

CS280 – Even More Graphs

March 28, 2016

<http://azrael.digipen.edu/~mmead/www/Courses/CS280/Graphs-1.html>

Homework Tips

- Make sure and use Count + 1 when checking load factor, since we haven't grown table

Graphs

- We can merge the second step of finding the shortest path into the loop

* Arbitrarily starting at A

Node	Init	1	2	3	4	5	6
A	0*	0*	0*	0*	0*	0*	0*
B	∞	13(A)	13(A)	13(A)*	13(A)*	13(A)*	13(A)*
C	∞	∞	∞	33(AB)	30(ABD)	30(ABD)*	30(ABD)*
D	∞	29(A)	26(AF)	25(AB)	25(AB)*	25(AB)*	25(AB)*
E	∞	∞	35(AE)	35(AF)	35(AF)	34(ABDC)	34(ABDC)*
F	∞	11(A)*	11(A)*	11(A)*	11(A)*	11(A)*	11(A)*

* We have found cheapest route

* cast-1 to unsigned to get " ∞ "

PQ

1. F, B, D, 29	4. D, 26, D, 29, E, 30, C, 33, E, 35
2. B, 13, D, 26, D, 29, E, 35	5. C, 33, E, 34, E, 35
3. D, 25, D, 26, D, 29, C, 33, E, 35	

- We are expected to make our own classes
 - Node
 - List
 - Overload *operator<* for our priority queue to make life easy
 - We might have directed and undirected graphs, nodes that have no neighbors, etc...
 - If we have a tie, go to lower node value
 - Assignment due on SUNDAY!
 - *AddUEdge* will just call *AddDEdge*
 - GraphViz is part of Dpxygen you can install
 - Open source to draw all sorts of data structures
 - We will need to derive myStack/myQueue so that interface will work for both types of graphs
 - The algorithm doesn't change for directed/undirected since we receive the same data structure.
- Just a convenience for the driver
- Start on assignment today
 - Algorithm depends on underlying data structure
 - Priority queue is implemented on the heap
 - We can either sort as we build the graph and then run Dijkstra's, or just push back graph, do one big sort, then run Dijkstra's
 - Using pipes in code to send output to console window
 - This is how visual studio sends compiler output to the console