EECS 233 SI Session
Leader: Bertram Su
November 14, 2019

Objectives:
Upon completion of this SI session, participants will be able to:

1. Recognize the big O expression of merge sort
2. Determine how arrays will look after cycles of merge sort

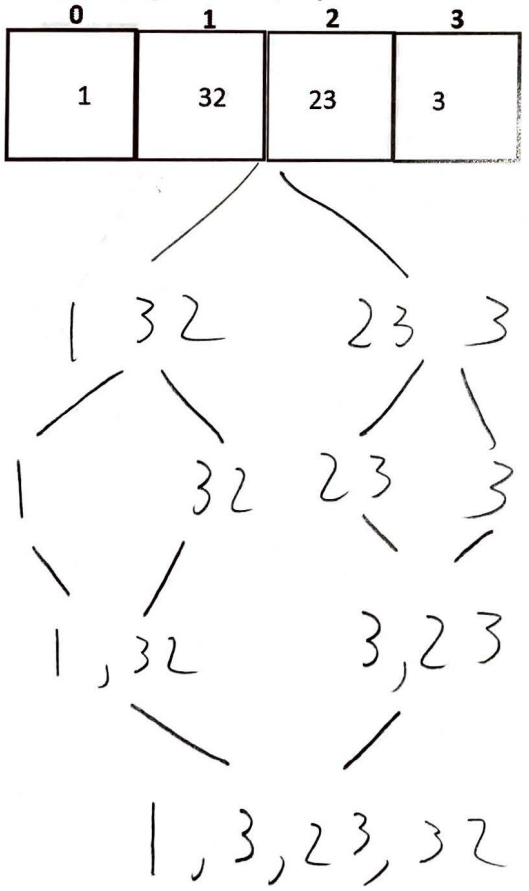
Foundations:

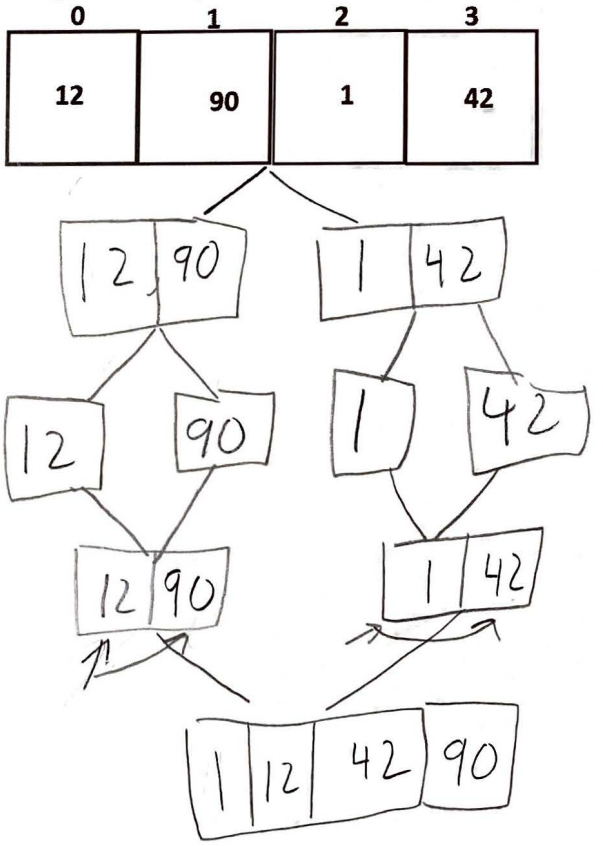
1. What are the steps for merge sort? Words: sort, merging, ~~divide~~, merging

1) divide array into left and right subarrays until size 1
2) While merging, sort
3) continue merging until sorted

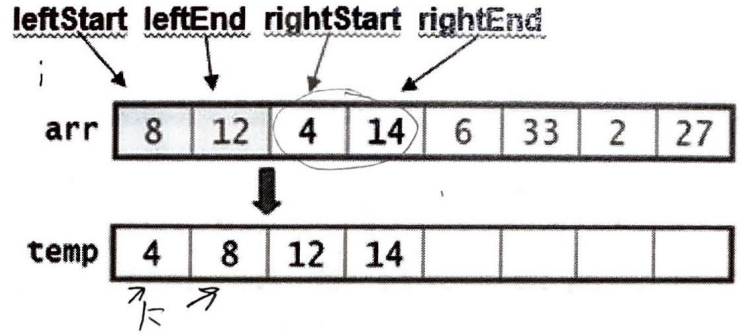
Exercises:

2. Using a merge sort where you create a new array at each step, draw the phases of each array.





3. Finish the following merge code for merge sort with only two arrays.



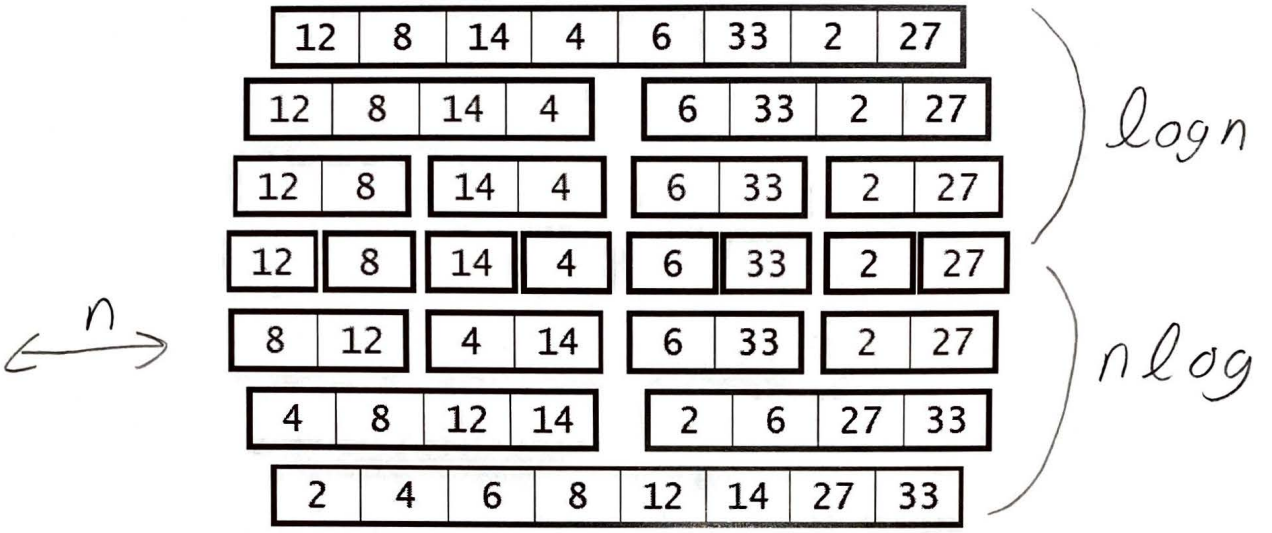

```
static void merge(int[] arr, int[] temp, int leftStart, int leftEnd, int rightStart, int rightEnd)
    {
      int i = leftStart; // index into left subarray
      int j = rightStart; // index into right subarray
      int k = leftStart; // index into temp
      while ( i <= leftEnd && j <= rightEnd ) {
            if(arr[i] <= arr[j]){
               temp[ k ] = arr[ i ];
               i++ ;
            }
            else{
               temp[k] = arr[j];
               j++;
            }
            k ++;
      }
       /* Copy remaining elements of left array if any */
          while (i <= leftEnd )
          {
            temp[k] = arr[i];
            i ++;
            k ++;
          }

        /* Assume we copy the code from above but for right subarray*/

       for (i = leftStart; i <= rightEnd; i++)  // copy back
            arr[i] = temp[i];

}
```

4. Let's figure out the Big O of merge sort intuitively.
Hint: How many levels get merged?



Big O: nlogn

Best: nlogn

Summary

5. What is the best case and worst case of merge sort for the implementation we talked about in class?

6. Given the following conditions, what sorting method would you use?
   a. The data is mostly sorted, but there are a few out of place elements
   b. The data is random, shifting is expensive, and you have extremely limited memory.
   c. The data is random and you have practically unlimited space in memory.

a) insertion , maybe shell

b) quicksort

c) merge

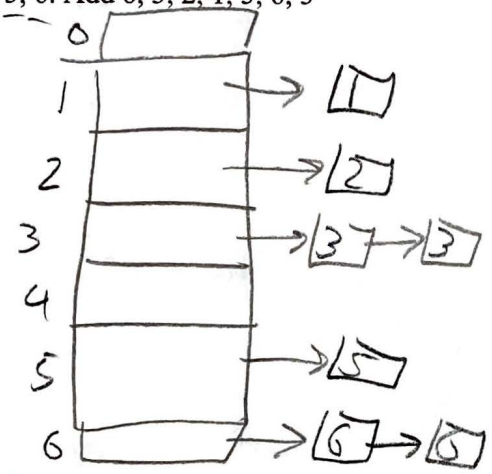
EECS 233 SI Session
Leader: Bertram Su
November 17, 2019

Objectives:
Upon completion of this SI session, participants will be able to:

1. Recognize Graph terminology

Exercises:

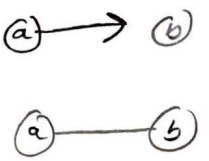
1. Using bucket sort, determine where the following numbers would go with buckets 1, 2, 3, 4, 5, 6. Add 6, 3, 2, 1, 5, 6, 3



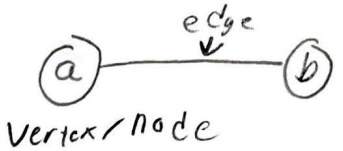
1, 2, 3, 3, 5, 6, 6

2. Draw a graph with a cycle

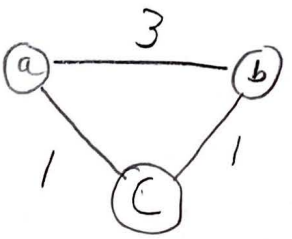
directed



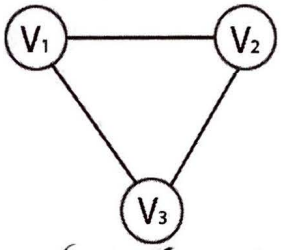
3. What is a vertex/node? What is an edge? Draw them



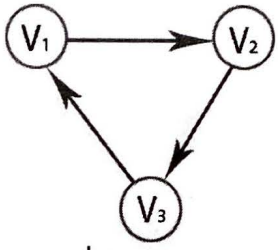
4. What is a weight?



5. Label the following graphs as directed or undirected graphs.

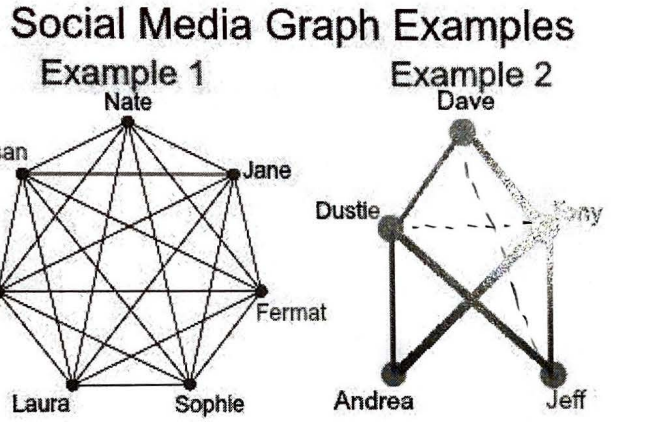


Undirected



directed

6. Which of the following is a complete graph? Which is only a connected graph?



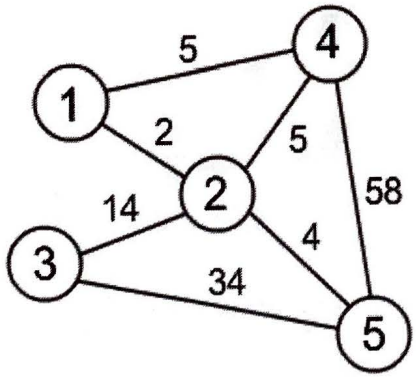
There is an edge directly to all other vertices for each vertex

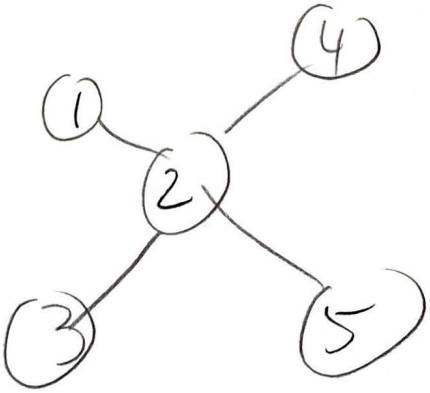
Complete

Connected

7. List the neighbors of 5. Draw a spanning tree

N: 2, 3, 4





also others

8. Are the given graphs acyclic?

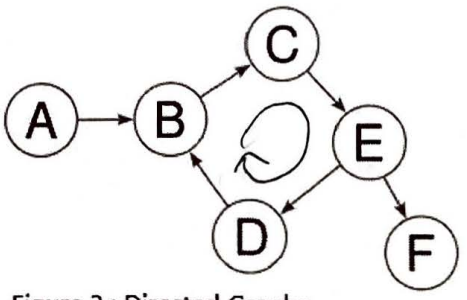
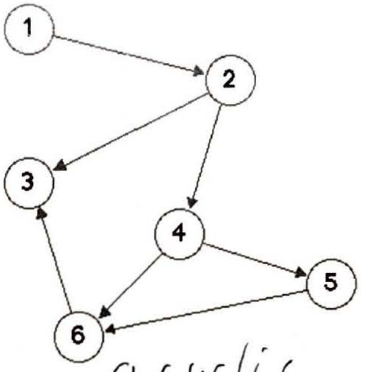


Figure 3 : Directed Graphs

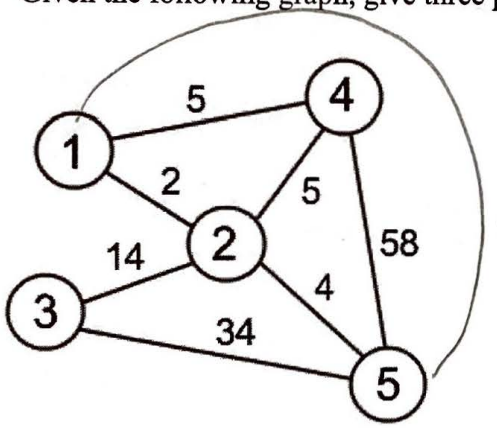
not



acyclic

Summary

9. Given the following graph, give three paths from 1 to 4.



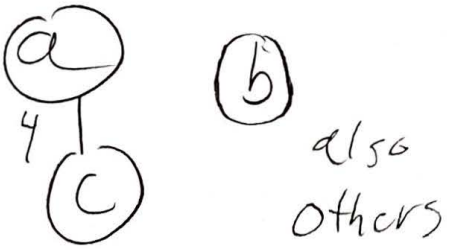
5

1 → 4

1 → 2 → 3 → 5 → 4

1 → 2 → 4

10. Draw a graph that is not connected but has three vertices a, b, and c that contain a path from a to c that costs 4.



also others

Upcoming Events and Suggestions for Further Study:

Events: thurs

- Next SI session is ~~Wednesday~~ from 6:00 to 8:00 at Olin 313

Further Study:

- https://cwru.az1.qualtrics.com/jfe/form/SV_1Th0sizrbealYXz
  - Survey link
- In class lecture slides

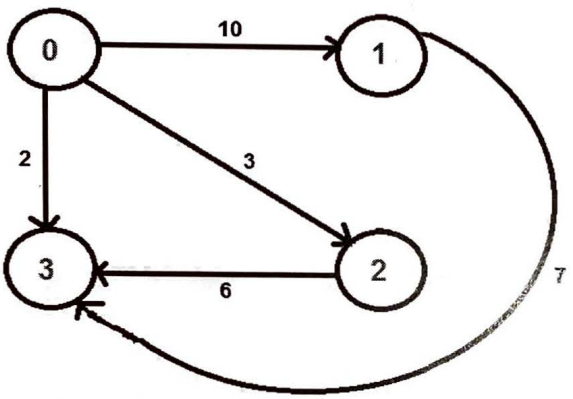
EECS 233 SI Session
Leader: Bertram Su
November 21, 2019

Objectives:
Upon completion of this SI session, participants will be able to:

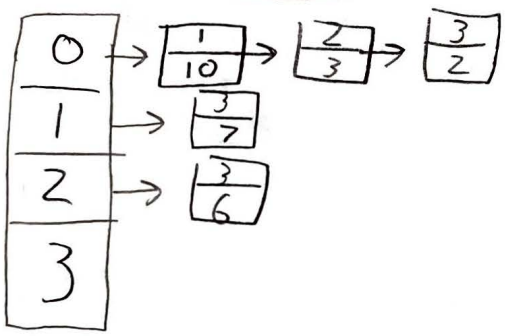
1. Determine how graphs look in matrix/list form
2. Traverse through a graph using breadth and depth strategies

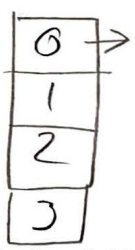
Foundations:

1. Draw the following in adjacency matrix (2D array) and adjacency list (LL) form.



	0	1	2	3
0		10	3	2
1				7
2				6
3				





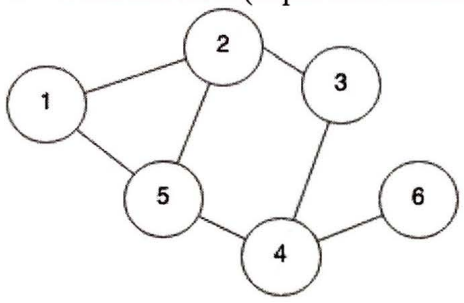
2. What are the steps for depth first search? Words: process, visited, neighbors
a) Choose a starting vertex
b) Mark vertex as visited and process it
c) For all neighbors that aren't visited, recurse DFS on each (Choose neighbors in order of lowest to highest for convenience)

process = print / if ( )

3. What are the steps for breadth first search? Words: queue, mark, queue
a) Process starting vertex, mark, and place it in a queue
b) Remove a vertex, v, from the front of the queue, process ✓
c) For each unmarked neighbor u of v, ~~process u~~, mark u, and then place u in the queue (since u may have further processed neighbors)
d) Repeat b

Exercises:

4. Write the DFS (depth first search) and BFS (breadth first search) for the following graph

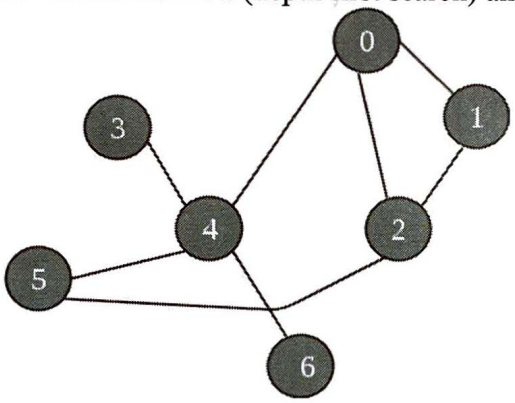


DFS: stack: ~~1, 2, 3, 4, 5, 6~~
visited: 1, 2, 3, 4, 5, 6
print: 1, 2, 3, 4, 5, 6

BFS: Q: ~~1, 2, 5, 3, 4, 6~~
marked: 1, 2, 5, 3, 4, 6
print: 1, 2, 5, 3, 4, 6

BFS: Q: ~~1, 2, 5~~, 3, 4
marked: 1, 2, 5, 3, 4
print: 1, 2, 5

5. Write the DFS (depth first search) and BFS (breadth first search) for the following graph



DFS 0, 1, 2, 5, 4, 3, 6

BFS: 0, 1, 2, 4, 5, 3, 6

6. Your coworker is on vacation and he doesn't name methods well. Determine what type of traversal this is and tell me how you know.

```
public void hawaii(int i, int parent){
        System.out.println(vertices[i].id);
        verticies[i].encountered = true;
        verticies[i].parent = parent;
        Iterator<Edge> edgeItr = edges.iterator();
        while(edgeItr.hasNext()){
                Edge curEdge = edgeItr.next();
                j = curEdge.endNode;
                if(vertices[j].encountered == false)
                        hawaii(j, i);
        }
}
```

DFT

recursion

7. The following is Pseudocode that Dr. Ayday gave us in class. Fill in the missing pseudocode

```
Trav(origin)
    origin.parent = null
    create a new Queue q
    q.insert(origin)

    while(!q.isEmpty())
        v = q.remove()
        process v

        for each vertex w in v's neighbors
            if w ! marked
                mark w as encountered
                w.parent = v
                q.insert(w)
```

Summary

8. Is it possible to implement DFS and BFS without recursion?

yes

9. What type of a graph would work well with an adjacency list implementation?

Sparse

10. What is the Big O of BFS when we use arrays?

~~O(V+E)~~ $O(V^2)$

11. What is the Big O of DFS when we use arrays?

~~O(V+E)~~ $= O(V^2)$

Upcoming Events and Suggestions for Further Study:

Events:

- Next SI session is Wednesday from 6:00 to 8:00 at Sears 336

Further Study:

- https://cwru.az1.qualtrics.com/jfe/form/SV_1Th0sizrbealYXz
  - Survey link
- https://www.hackerearth.com/practice/algorithms/graphs/breadth-first-search/tutorial/
  - This contains a connection between the BFS of trees and graphs
- https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
  - A topic just for graph DFS
- All the websites above have their DFS equivalent

EECS 233 SI Session
Leader: Bertram Su
April 10, 2019

Objectives:
Upon completion of this SI session, participants will be able to:

1. Find the shortest path through a graph using Djikastra's algorithm
2. Read/Create Dijkstra tables

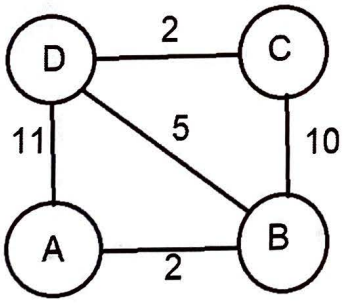
Foundations:
1. Comment the steps of Dijkstra's algorithm

Choose a starting vertex A // begin vertex
Create an array D[v] to keep track of the distance from A to v // min distance of A to V
Create an array P[v] to keep track of the parent of v // holds parent of vertex V
Create a set N to keep track of nodes with least cost path known // least cost path known

For all vertex v
    D[v] = infinity
    P[v] = null

//The important part starts here
until all nodes are in N or we reach our target vertex // until visited all or we visited our target
    find w not in N with smallest D[] value // find vertex w that we haven't visited with smallest distance from A
    add w to N // visited
    For each neighbor n not in N
        if D[w] + c(w, n) < D[n] // update
            D[n] = D[w] + c(w, n) distance if we found a smaller one
            P[n] = w update parent

2. Determine the shortest path from A to C



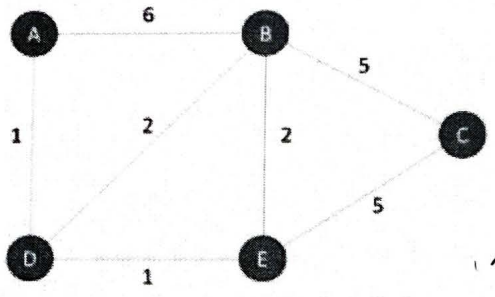
N	D[B], P[B]	D[C], P[C]	D ✓
A	2, A	∞, null	11, A
A, B		12, B	7, B
A, B, D		9, D	
A, B, D, C			

C, D, B, A

A → B → D → C
9

Exercises:

3. The president has tasked you with finding the shortest flight time from city A to all these individual cities in different trips that all start from A



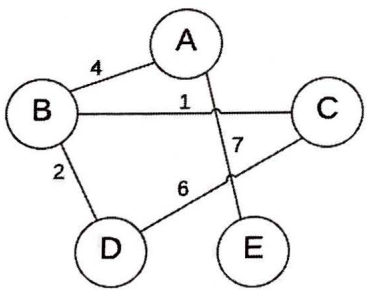
N	B ✓	C ✓	D ✓	E ✓
A	6, A	∞, null	1, A	∞, null
A, D	3, D	∞, null		2, D
A, D, E	3, D	7, E		
A, D, E, B		7, E		
A, D, E, B, C				

Using your chart, what would be the shortest flight time from A to B? What's the path?

3

pop → A → D → B

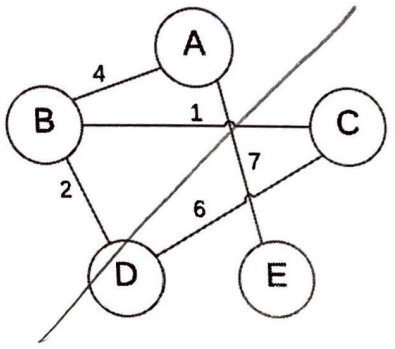
4. Determine the shortest path from A to C.



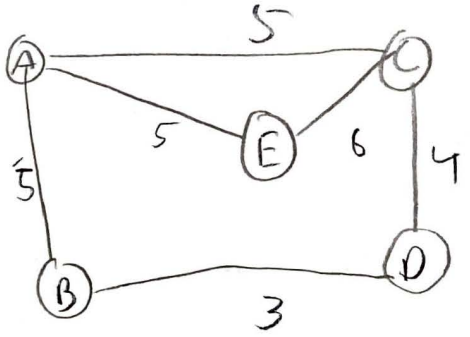
A → B → C

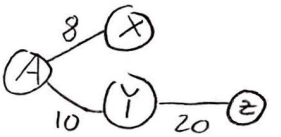
Summary

5. Determine the shortest path for all the vertexes. Start from A



N	B ✓	C ✓	D	E ✓
A	5, A	5, A	∞	5, A
A, B		5, A	8, B	5, A
A, B, C			8, B	5, A
A, B, C, E			8, B	
A, B, C, E, D				





6. Given the following chart, what was the total cost of the path of a to z?

Step	n	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	a	8, a	10, a	infinite
1	a, x		10, a	infinite
2	y, a, x			30, y
3	z, y, a, x			

30

7. From the chart above, what was the path from a to z?

z → Y → A => A → Y → z

Upcoming Events and Suggestions for Further Study:

Events:

- Next SI session is Sunday from 1:00 to 2:30 at Sears 336

Further Study:

- https://cwru.az1.qualtrics.com/jfe/form/SV_1Th0sizrbealYXz
  - Survey Link
- https://www.youtube.com/watch?v=eFZCPlZCyIM
  - If you're more inclined to videos, here's a nice video tutorial