COMP20003 Workshop Week 10

1	Graphs: concepts
2	Graph representation
3	BFS & Dijkstra's Algorithm
4	Teaching Surveys
5	Lab: Simple Implementation of Dijkstra

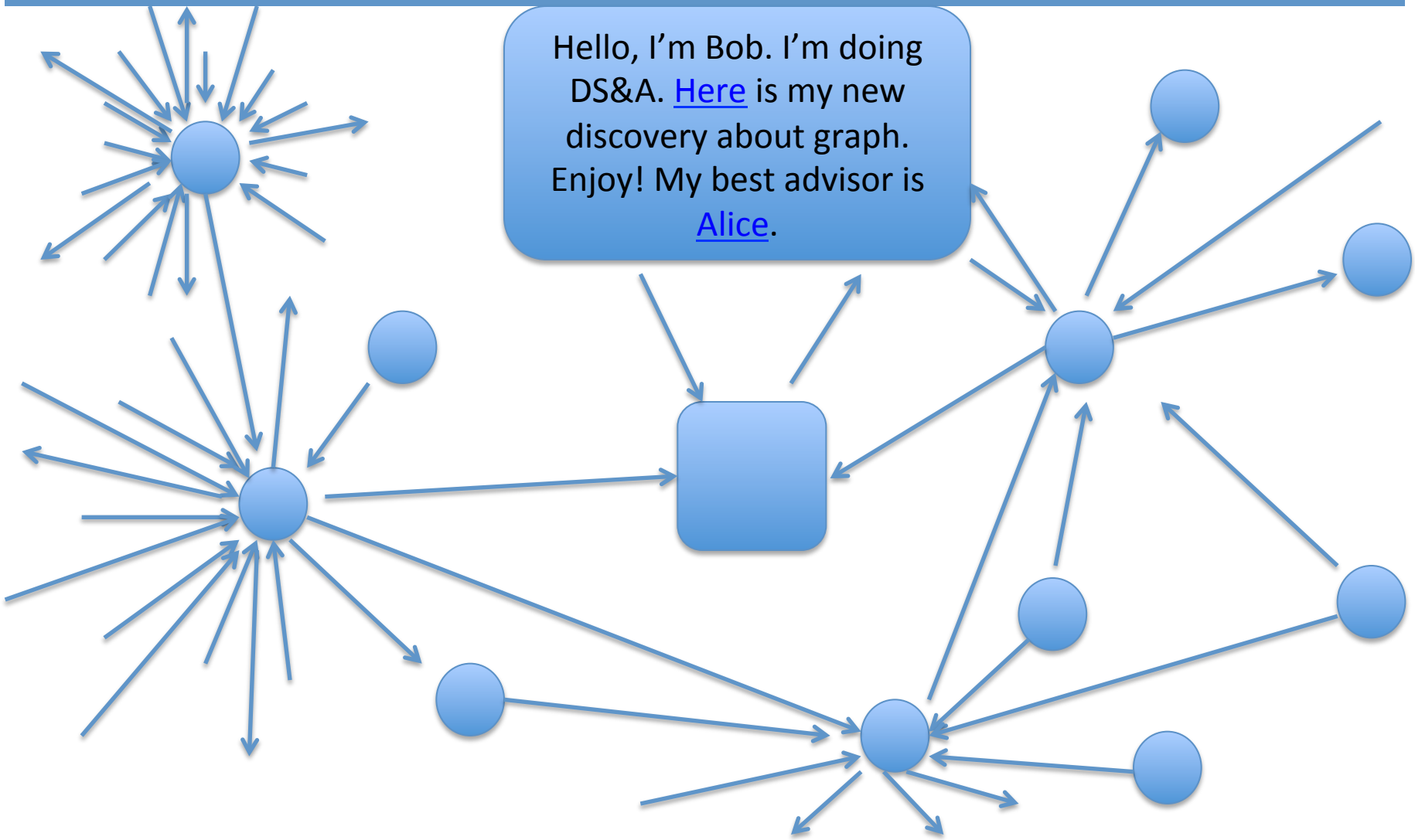
Graphs

Visualization

Any problem where graph can be applied?

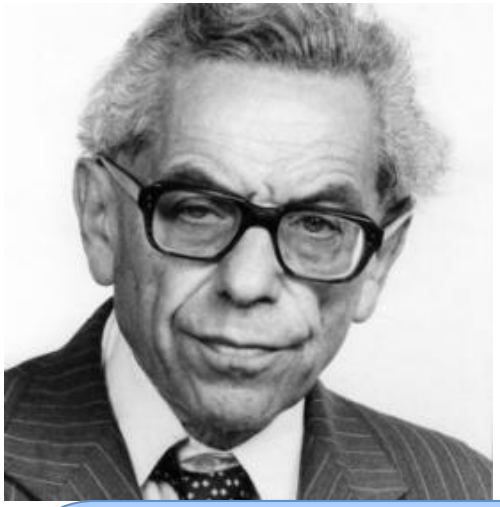
Are link lists and trees graphs? What special about them?

PageRank: Bob becoming famous!

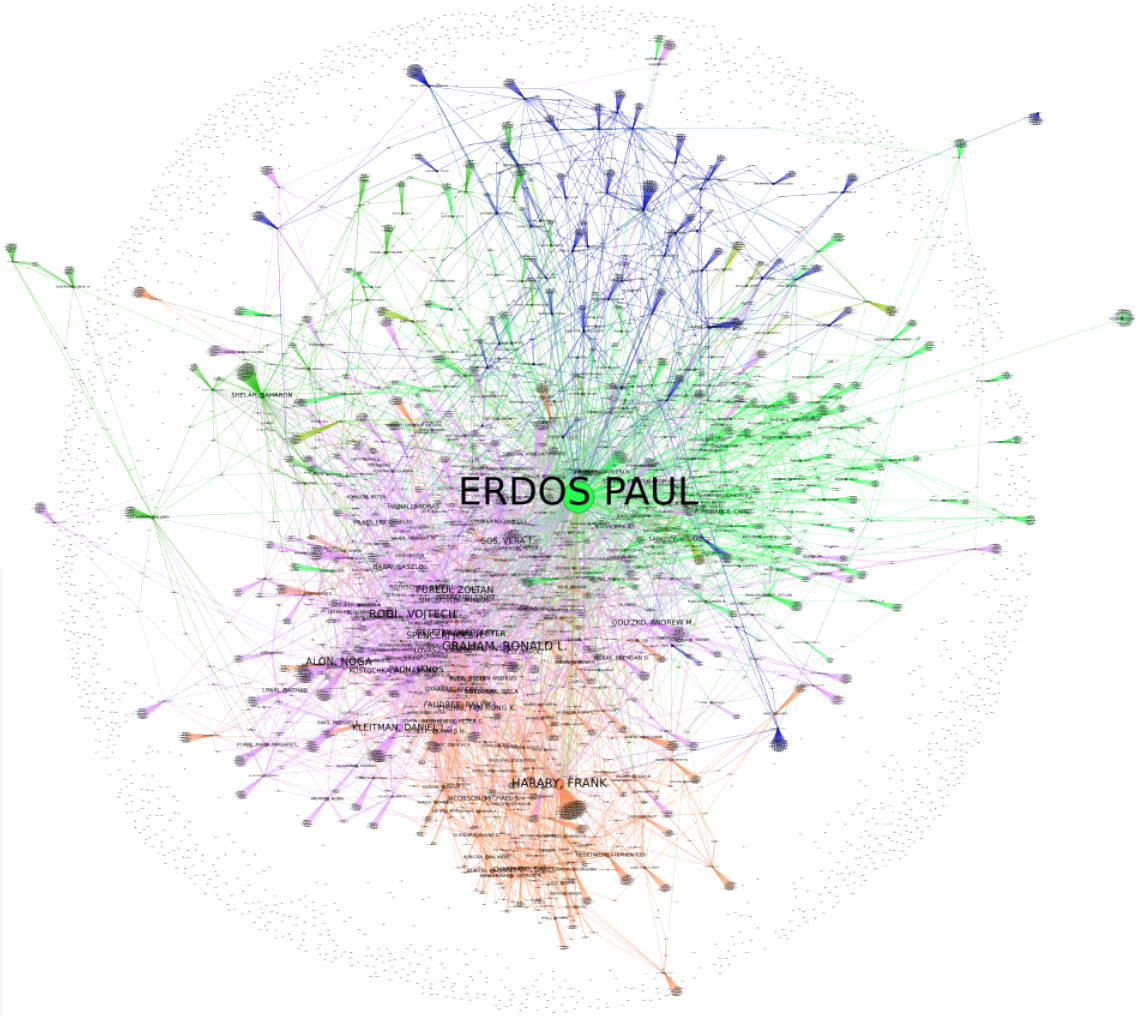


Erdős' Collaborators Graph (6927 nodes)

(http://genedan.com/tag/erdos-network-graph/)



Erdos (1913-1996) is a great math-cian. He wrote ~1500 papers, co-authored with >500 authors…



Graphs: Concepts

Formal definition: $G = (V, E)$ where

$V = \{v_i\}$: set of *vertices*, or *nodes*

$E = \{(v_i, v_j) | v_i \in V, v_j \in V\}$: set of *edges*, *arcs*, or *links*;

$|V|$ is called the *order* of the graph

$|E|$ is called the *size* of the graph

dense and *sparse* graphs

directed, *di-graph*, *undirected*, *acyclic*, *DAG*

connected graph, connected component

weakly and strongly connected components

Graphs: Representation

Representation (general approaches):

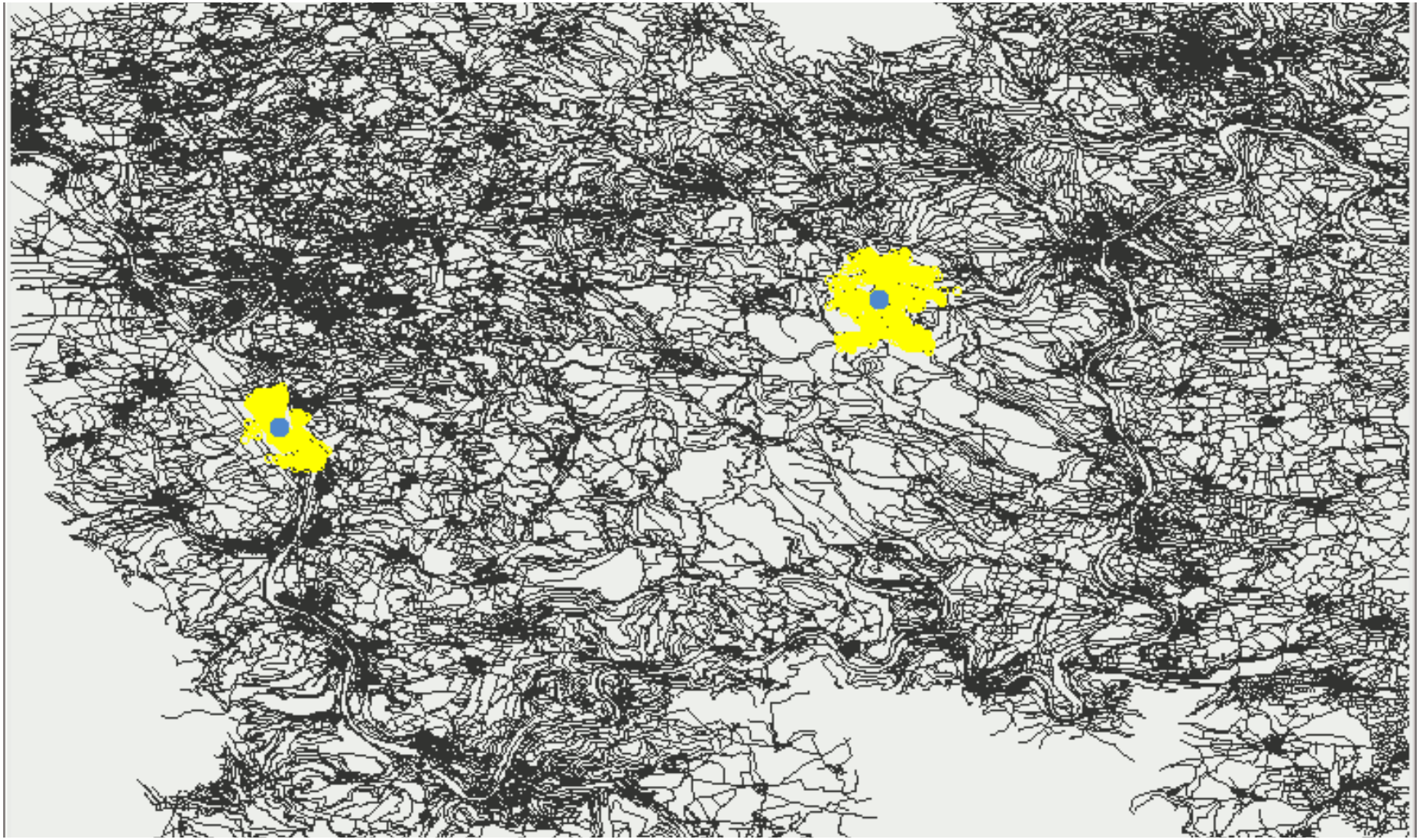
1. *Adjacency matrix* (what is that?)
2. Set of *adjacency lists* (what is that?)

What is a suitable representation method for:

- a graph of this class, where nodes represent students, edge (a,b) means “a knows b”,
- the Erdős’s collaboration graph (check your answer by visiting the website),
- the webgraph?

Graphs: DFS & BFS

Breath-first Search (BFS): visit all neighbors first



BFS from a single vertex

The task:

- Given a weighted graph $G=(V,E)$, and $s \in V$
- Visit all vertices which are reachable from s.

The algorithm:

```
enqueue(s, Q)
while (Q is not empty) {
    x= dequeue(Q)
    visit x
    for all v that (x, v) ∈ E: enqueue(v, Q)
}
```

Dijkstra's Algorithm

The task:

- Given a weighted graph $G=(V,E,w(E))$, and $s \in V$, and supposing that *all weights are positive*.
- Find shortest path (path with min total weight) from s to all other vertices.

QoCT: do it now

Please do Tutor Quality of Casual Teaching (QoCT) survey (right now!) for COMP20003 by:

Goto Link:

https://apps.eng.unimelb.edu.au/casmas/index.php?r=qoct/feedback&subjCode=COMP20003

OR visit the link from googling:

unimelb tutor survey comp20003

Dijkstra's Algorithm as a (special) BFS

Basic idea

if A → B → C → D is a shortest path

then

A → B → C is a shortest path (from A to C)

A → B is a shortest path (from A to B)

Dijkstra's Algorithm as a (special) BFS

Basic idea

if `s` → `A` → `B` is a shortest path then `s` → `A` is a shortest path.

Init:

- start with `dist[s]= 0`, and `dist[*]`= , set `unvisited_set= V`

Round 1:

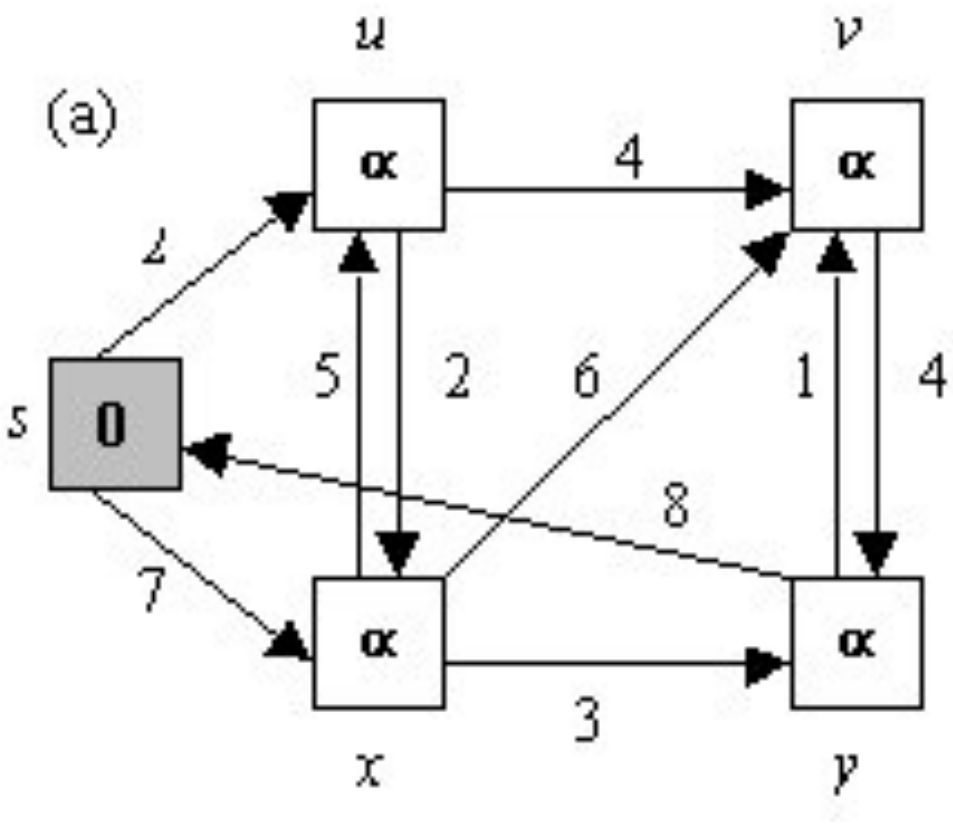
- choose node with min `dist[],` which is `s`;
- visit all nodes `u` adjacent to `s` and update `dist[u]`;
- mark `s` as visited (remove it from the `unvisited_set)`;

Round 2:

- choose the node with min `dist[]` from `unvisited_set`
- do the other steps as in Round 1

Dijkstra algorithm from vertex s: initialization

In each round, one element will be removed from A and added to B. Repeat round until A become empty.



B= { }

A= V = { s, u , v , x , y }

dist = { **0** , ∞ , ∞ , ∞ , ∞ }

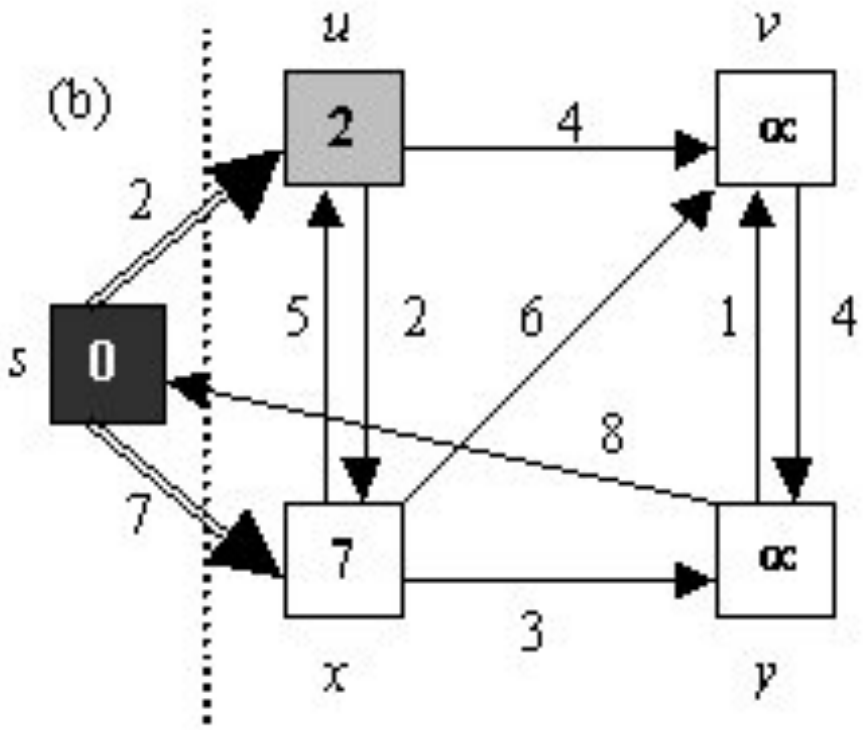
prev = { s , nil , nil , nil , nil }

visited= { 0 , 0 , 0 , 0 , 0 }

Dijkstra algorithm from vertex s

Round 1:

- choose s (node with least dist[] in A)
- update dist[] of nodes adjacent to s *if applicable*



B = { s }

dist= { 0 }

A = { , u , v , x , y }

dist = { , **2** , ∞ , 7 , ∞ }

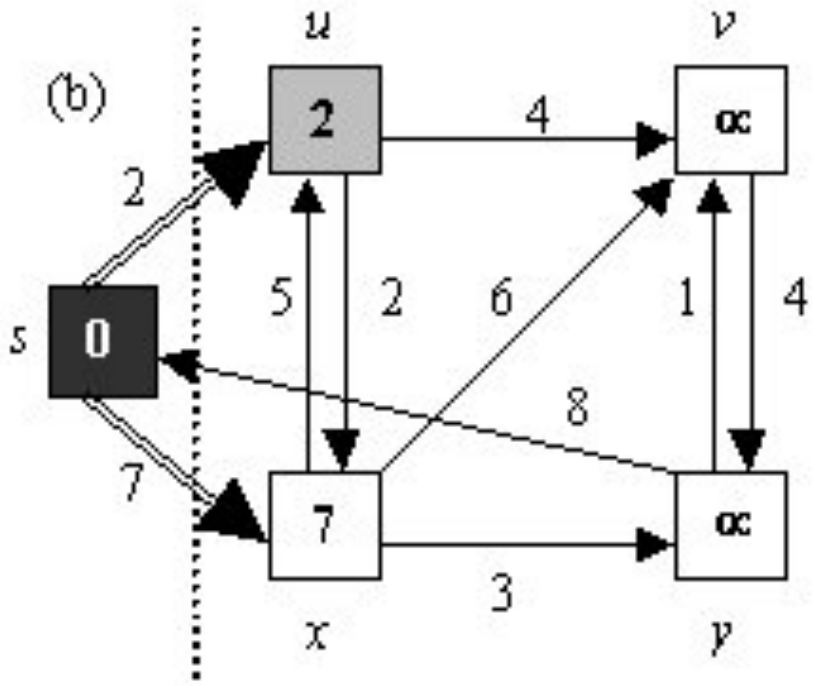
prev = { nil, s , , s , }

Dijkstra algorithm from vertex s

Round 1:

	s	u	v	x	y
0	0 s	∞ N	∞ N	∞ N	∞ N
1		2 s	∞ N	7 s	∞ N

visited=1 dist prev



B = { s }

dist= { 0 }

A = { , u , v , x , y }

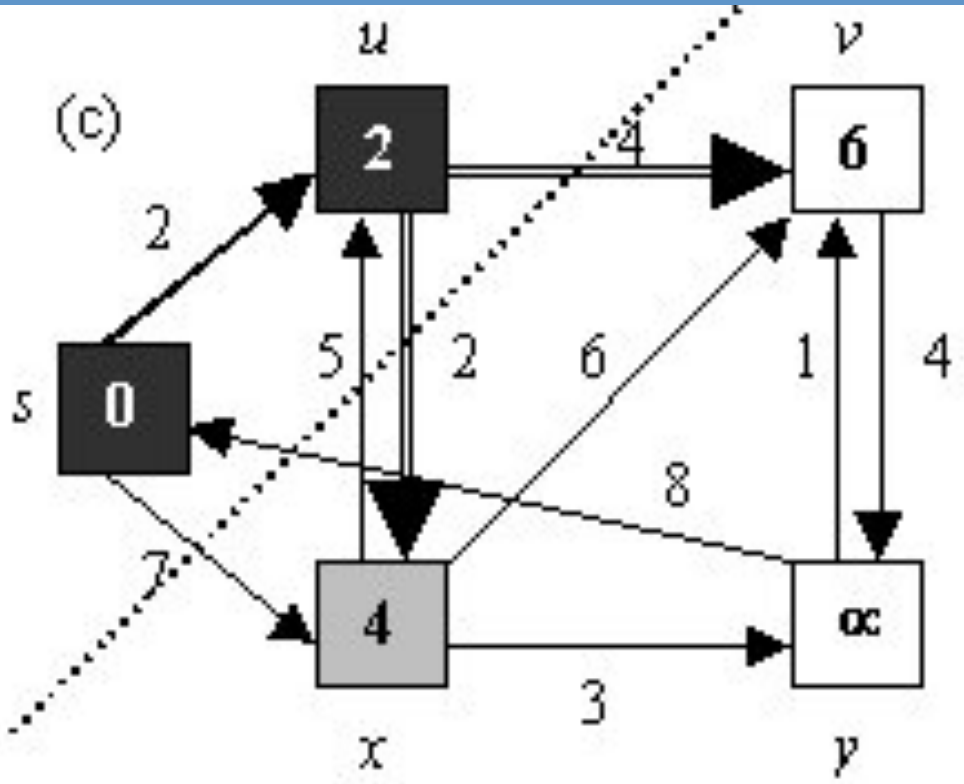
dist = { , **2** , ∞ , 7 , ∞ }

prev = { s , s , , s , }

Dijkstra algorithm from vertex s

Round 2:

- choose u (node with least dist[] in A)
- update dist[] of nodes adjacent to s *if applicable*

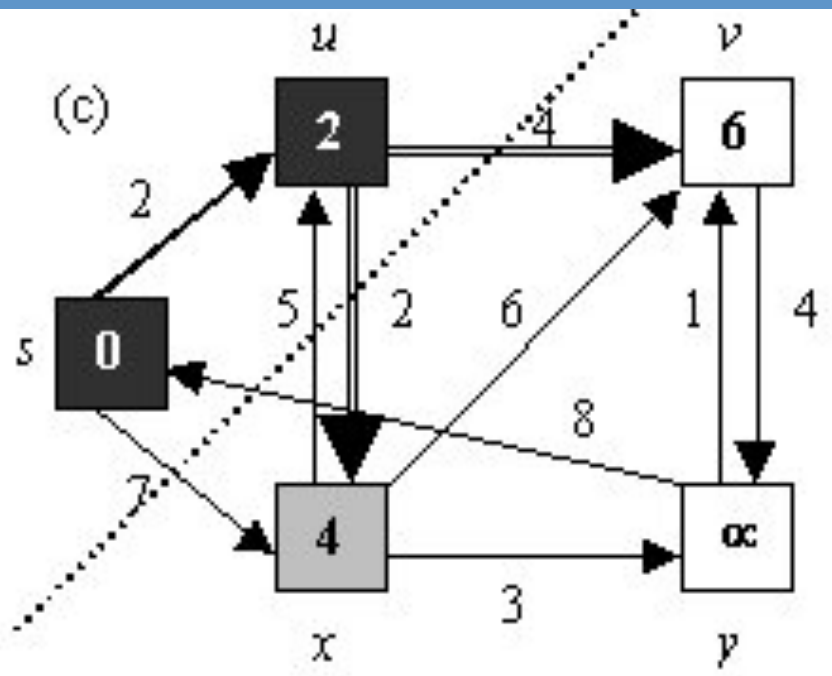


B= { s, u}
{0, 2}
{ nil, s }

A = { , , v , x , y }
dist = { , , 6 , 7→**4** , ∞ }
prev = { s, s , u, u , N }

Dijkstra algorithm from vertex s

	s	u	v	x	y
0	0 s	∞ N	∞ N	∞ N	∞ N
1		2 s	∞ N	7 s	∞ N
2			6 u	**4** u	∞ N



B= { s, u}

{0, 2}

{ nil, s }

A = { , , v , x , y }

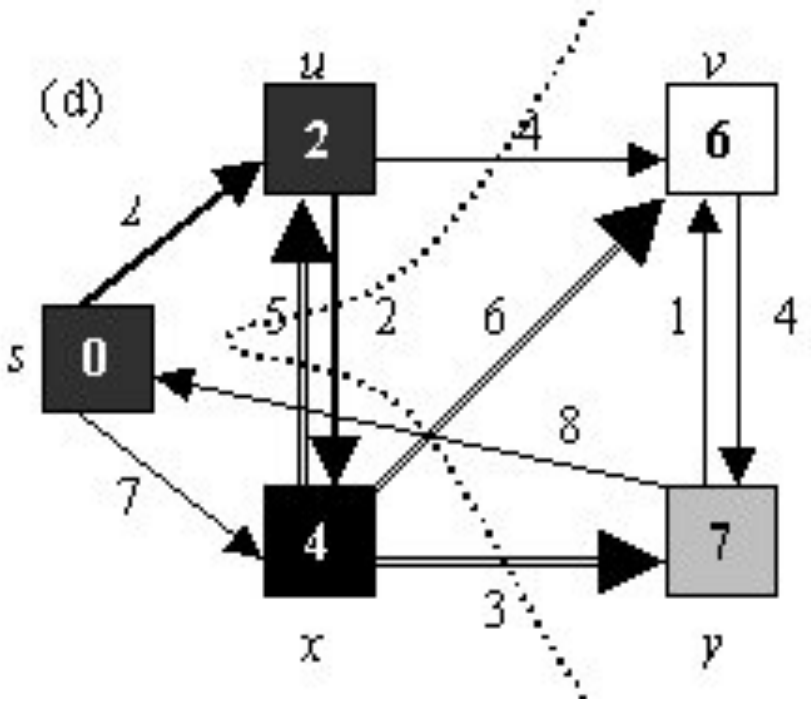
dist = { , , 6 , 7→**4** , ∞ }

prev = { s, s , u, u , N }

Dijkstra algorithm from vertex s

Round 3

	s	u	v	x	y
0	0 s	∞ N	∞ N	∞ N	∞ N
1		2 s	∞ N	7 s	∞ N
2			6 u	**4** u	∞ N
3			**6** u		7 x



B= { s, u, x }

{0, 2, 4 }

{ nil, s, u }

A = { , , v , , y }

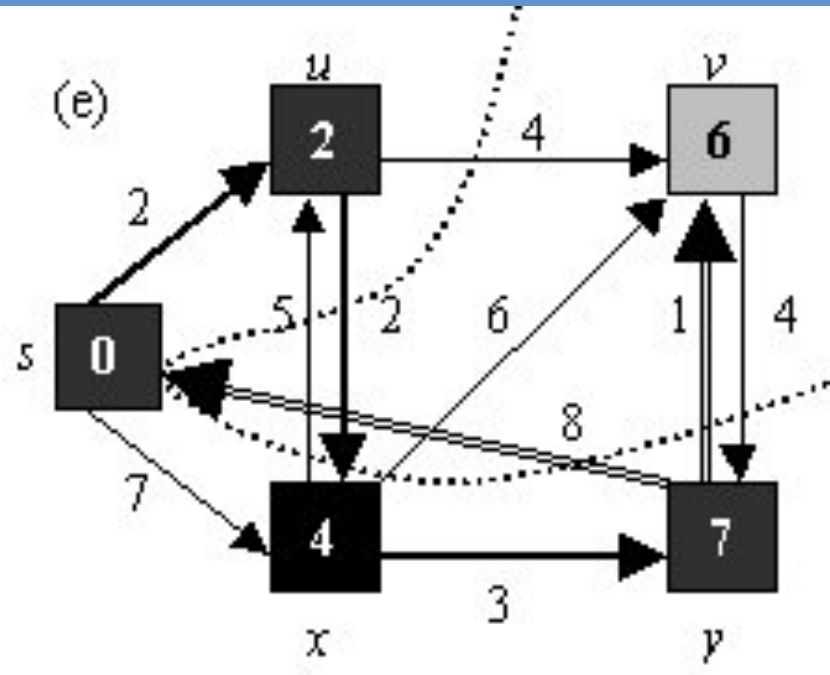
dist = { , , **6** , , 7 }

prev = { , , u , , x }

Dijkstra algorithm from vertex s

Round 4

	s	u	v	x	y
0	0 s	∞ N	∞ N	∞ N	∞ N
1		2 s	∞ N	7 s	∞ N
2			6 u	**4** u	∞ N
3			**6** u		7 x
4					**7** x



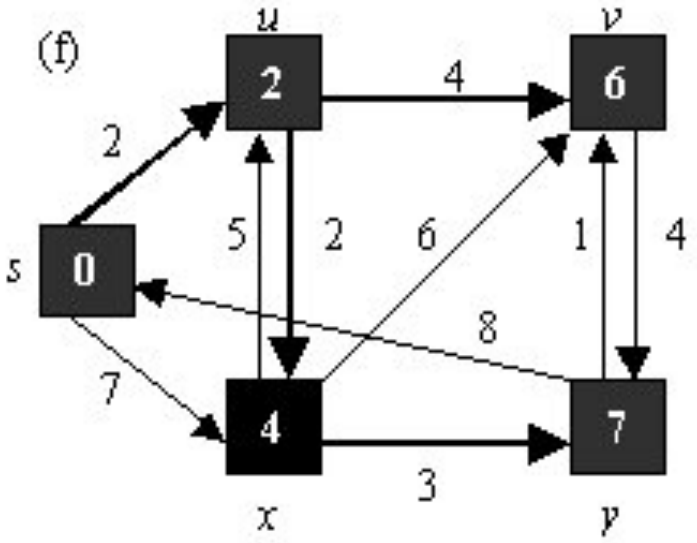
B= { s, u, x, v}

A = { y }

Dijkstra algorithm from vertex s

Round 5

	s	u	v	x	y
0	0 s	∞ N	∞ N	∞ N	∞ N
1		2 s	∞ N	7 s	∞ N
2			6 u	**4** u	∞ N
3			**6** u		7 x
4					7 x
5					



B= { s, u, x, v, y} A = { }

Question: what is the shortest path from s to y?

Dijkstra's algorithm [conceptual only]

Purpose: Find shortest path from vertex s

set dist[u] = ∞, pred[u]=nil for all u, then dist[s]= 0;

set B= ∅, A= V. B is set of vertices where shorted path from s found, A is set of other vertices.

```
while (A is not empty):
    select u from A such that dist[u] is smallest
    remove u from A and add it to B
    for all (u,v) in G:
        if (dist[v] > dist[u]+w(u,v): update
        dist[v] and pred[v]
```

Q: How to represent A?

Dijkstra's algorithm [conceptual only]

```
set dist[u]= ∞, pred[u]=nil for all u, then dist[s]= 0;
set B= ∅, A= makePQ(V)
while (A not empty)
   u= deleteMin(A)
   add u to B  // practice: just mark u as visited
   for all (u,v) in G:
      if (dist[v] > dist[u]+w(u,v):
         update dist[v] and pred[v]
```

Practical note: "update dist[v] and pred[v]" means

```
dist[v]= dist[u]+w(u,v) ;   pred[v]= u
```

decrease weight of v in PQ to dist[v], hence, need to locate v in PQ, change weight and upheap

Q10.1: For a directed graph with the following edges:

Data	**For a directed graph with the edges listed in LHS:**
a b 3 **a d 7** **b d 2** **c e 6** **d b 2** **d c 5** **d e 4** **e d 2**	1. Draw a weighted directed graph that reflects these edges and weights (logical representation). 2. Construct an adjacency matrix for the weighted digraph you have just drawn, including the weights. Be explicit about how you are going to handle matrix cells for which there is no information in the data. 3. Run through Dijkstra's Algorithm starting from the vertex a.

Lab

Implement priority queue (see LMS)

- you can use the heap implementation of previous weeks

Assignment 2:

to fill in code within `puzzle.c`

- Write functions at FILL WITH YOUR CODE
- You might want to add to existed data structures, see comments in `typedef struct node`
- You might want to add some additional functions

Interface:

```
make
./15puzzle 1.puzzle
```