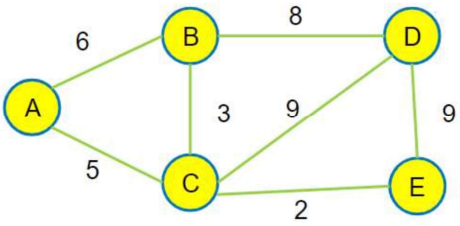
PASS Session

Tuesday, 31 May, 2022 19:39

Minimum Spanning Tree (Prim's)

Fundamentals

- Greedy Algorithm (choose local optimal)
- Like Dijkstra (choose vertex with shortest distance)



1. Update adjacent vertex and distance to respective position of array *(if non-inf, compare and get minimum value and store vertex accordingly)*
2. Choose closest vertex
3. Repeat Steps 1 and 2
4. If same distance? No need to update

Modifications

- Modify distance calculation

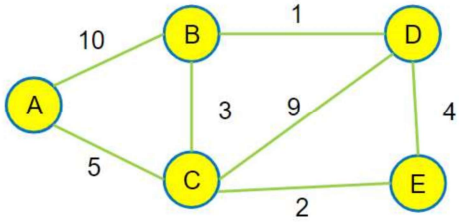
Complexities

- Time → O(E log V)

Minimum Spanning Tree (Kruskal's)

Fundamentals

- Trees are connected by edges. So? We can merge vertices together to form a tree!
- However, we do not merge vertices that fall under same subtree, because a tree has no cycle.



1. List all edge weights and sort them
2. Connect each pair of vertices one by one, while preventing a creation of a cycle
3. Repeat steps 1 and 2.

Union-Find Implementation (Only need to roughly know for Exam)

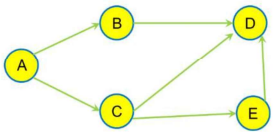
- Sorting Edges (Quick Sort) → O(E log V)
- Find (Check if u and v in same tree) → O(1)
- Union (joint if not same set) → O(V)
- Total → O(EV) but amortizing it gives you O(E log V)

Directed Acyclic Graph – Topological Sort

Fundamentals

- For edge <A, B>, A comes first, then B, **always**
- Topological Sort → if edge exists, <vertex that appears first, vertex that appears then> (also vertex that appears first < vertex that appears then). If not, vertices same order

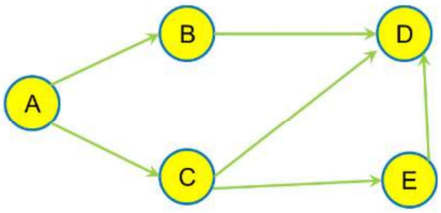
- Which one of these are not a valid topological sort of the DAG?
  1. A, B, C, E, D
  2. A ,C, B, E, D
  3. A, C, E, B, D
  4. A, B, E, C, D



Directed Acyclic Graph – Kahn's

Fundamentals

- Start with vertices without incoming edges (i.e. edges pointing to the vertex)
- Delete all outgoing edges from vertex (i.e. edges pointing from the vertex)
- Add in vertices without incoming edges.
- Repeat



- After removing, if vertex it connects to have incoming edges, no putting vertex under process
- Can use either Stack or Queue. Why?

Complexity

- Time and Space → O(V+E)

Why are these important?

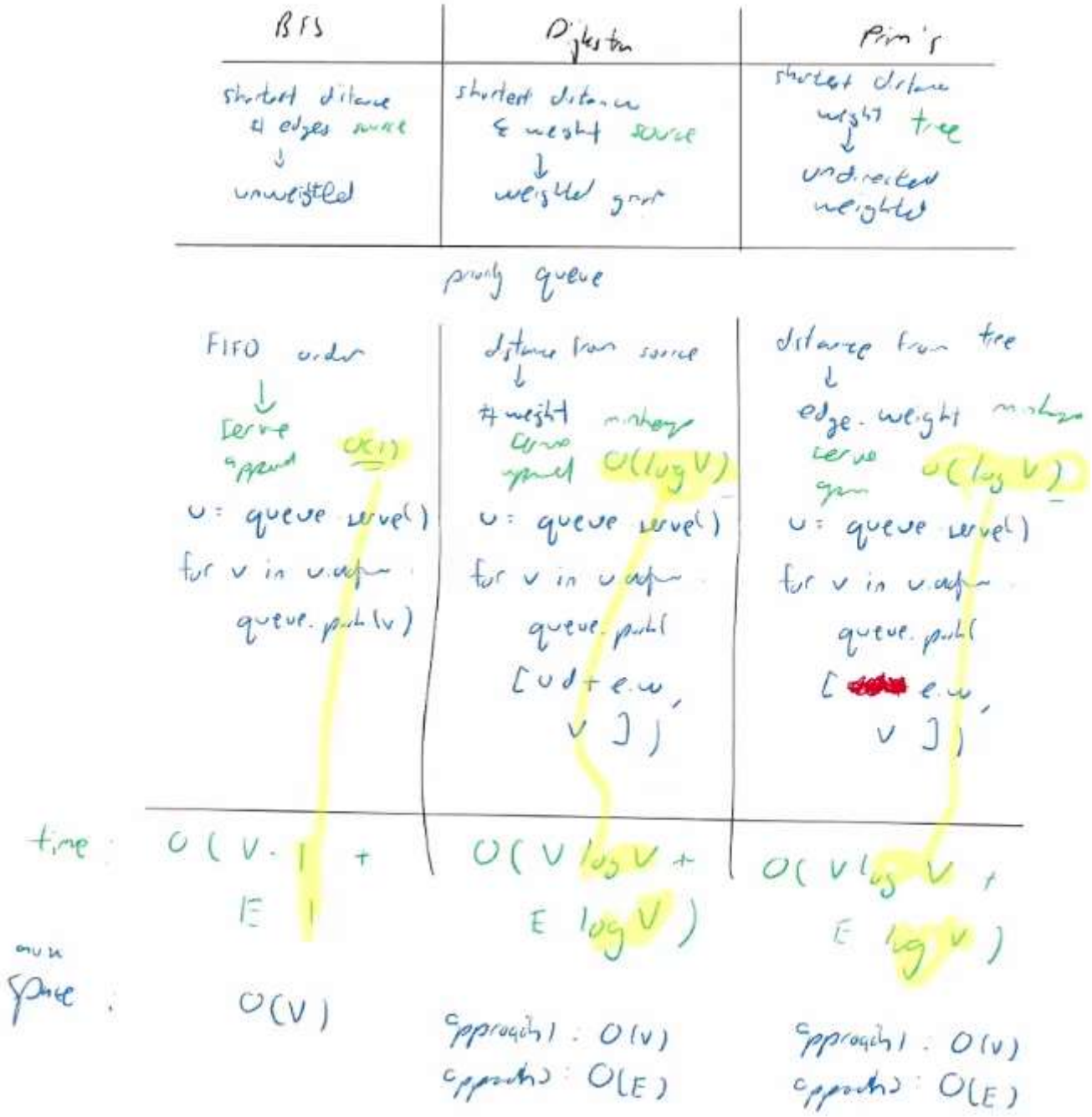
- **Minimum Spanning Tree** → Used for network diagrams (i.e. telephone or cable networks)
- **Directed Acyclic Graph** → Progression based questions can be tackled with the algorithms implemented in this manner

What's Next?

- **Dynamic Programming**

Sanity Check Summary

Wednesday, 1 June, 2022 02:55



Prim's Algorithm
Growing of MST

MONASH University

- So take Dijkstra
  - Modify the distance update/ calculation for edge <u,v,w>
    - Instead of v.distance = u.distance + w
    - Change to v.distance = w
    - Perform relaxation only if distance is smaller
- So what is the complexity?
  - Same as Dijkstra O(V log V + E log V)
  - Thus O(E log V)

Kruskal's Algorithm
Combining (Union of) Trees

MONASH University

- But how do we implement it?
  - Take the list of edges and sort.
    - Easy… just use quicksort
    - This is O(E log E)
  - Check if vertex u and vertex v in <u,v,w> is in the same tree (Find)
    - We use set! Any set data structure
      - Built-in Python Set (which is based on Dictionary/ Hashtable)
      - Disjoint-Set (which you learn in FIT3155)
    - This is O(1)
  - If not the same set, you joint them with the edge (Union)
    - This is O(V) for now
  - Thus, known as Union-Find
  - Complexity? O(E log E + E(1+V)) = O(EV)

For each edge

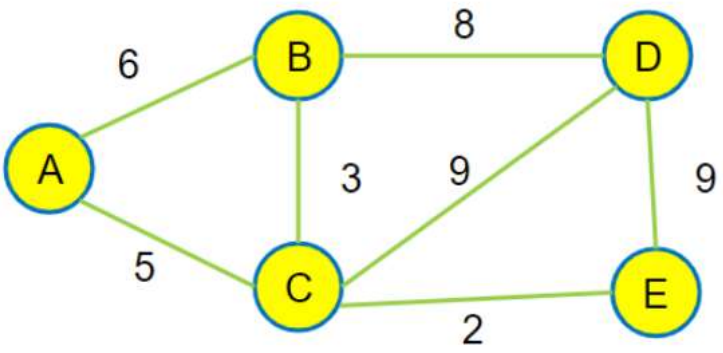
Minimum Spanning Tree

Tuesday, 31 May, 2022 19:40

Minimum Spanning Tree (Prim's)

Fundamentals

- Greedy Algorithm (choose local optimal)
- Like Dijkstra (choose vertex with shortest distance)



1. Update adjacent vertex and distance to respective position of array *(if non-inf, compare and get minimum value and store vertex accordingly)*
2. Choose closest vertex
3. Repeat Steps 1 and 2
4. If same distance? No need to update

Modifications

- Modify distance calculation

Complexities

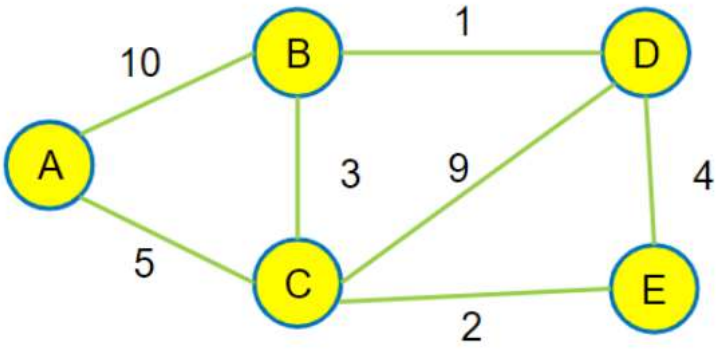
- Time → O(E log V)

Basically staring from a vertex and slowly adds more vertex to the tree.

Minimum Spanning Tree (Kruskal's)

Fundamentals

- Trees are connected by edges. So? We can merge vertices together to form a tree!
- However, we do not merge vertices that fall under same subtree, because a tree has no cycle.



1. List all edge weights and sort them
2. Connect each pair of vertices one by one, while preventing a creation of a cycle
3. Repeat steps 1 and 2.

Union-Find Implementation (Only need to roughly know for Exam)

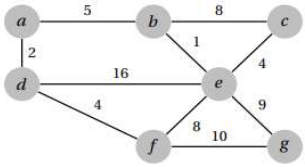
- Sorting Edges (Quick Sort) → O(E log V)
- Find (Check if u and v in same tree) → O(1)
- Union (joint if not same set) → O(V)
- Total → O(EV) but amortizing it gives you O(E log V)

For Kruskal it joins different tree together.

Studio06 Q2

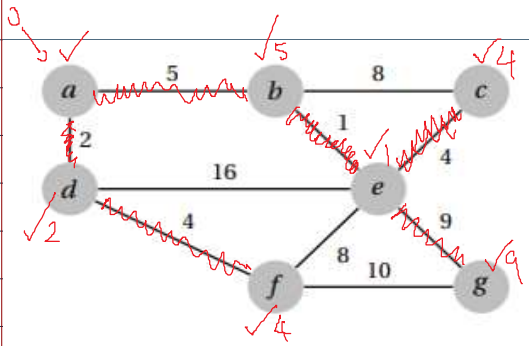
Sunday, 3 April, 2022 11:50

Problem 2. Show the steps taken by Prim's and Kruskal's algorithms for computing a minimum spanning tree of the following graph. Use vertex *a* as the root vertex for Prim's algorithm. Make sure that you indicate the order in which edges are selected, not just the final answer.



Prim —— growing tree similar dijkstra

Kruskal —— union-find



Prim's —> tree
list of edges

① 2ad — 5ab

② 4df — 16de, 5ab

③ 5ab — 10fg, 8fe — 16de

④ 1be — 8bc, 8fe — 16de, 10fg

⑤ 4ec — 8fe, 9eg — 16de, 8bc, 10fg

⑥ 8fe — 8bc, 9eg — 10fg

Same but doesn't append

⑦ 8bc — 10fg, 9eg — 10fg

⑧ 9eg — 16de, 10fg

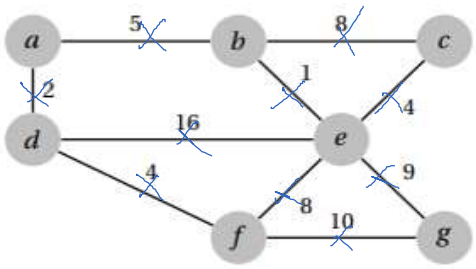
V = 7

tree = V − 1

6 edges

terminate

[ad, df, ab, be, ec, eg]



Kruskal

↓

union-find by size

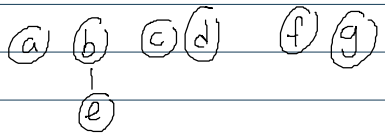
	a	b	c	d	e	f	g
Parent	-1	-1	-1	-1	-1	-1	-1

1be ✓
2ad ✓
4df ✓
4ec ✓
5ab X
8bc X
8de X
9eg ✓
10fg
16de

tree minimale

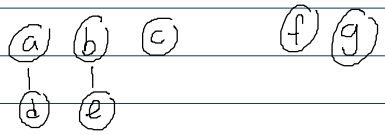
1be

	0	1	2	3	4	5	6
	a	b	c	d	e	f	g
Parent	-1	-1	-1	-1	1	-1	-1



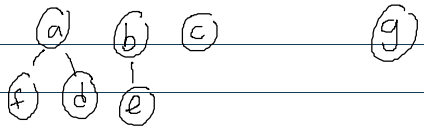
2ad

	0	1	2	3	4	5	6
	a	b	c	d	e	f	g
Parent	-2	-2	-1	0	1	-1	-1



4df

	0	1	2	3	4	5	6
	a	b	c	d	e	f	g
Parent	-3	-2	-1	0	1	0	-1

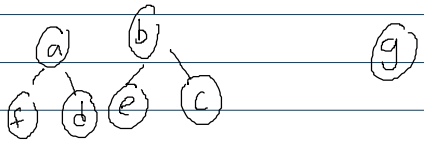


4ec

	0	1	2	3	4	5	6
	a	b	c	d	e	f	g
Parent	-5	-3	1	0	1	0	-1

b -2

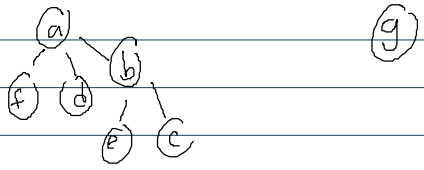
c



5ab

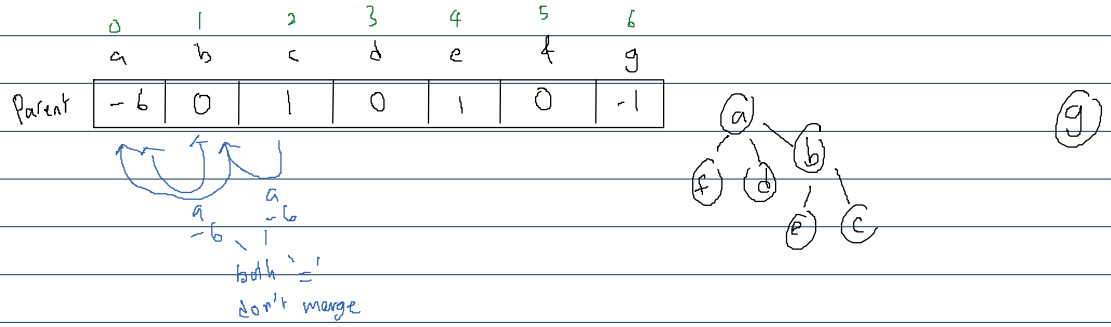
	0	1	2	3	4	5	6
	a	b	c	d	e	f	g
Parent	-6	0	1	0	1	0	-1

a -3 b -3

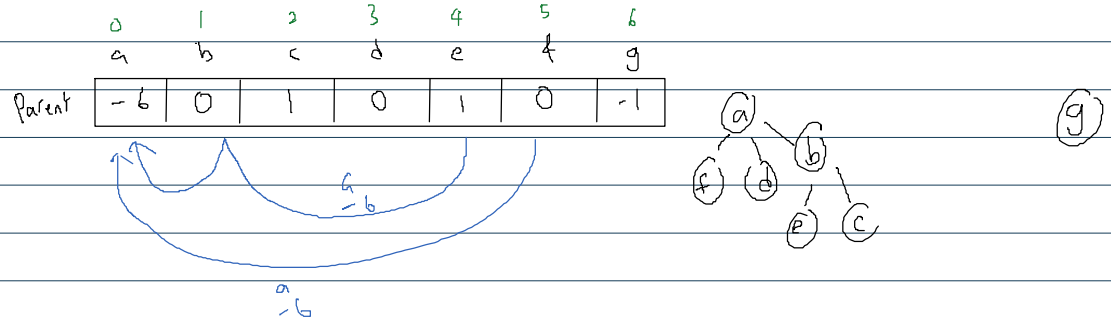


8bc

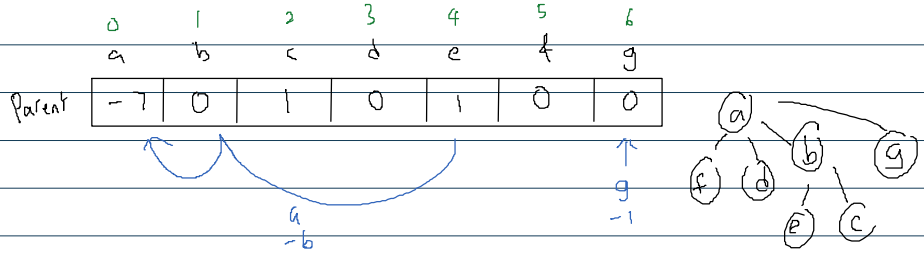
8bc



8fe



9eg



q = -7, tree = 7 node

v = 7

terminate

∴ be, ad, df, ec, ab, eg