 & 9 & 2 & 6 \\ \infty & \infty & 6 & 0 & \infty & 7 & 6 \\ 7 & 4 & 9 & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 7 & 4 & 0 & 7 \\ \infty & \infty & 6 & 6 & \infty & 7 & 0 \end{bmatrix}$ 

#### Greedy Algorithms

- A problem solving strategy is to take the locally best choice among all feasible ones.
  - Once we do this, Assignment Preject Fram Help

#### https://powcoder.com

- We want to change 30 cents using the smallest number of coins.
  - If we assume coin denomination of 25,010,05,05, we could use as many 25-cent pieces as we can, then do the same for 10-cent pieces, and so on, until we have reached 30 cents (25+5).
  - This **greedy** strategy would not work for denominations {25, 10, 1} (25+1+1+1+1+1 compared to 10+10+10).

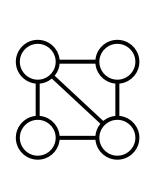
#### Greedy Algorithms

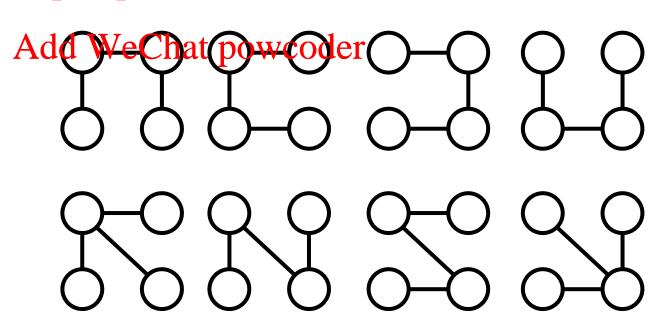
- In general, it is unusual that **locally best** choices yield **global best** results.
  - However, there are problems for which a greedy algorithm is correct and fast.

    https://powcoder.com
  - https://powcoder.com
     In some other problems, a greedy algorithm serve as an acceptable approximation algorithmed WeChat powcoder
- Here we shall look at:
  - Prim's algorithm for finding minimum spanning trees
  - Dijkstra's algorithm for single-source shortest paths

#### What is an Spanning Tree?

- Recall that a **tree** is a connected graph with no cycles.
- A spanning tree of a graph  $\langle V, E \rangle$  is a tree  $\langle V, E' \rangle$  where E' is a subset of Assignment Project Exam Help
- For example, the graphhopsheplest doale eight different spanning trees:



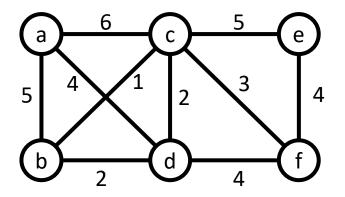


#### Minimum Spanning Trees of Weighted Graphs

- For a weighted graph, some spanning trees are more desirable than others.
  - For example, suppose we have a set of "stations" to connect in a network, and also some possible connections, each with its own cost.

- https://powcoder.com

   This is the problem of finding a spanning tree with the smallest possible • Such tree is a minimum spanning tree for the graph. cost.

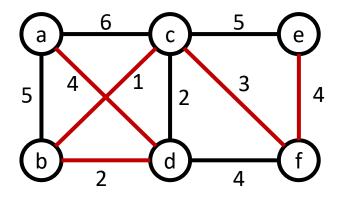


#### Minimum Spanning Trees of Weighted Graphs

- For a weighted graph, some spanning trees are more desirable than others.
  - For example, suppose we have a set of "stations" to connect in a network, and also some possible connections, each with its own cost.

- https://powcoder.com

   This is the problem of finding a spanning tree with the smallest possible • Such tree is a minimum spanning tree for the graph. cost.



- Prim's algorithm is an example of a greedy algorithm.
  - It constructs a sequence of subtrees T, by adding to the latest tree the closest node not charging them. Project Exam Help
- A simple version:

https://powcoder.com

```
function PRIM(\langle V, E \rangle) eChat powcoder V_T \leftarrow \{v_0\} E_T \leftarrow \emptyset for i \leftarrow 1 to |V| - 1 do find a minimum-weight edge (v, u) \in V_T \times (V \setminus V_T) V_T \leftarrow V_T \cup \{u\} E_T \leftarrow E_T \cup \{(v, u)\} return E_T
```

• But how to find the **minimum-weight edge** (*v*,*u*)?

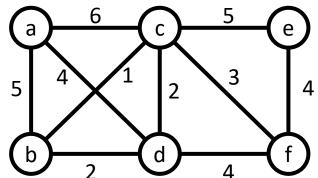
#### Assignment Project Exam Help

- A standard way to do this is to organise the nodes that are not yet included in the spanning tree pas a priority queue, organised in a min-heap by edge cost<sub>Add</sub> WeChat powcoder
- The information about which nodes are connected in T can be captured by an array *prev* of nodes, indexed by V. Namely, when (v,u) is included, this is captured by setting prev[u] = v.

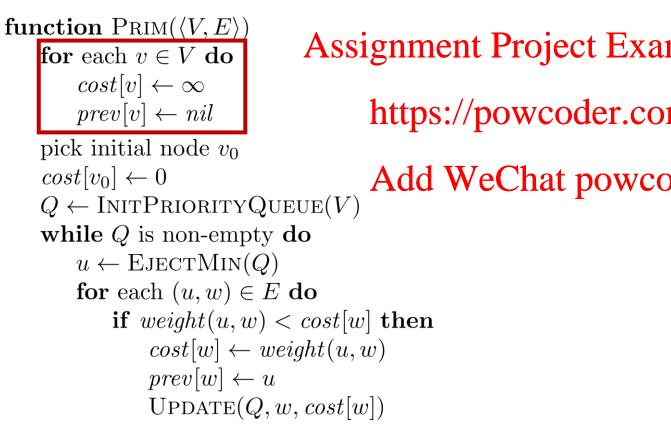
• The complete algorithm is:

```
function Prim (V, F) nment Project Exam Help
        \begin{array}{l} cost[v] \leftarrow \infty \\ prev[v] \leftarrow nil \end{array} https://powcoder.com
    pick initial node v_0

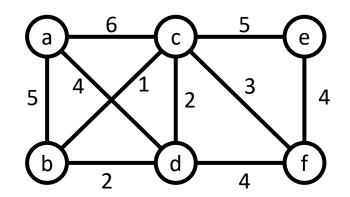
cost[v_0] \leftarrow 0 Add WeChat powcoder
    Q \leftarrow \text{InitPriorityQueue}(V)
                                                             > priorities are cost values
    while Q is non-empty do
        u \leftarrow \text{EJECTMIN}(Q)
        for each (u, w) \in E do
            if weight(u, w) < cost[w] then
                cost[w] \leftarrow weight(u, w)
                prev[w] \leftarrow u
                UPDATE(Q, w, cost[w])
                                                            > rearranges priority queue
```



On the first loop, we only create the table



	2			4				
	Tree T		a	b	С	d	е	f
ลา	n Help	cost	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
		prev	nil	nil	nil	nil	nil	nil
01	n							
co	der							



cost

cost

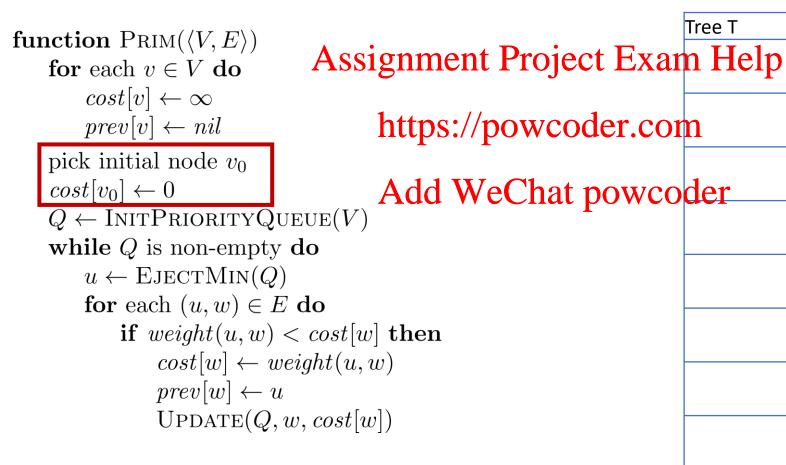
 $\infty$   $\infty$   $\infty$   $\infty$ 

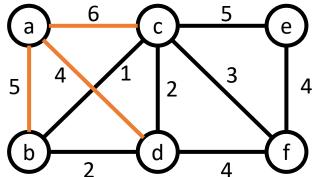
 $0 \quad \infty \quad \infty \quad \infty \quad \infty$ 

prev nil nil nil nil nil nil

prev **nil** nil nil nil nil nil

Then we pick the first node as the initial one





 We take the first node out of the queue and update the costs

```
function PRIM(\langle V, E \rangle)
for each v \in V do

cost[v] \leftarrow \infty
prev[v] \leftarrow nil
pick initial node v_0
cost[v_0] \leftarrow 0
Q \leftarrow INITPRIORITYQUEUE(V)
while Q is non-empty do

u \leftarrow EJECTMIN(Q)
Assignment Project Example of the each v \in V do

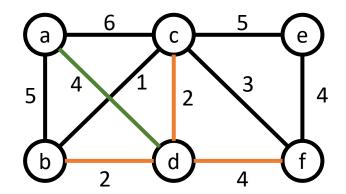
Assignment Project Example of the each v \in V do

https://powcoder.complete of the each <math>v \in V do

https://powcoder.co
```

THE & 15 HOH CHIPTY GO
$u \leftarrow \mathrm{EjectMin}(Q)$
for each $(u, w) \in E$ do
if $weight(u, w) < cost[w]$ then
$cost[w] \leftarrow weight(u, w)$
$prev[w] \leftarrow u$
$\mathrm{UPDATE}(Q, w, cost[w])$

b 2	-d	<b>)</b> —	4		f		
Tree T		a	b	С	d	е	f
.m Help	cost	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	prev	nil	nil	nil	nil	nil	nil
	cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
m	prev	nil	nil	nil	nil	nil	nil
	cost		5	6	4	$\infty$	$\infty$
o <mark>der</mark>	prev		a	a	a	nil	nil



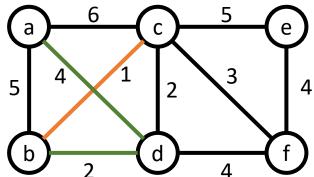
• We eject the node with the lowest cost and update the queue.

```
function PRIM(\langle V, E \rangle)
for each v \in V do
cost[v] \leftarrow \infty
prev[v] \leftarrow nil
pick initial node v_0
cost[v_0] \leftarrow 0
Q \leftarrow INITPRIORITYQUEUE(V)
while Q is non-empty do

Assignment Project Example to Assignment Proj
```

 $u \leftarrow \text{EJECTMIN}(Q)$   $\textbf{for } \text{each } (u, w) \in E \textbf{ do}$  if weight(u, w) < cost[w] then  $cost[w] \leftarrow weight(u, w)$   $prev[w] \leftarrow u$  UPDATE(Q, w, cost[w])

Tree T		a	b	С	d	e	f
n Help	cost	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
ii iioip	prev	nil	nil	nil	nil	nil	nil
	cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
n	prev	nil	nil	nil	nil	nil	nil
2	cost		5	6	4	$\infty$	$\infty$
der	prev		а	а	a	nil	nil
a,d	cost		2	2		$\infty$	4
a,u	prev		d	d		nil	d



We eject the next node based on alphabetical order.

## Why is f not updated? function $PRIM(\langle V, E \rangle)$ for each $v \in V$ do Assignment Project Example $cost[v] \leftarrow \infty$ $prev[v] \leftarrow nil$ https://powcoder.com/

pick initial node  $v_0$   $cost[v_0] \leftarrow 0$ 

 $Q \leftarrow \text{InitPriorityQueue}(V)$ 

while Q is non-empty do

then
$\mathcal{O}$
])

				4				
on alphabetical order.	<del>-</del>							
	Tree T		a	b	С	d	е	f
ignment Project Exar	n Help	cost	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1811110110 1 1 0 1 0 0 0 0 0 1 1 1 1 1	т 1101р	prev	nil	nil	nil	nil	nil	nil
https://porreador.com		cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
https://powcoder.com	П	prev	nil	nil	nil	nil	nil	nil
	a	cost		5	6	4	$\infty$	$\infty$
Add WeChat powco	der	prev		a	a	а	nil	nil
1	a,d	cost		2	2		$\infty$	4
		prev		d	d		nil	d
	a,d,b	cost			1		$\infty$	4
	4,4,5	prev			b		nil	d
v] then								
w)								
ω)								
[v])							$\Box$	
~] <i>)</i>							$\Box$	

a 6 c 5 e 5 b 2 d f

• We now update f

function $PRIM(\langle V, E \rangle)$ for each $v \in V$ do	Assignment Project Example 1
$cost[v] \leftarrow \infty$ $prev[v] \leftarrow nil$	https://powcoder.com
pick initial node $v_0$ $cost[v_0] \leftarrow 0$ $Q \leftarrow \text{InitPriorityQue}$	Add WeChat powco
while $Q$ is non-empty $\mathbf{c}$	· /
$u \leftarrow \text{EJECTMIN}(Q)$ <b>for</b> each $(u, w) \in E$	d o

if weight(u, w) < cost[w] then

 $cost[w] \leftarrow weight(u, w)$ 

UPDATE(Q, w, cost[w])

 $prev[w] \leftarrow u$ 

	а	b	С	d	е	f
cost	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
prev	nil	nil	nil	nil	nil	nil
cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
prev	nil	nil	nil	nil	nil	nil
cost		5	6	4	$\infty$	$\infty$
prev		а	а	a	nil	nil
cost		2	2		$\infty$	4
prev		d	d		nil	d
cost			1		$\infty$	4
prev			b		nil	d
cost					5	3
prev					C	C
	cost prev cost prev cost prev cost prev cost	cost   prev  nil cost   prev  nil cost   prev   cost	cost $\infty$ $\infty$ prev nil nil cost $0$ $\infty$ prev nil nil cost $5$ prev a cost $2$ prev $d$ cost prev $d$	cost         ∞         ∞           prev         nil         nil         nil           cost         0         ∞         ∞           prev         nil         nil         nil           cost         5         6           prev         a         a           cost         2         2           prev         d         d           cost         1         prev           cost         b         cost	cost         ∞         ∞         ∞         ∞           prev         nil         nil         nil         nil           cost         0         ∞         ∞         ∞           prev         nil         nil         nil         nil           nil         nil         nil         nil         nil           cost         5         6         4         4           prev         a         a         a         a           cost         2         2         2           prev         d         d         d           cost         1         prev         b           cost         .         .         .	cost         ∞         ∞         ∞         ∞           prev         nil         nil         nil         nil         nil           cost         0         ∞         ∞         ∞         ∞           prev         nil         nil         nil         nil           cost         2         2         ∞           prev         d         d         nil           cost         1         ∞           prev         b         nil           cost         5         6         4

a 6 c 5 e 5 b 2 d f

We reach the last choice

function $PRIM(\langle V, E \rangle)$ for each $v \in V$ do	Assignment Project Exar
$\begin{array}{l} cost[v] \leftarrow \infty \\ prev[v] \leftarrow nil \end{array}$	https://powcoder.cor
pick initial node $v_0$ $cost[v_0] \leftarrow 0$ $Q \leftarrow \text{InitPriorityQue}$	Add WeChat powco
while $Q$ is non-empty $\mathbf{c}$	` /
$u \leftarrow \text{EjectMin}(Q)$	
for each $(u, w) \in E$	do
if $weight(u, w) <$	

 $cost[w] \leftarrow weight(u, w)$ 

UPDATE(Q, w, cost[w])

 $prev[w] \leftarrow u$ 

Tree T		a	b	С	d	е	f
n Help	cost	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
птогр	prev	nil	nil	nil	nil	nil	nil
	cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
$\mathbf{n}$	prev	nil	nil	nil	nil	nil	nil
	cost		5	6	4	$\infty$	$\infty$
der	prev		a	a	a	nil	nil
	cost		2	2		$\infty$	4
a,d	prev		d	d		nil	d
a d b	cost			1		$\infty$	4
a,d,b	prev			b		nil	d
adba	cost					5	3
a,d,b,c	prev					С	С
- d b - f	cost					4	
a,d,b,c,f	prev					f	

a 6 c 5 e 5 b 2 d f

• The resulting tree is {a,d,b,c,f,e}

function $PRIM(\langle V, E \rangle)$ for each $v \in V$ do	Assignment Project Exan
$\begin{array}{l} cost[v] \leftarrow \infty \\ prev[v] \leftarrow nil \end{array}$	https://powcoder.com
pick initial node $v_0$ $cost[v_0] \leftarrow 0$ $Q \leftarrow \text{InitPriorityQue}$	Add WeChat powco
while $Q$ is non-empty $\mathbf{c}$	
$u \leftarrow \text{EJECTMIN}(Q)$ <b>for</b> each $(u, w) \in E$	do
if $weight(u, w) \in E$	cost[w] then

 $cost[w] \leftarrow weight(u, w)$ 

UPDATE(Q, w, cost[w])

 $prev[w] \leftarrow u$ 

Tree T		a	b	С	d	е	f
n Help	cost	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
ii iioip	prev	nil	nil	nil	nil	nil	nil
	cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
n	prev	nil	nil	nil	nil	nil	nil
2	cost		5	6	4	$\infty$	$\infty$
der	prev		a	a	a	nil	nil
a,d	cost		2	2		$\infty$	4
a,u	prev		d	d		nil	d
a d b	cost			1		$\infty$	4
a,d,b	prev			b		nil	d
adba	cost					5	3
a,d,b,c	prev					С	C
adbaf	cost					4	
a,d,b,c,f	prev					f	
adhafa	cost						
a,d,b,c,f,e	prev						

## Analysis of Prim's Algorithm

- First, a crude analysis: For each node, we look through the edges to find those incident to the node, and pick the one with smallest cost. Thus we get  $O(|V| \times |E|)$ . However means property that patroctures.
- Using adjacency lists for the graph and a min-heap for the priority queue, we perform |V| 1 heap deletions (each at cost  $O(\log |V|)$ ) and |E| updates of priorities (each at cost  $O(\log |V|)$ ).
- Altogether  $(|V|-1+|E|) O(\log |V|)$ .
- Since, in a connected graph,  $|V|-1 \le |E|$ , this is  $O(|E| \log |V|)$ .

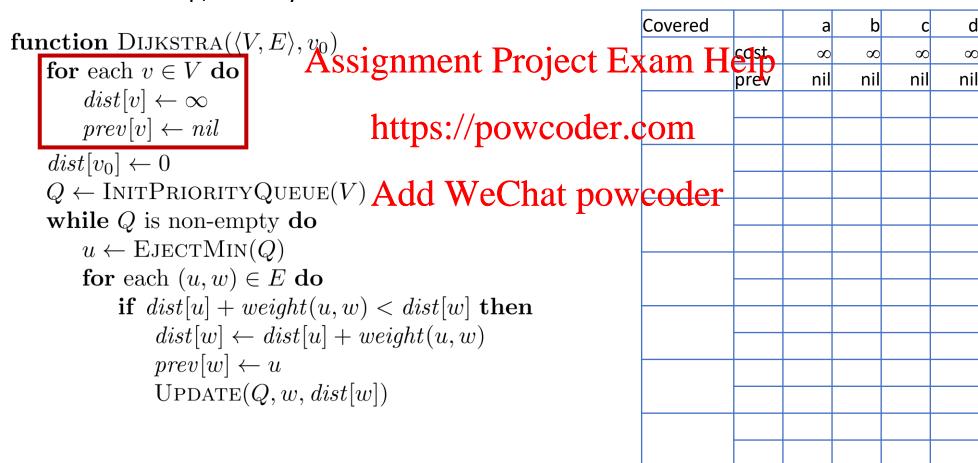
- Another classical greedy weighted-graph algorithm is Dijkstra's algorithm, whose overall structure is the same as Prim's.
   Assignment Project Exam Help
- Recall that Floyd's algorithm/gave us the chartest paths, for every pair of nodes, in a (directed or undirected) weighted graph. It assumed an adjacency matrix representation bandphadcool plexity  $O(|V|^3)$ .
- **Dijkstra's algorithm** is also a shortest-path algorithm for (directed or undirected) weighted graphs. It finds all shortest paths **from a fixed start node**. Its complexity is the same as that of Prim's algorithm.

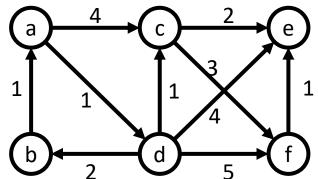
• The complete algorithm is:

```
function Dijkstra(\langle V, E \rangle, v_0)
for each v \in \mathbf{A} signment Project Exam Help
        dist[v] \leftarrow \infty
       prev[v] \leftarrow nil https://powcoder.com
    dist[v_0] \leftarrow 0
   Q \leftarrow \text{InitPriority}  Chat powcoderriorities are distances
    while Q is non-empty do
        u \leftarrow \text{EJECTMIN}(Q)
        for each (u, w) \in E do
            if dist[u] + weight(u, w) < dist[w] then
                dist[w] \leftarrow dist[u] + weight(u, w)
               prev[w] \leftarrow u
                UPDATE(Q, w, dist[w])
                                                         > rearranges priority queue
```

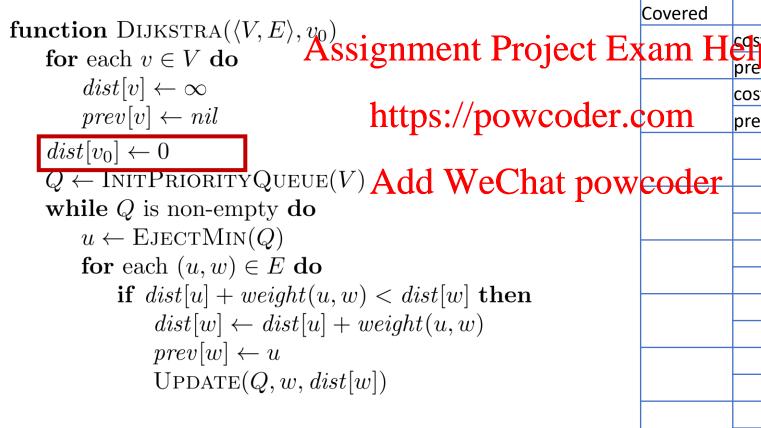
nil

• On the first loop, we only create the table

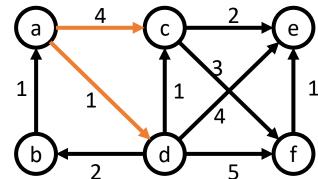




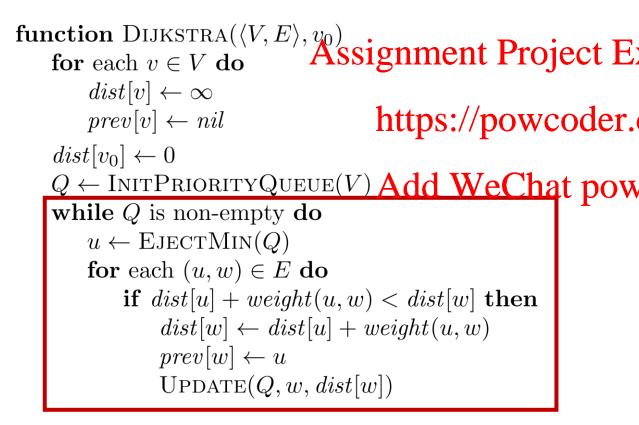
Then we pick the first node as the initial one



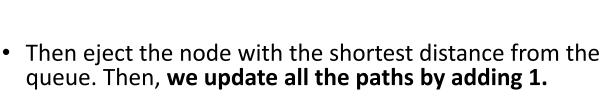
			2		J	,		
	Covered		a	b	С	d	е	f
Y	am H	cast	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
Λ	am H	prev	nil	nil	nil	nil	nil	nil
		cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	om	prev	nil	nil	nil	nil	nil	nil
<b>V</b>	coder							



Then we pick the first node as the initial one



			2		5			
	Covered		a	b	С	d	е	f
V	am H	cast	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
1	am m	prev	nil	nil	nil	nil	nil	nil
		cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
¢	om	prev	nil	nil	nil	nil	nil	nil
		cost		$\infty$	4	1	$\infty$	$\infty$
7	coder	prev		nil	a	a	nil	nil
V	Juci							



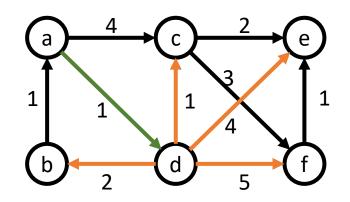
function Dijkstra $(\langle V, E \rangle, v_0)$ Assignment Project Ex  $dist[v] \leftarrow \infty$ 

 $prev[v] \leftarrow nil$ 

 $dist[v_0] \leftarrow 0$ 

 $Q \leftarrow \text{InitPriorityQueue}(V) \text{ Add WeChat pow}$ 

while Q is non-empty do  $u \leftarrow \text{EJECTMIN}(Q)$ for each  $(u, w) \in E$  do if dist[u] + weight(u, w) < dist[w] then  $dist[w] \leftarrow dist[u] + weight(u, w)$  $prev[w] \leftarrow u$ UPDATE(Q, w, dist[w])



paths by adding 1.									
patris by adding 1.		Covered		a	b	С	d	e	f
ignment Proj	ect Ex	am H	sast	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
		Calli II	prev	nil	nil	nil	nil	nil	nil
4	4		cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
https://powc	oder.c	com	prev	nil	nil	nil	nil	nil	nil
			cost		$\infty$	4	1	$\infty$	$\infty$
Add WeCha	t now	coder	prev		nil	a	а	nil	nil
	it pow		cost		3	2		5	6
		a,d	prev		d	d		d	d
) < dist[su] then									
(v) < dist[w]  then									
weight(u, w)									
7.									
w])									
	•								

 $dist[v_0] \leftarrow 0$ 

 Our next node will be the one with the shortest path in overall (b)

```
function Dijkstra(\langle V, E \rangle, v_0)

for each v \in V do Assignment Project Ex

dist[v] \leftarrow \infty

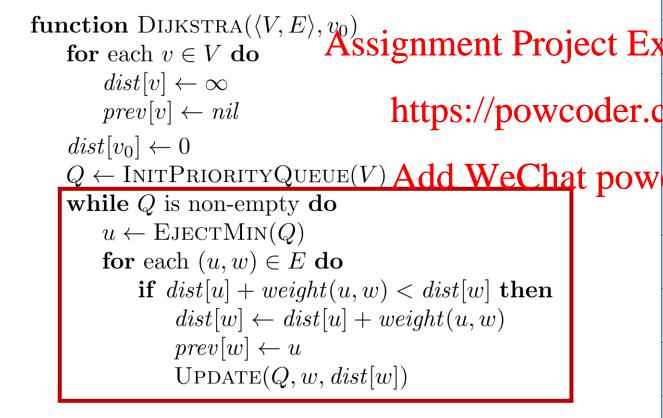
prev[v] \leftarrow nil https://powcoder.com
```

 $Q \leftarrow \text{InitPriorityQueue}(V) \text{ Add WeChat power}$ 

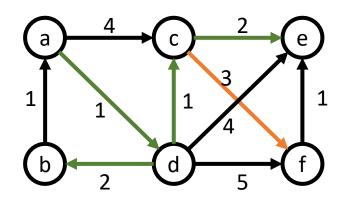
while Q is non-empty do  $u \leftarrow \operatorname{EJECTMIN}(Q)$  for each  $(u,w) \in E$  do if  $\operatorname{dist}[u] + \operatorname{weight}(u,w) < \operatorname{dist}[w]$  then  $\operatorname{dist}[w] \leftarrow \operatorname{dist}[u] + \operatorname{weight}(u,w)$   $\operatorname{prev}[w] \leftarrow u$  UPDATE $(Q,w,\operatorname{dist}[w])$ 

Covered		a	b	С	d	е	f
am H	cast	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
kam H	prev	nil	nil	nil	nil	nil	nil
	cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
com	prev	nil	nil	nil	nil	nil	nil
2	cost		$\infty$	4	1	$\infty$	$\infty$
coder	prev		nil	a	а	nil	nil
	cost		3	2		5	6
a,d	prev		d	d		d	d
a d a	cost		3			4	5
a,d,c	prev		d			С	С

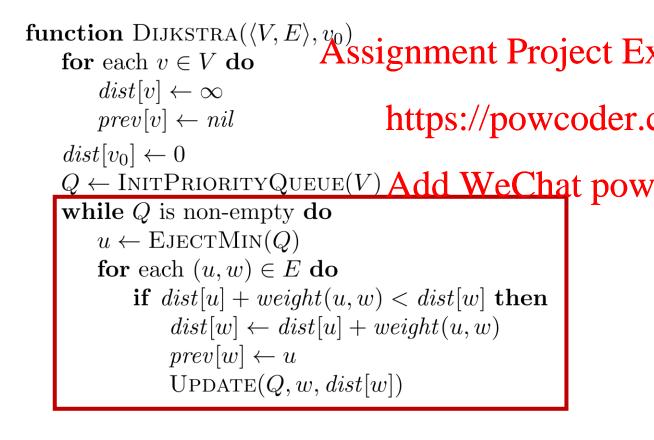
Now, we continue evaluating from (c)



Covered		a	b	С	d	e	f
am H	sqst	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
am H	prev	nil	nil	nil	nil	nil	nil
	cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
com	prev	nil	nil	nil	nil	nil	nil
2	cost		$\infty$	4	1	$\infty$	$\infty$
a coder	prev		nil	a	а	nil	nil
	cost		3	2		5	6
a,d	prev		d	d		d	d
a d a	cost		3			4	5
a,d,c	prev		d			С	C
a d a b	cost					4	5
a,d,c,b	prev					C	C



We arrive at our last decision.

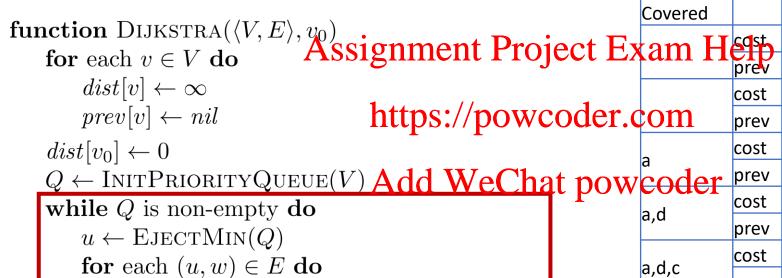


Covered		a	b	С	d	e	1
xam H	cost	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$	$\propto$
Calli II	prev	nil	nil	nil	nil	nil	ni
	cost	0	$\infty$	$\infty$	$\infty$	$\infty$	$\propto$
com	prev	nil	nil	nil	nil	nil	ni
2	cost		$\infty$	4	1	$\infty$	$\propto$
coder	prev		nil	a	а	nil	ni
	cost		3	2		5	$\epsilon$
a,d	prev		d	d		d	C
a d a	cost		3			4	5
a,d,c	prev		d			С	(
a d a b	cost					4	5
a,d,c,b	prev					С	C
	cost						5
a,d,c,b,e	prev						(

• Our complete tree is {a,d,c,b,e,f}

 $prev[w] \leftarrow u$ 

UPDATE(Q, w, dist[w])



if dist[u] + weight(u, w) < dist[w] then

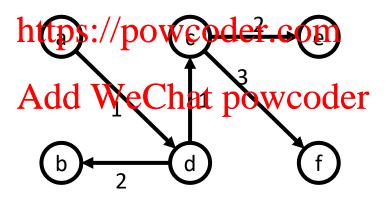
 $dist[w] \leftarrow dist[u] + weight(u, w)$ 

nil nil nil nil cost nil nil nil nil prev nil cost nil nil nil prev cost a,d prev cost a,d,c prev cost a,d,c,b prev cost a,d,c,b,e prev cost a,d,c,b,e,f prev

#### Tracing paths

• The array prev is not really needed, unless we want to retrace the shortest paths from node a

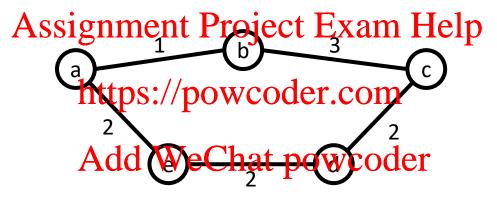
Assignment Project Exam Help



• This tree is referred as the shortest-path tree

#### Spanning trees and Shortest-Path trees

 The shortest-path tree that results from Dijkstra's algorithm is very similar to the minima spaning tree.



- Exercise:
  - Which edge is missing in the minimal spanning tree?
  - Which edge is missing from the shortest-path tree?
  - Assume that you always started from node a.

#### Next lecture

#### Assignment Project Exam Help

We will have a look to Huffman encoding for data compression
 Add WeChat powcoder