95

CMPE 130 Midterm Exam #2 Fall 2014

15:00—16:15 Thursday Nov. 6, 2014

Student Name [redacted] **(print)**

Student ID [redacted]

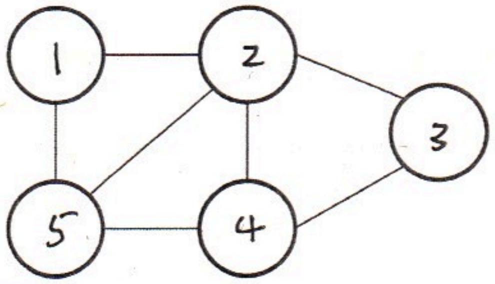
Problem 1: Consider inserting keys 10,22, 31, 4, 15, 28, 17, 88, 59 into hash table of length m=11 using open addressing with the auxiliary hash function h(k)=k. Illustrate the result of inserting these keys using

(a) **Linear probing (6 points)**
(b) **Quadratic probing using h(k, i)=h(k)+ i +3 i**2 (7 points)**
(c) **Double hashing with h1(k)=k and h2(k)=1 + (k mod(m-1)) (7 points)**

✓ **Problem 2: For the set of {1,4, 5, 10, 16, 17, 21} of keys, draw binary search trees of height 2, 3, 4, 5 and 6. (10 points)**

Problem 3: Write the adjacency-matrix representation of Graph (a) and Graph (b) below. (10 points)

OK
See textbook


Graph (a)

a)

$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{5}$

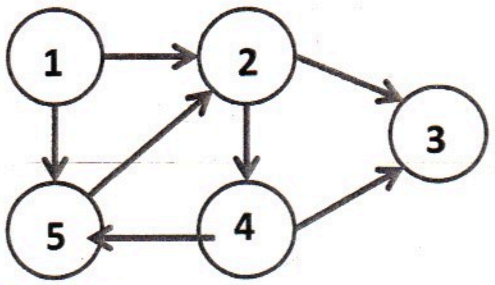
$\boxed{2} \rightarrow \boxed{1} \rightarrow \boxed{3} \rightarrow \boxed{4} \rightarrow \boxed{5}$

$\boxed{3} \rightarrow \boxed{2} \rightarrow \boxed{4}$

$\boxed{4} \rightarrow \boxed{2} \rightarrow \boxed{3} \rightarrow \boxed{5}$

$\boxed{5} \rightarrow \boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{4}$

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0


Graph (b)

b)

$\boxed{1} \rightarrow \boxed{2} \rightarrow \boxed{5}$

$\boxed{2} \rightarrow \boxed{3} \rightarrow \boxed{4}$

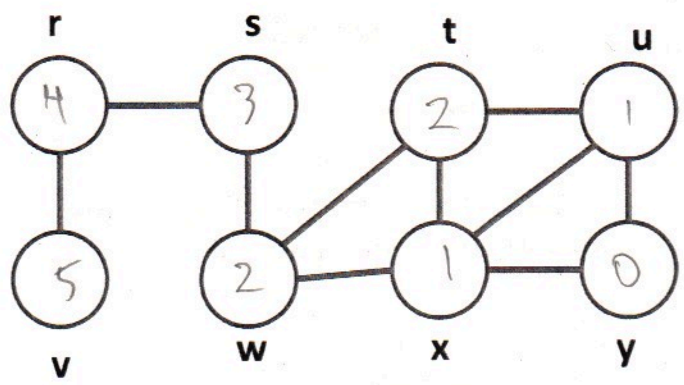
$\boxed{3} \rightarrow$

$\boxed{4} \rightarrow \boxed{3} \rightarrow \boxed{5}$

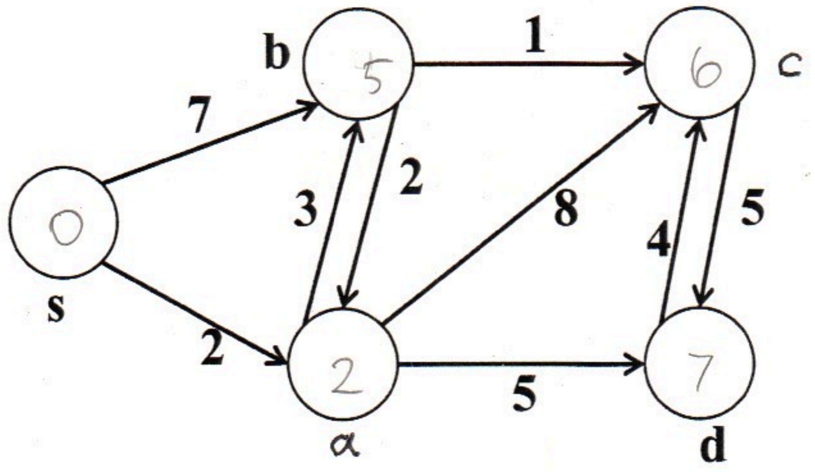
$\boxed{5} \rightarrow \boxed{2}$

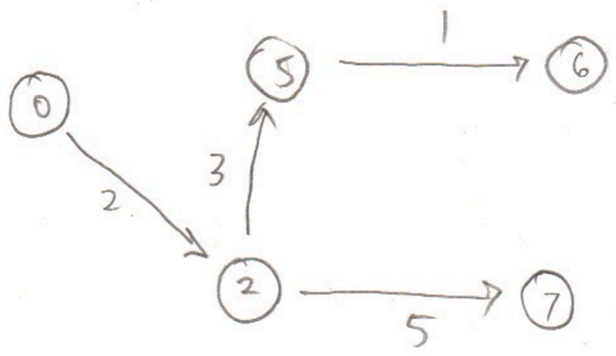
	1	2	3	4	5
1	0	1	0	0	1
2	0	0	1	1	0
3	0	0	0	0	0
4	0	0	1	0	1
5	0	1	0	0	0

Problem 4: Show the step-by-step result of running BFS on the graph below using y as the source. (20 points)



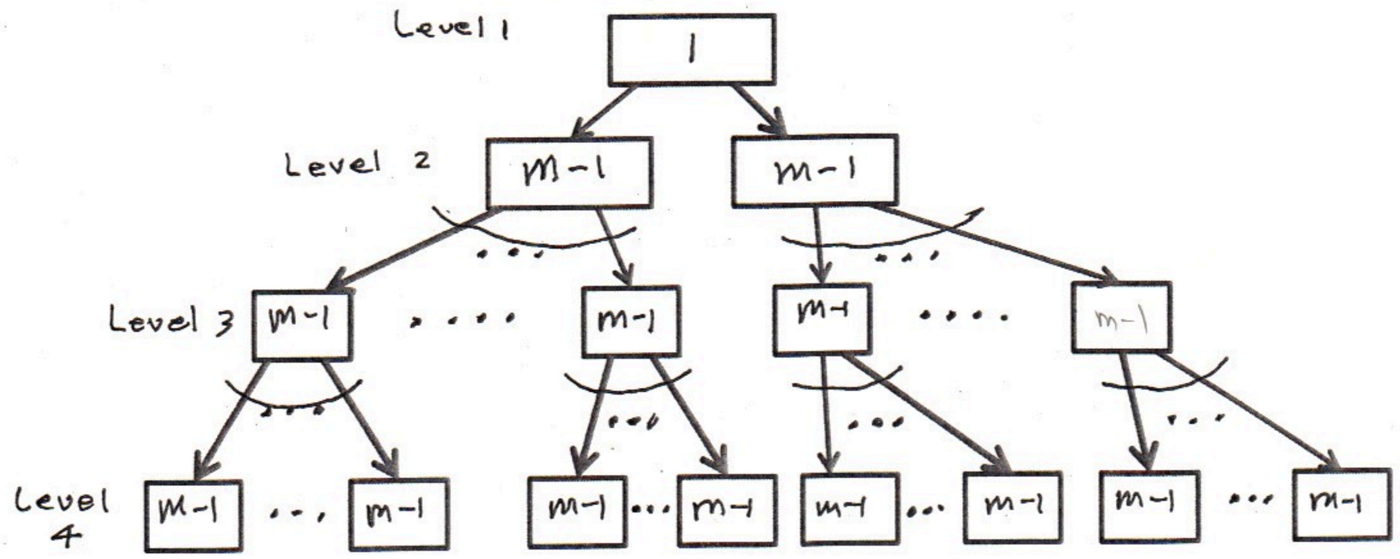
Problem 5: Work on the step-by-step Dijkstar algorithm for the graph below for the source node s. Give the priority queue at each step and fill in the final distance from s for each node. (20 points)





shortest path

Problem 6: Given a B-tree of order m as shown below, fill in the number of nodes at each tree level. (20 points)



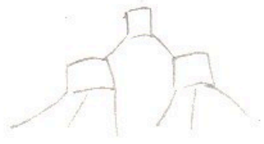
Number of nodes at level 2 = 2(m-1) (5 points) 2

-5 Number of nodes at level 3 = 4m(m-1) (5 points) 2m

Number of nodes at level 4 = $8m^2(m-1)$ (5 points) 2

Number of children fore each node (except root) = [m/2, m] (5 points)

If there are 1 million of children at level 4, what is m? Answer= 15 (5 points)



10 22 31 4 15 28 17 88 59 $m = 11$

1)

	T
0	22
1	88
2	
3	
4	4
5	15
6	28
7	17
8	59
9	31
10	10

m=11

a) Linear Probing

✓

Linear probing: K mod m

a) 10 mod 11 = 10, T[10] = 10

b) 22 mod 11 = 0, T[0] = 22

c) 31 mod 11 = 9, T[9] = 31

d) 4 mod 11 = 4, T[4] = 4

e) 15 mod 11 = 4, collision

(15+1) mod 11 = 5, T[5] = 15

f) 28 mod 11 = 6, T[6] = 28

g) 17 mod 11 = 6, collision

(17+1) mod 11 = 7, T[7] = 17

h) 88 mod 11 = 0, collision

(88+1) mod 11 = 1, T[1] = 88

i) 59 mod 11 = 4, collision

(59+1) mod 11 = 5, collision

(59+2) mod 11 = 6, collision

(59+3) mod 11 = 7, collision

(59+4) mod 11 = 8, T[8] = 59

Quadratic Probing ; $c_1 = 1$ $c_2 = 3$

a) 10 mod 11 = 10, T[10] = 10

b) 22 mod 11 = 0, T[0] = 22

c) 31 mod 11 = 9, T[9] = 31

d) 4 mod 11 = 4, T[4] = 4

e) 15 mod 11 = 4, collision

$[15 + 1(1) + 3(1^2)]$ mod 11

19 mod 11 = 8, T[8] = 15

f) 28 mod 11 = 6, T[6] = 28

g) 17 mod 11 = 6, collision

$(17 + 1 + 3(1^2))$ mod 11

17 + 1 + 3 = 21 mod 11 = 10, collision

$[17 + 1(2) + 3(2^2)]$ mod 11

17 + 2 + 12 = 31 mod 11 = 9, collision

$[17 + 1(3) + 3(3^2)]$ mod 11

47 mod 11 = 3, T[3] = 17

h) 88 mod 11 = 0, collision

$[88 + 1 + 3(1^2)]$ mod 11

= 88 + 1 + 3 = 92 mod 11 = 4, collision

$[88 + 1(2) + 3(2^2)]$ mod 11

[88 + 2 + 12] = 3, collision

$[88 + 1(3) + 3(3^2)]$ mod 11

= 118 mod 11 = 8, collision

$[28 + 1(4) + 3(4^2)]$ mod 11

140 mod 11 = 8, collision

$[88 + 1(5) + 3(5^2)]$ mod 11

168 mod 11 = 3, collision

[88 + 1(6) + 3(36)] mod 11 = 4, collision

88 + 1(7) + 3(49) mod 11 = 0, collision

88 + 1(8) + 3(64) mod 11 = 2, T[2] = 88

i) 59 mod 11 = 4, collision

$[59 + 1 + 3(1^2)]$ mod 11 = 8, collision

[59 + 2 + 3(4)] mod 11 = 7, T[7] = 59

	T
0	22
1	
2	88
3	17
4	4
5	
6	28
7	59
8	15
9	31
10	10

m=11

b) Quadratic Probing

✓

?

$\rightarrow$

c) Double Hashing : 10, 22, 31, 4, 15, 28, 17, 88, 59

double hash:
$[k + i(1 + k \bmod (m-1))] \bmod m$

T

22	0
	1
59	2
17	3
4	4
15	5
28	6
88	7
	8
31	9
10	10

m=11

a) 10 mod 11 = 10, T[10] = 10

b) 22 mod 11 = 0, T[0] = 22

c) 31 mod 11 = 9, T[9] = 31

d) 4 mod 11 = 4, T[4] = 4

e) 15 mod 11 = 4, collision

$[15 + i(1 + \underbrace{15 \bmod 10}_{5})] \bmod 11$

$[15 + 6] \bmod 11$

21 mod 11 = 10, collision

$[15 + 2(1 + 15 \bmod 10)] \bmod 11$

15 + 2(6) = 27 mod 11 = 5, T[5] = 15

f) 28 mod 11 = 6, T[6] = 28

g) 17 mod 11 = 6, collision

✓

$[17 + i(1 + 17 \bmod 10)] \bmod 11$

$[17 + 1 + 7] \bmod 11$

25 mod 11 = 3, T[3] = 17

h) 88 mod 11 = 0, collision

$[88 + (1 + 88 \bmod 10)] \bmod 11$

$[88 + (9)]$ = 97 mod 11 = 9, collision

$[88 + 2(1 + 88 \bmod 10)] \bmod 11$

$[88 + 2(9)] \bmod 11$

106 mod 11 = 7, T[7] = 88

i) 59 mod 11 = 4, collision

$[59 + (1 + 59 \bmod 10)] \bmod 11$

$[59 + (1 + 9)] \bmod 11$

69 mod 11 = 3, collision

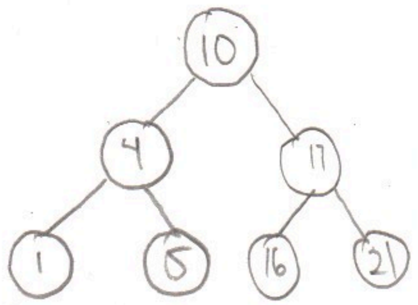
$[59 + 2(1 + 59 \bmod 10)] \bmod 11$

$[59 + 2(10)] \bmod 11$

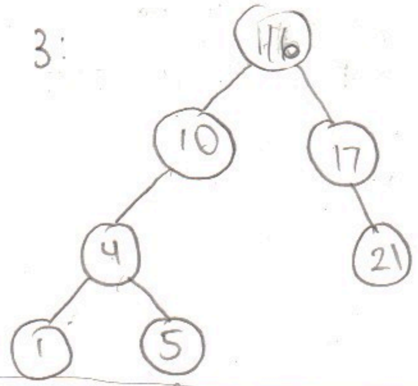
79 mod 11 = 2, T[2] = 59

2) [1, 4, 5, 10, 16, 17, 21] , binary search trees of height 2, 3, 4, 5, 6

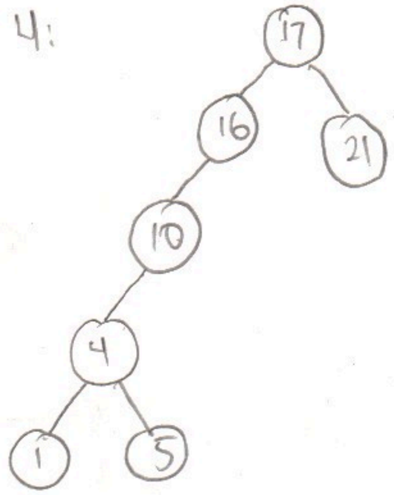
height 2:



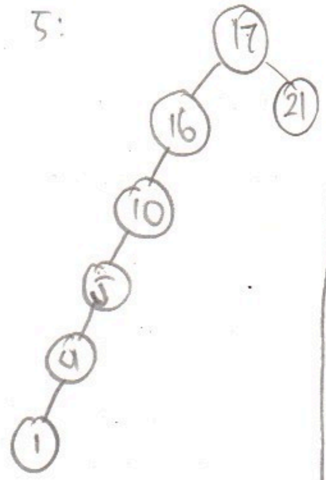
height 3:



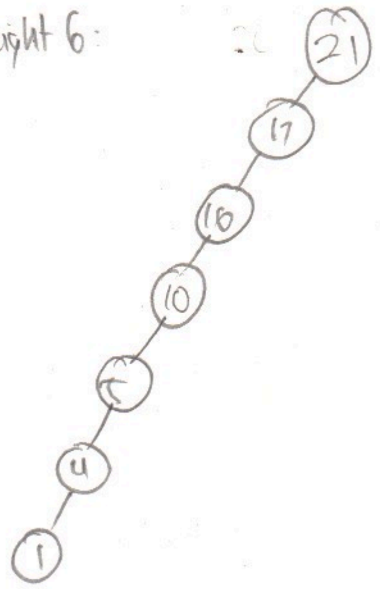
height 4:



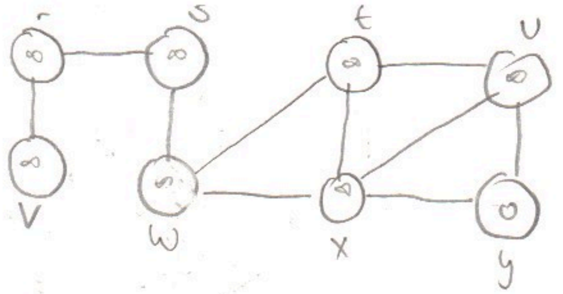
height 5:



height 6:



4.

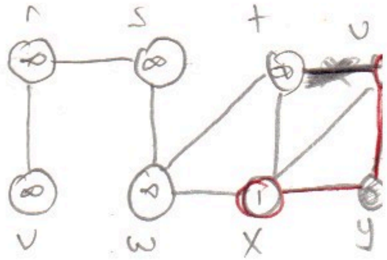


1) initialization: since y is the source, we set it to 0 and put it in the queue. All other nodes are set to infinity

Q $\underset{0}{\boxed{y}}$

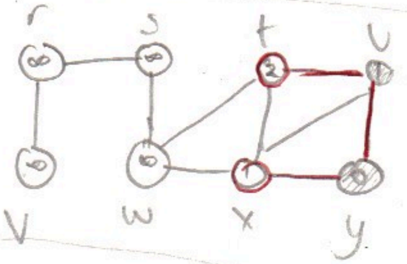
2) Take y out of the queue, mark it visited. See adjacent nodes, which are u and x. Put them in the queue w/ values of 1.

Q $\underset{1\ \ 1}{\boxed{u \mid x}}$



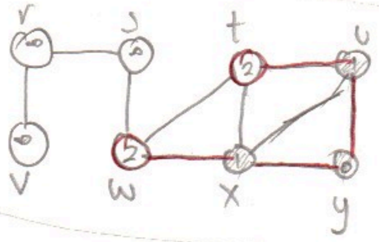
3) Take u out of queue, mark visited. Adjacent nodes are t and x, x is already in the queue, so we just add t with value 2.

Q $\underset{1\ \ 2}{\boxed{x \mid t}}$



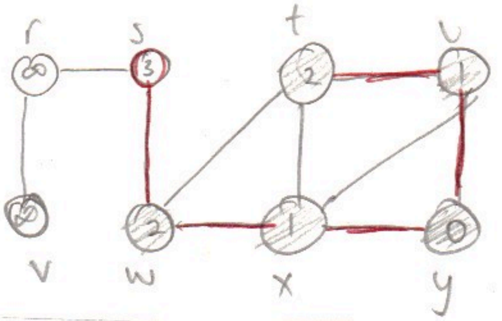
4) Take x out of the queue, mark visited. The only adjacent node that is not in the queue is w. Add w to queue with value 2.

Q $\underset{2\ \ 2}{\boxed{t \mid w}}$



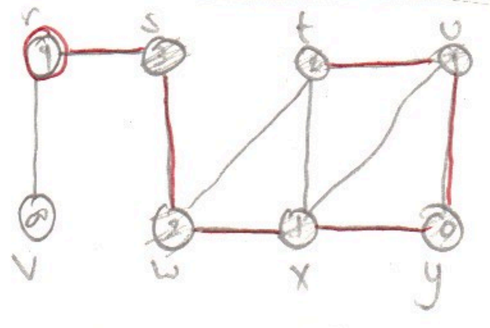
5) Take t out of queue, mark it visited. There are no adjacent node that can be put into the queue because they have already been in the queue, or are in the queue and would cause loops. Move onto w. Take w out of queue, and mark visited. Only adjacent node is s, put s in the queue with value 3.

Q $\underset{3}{\boxed{s}}$

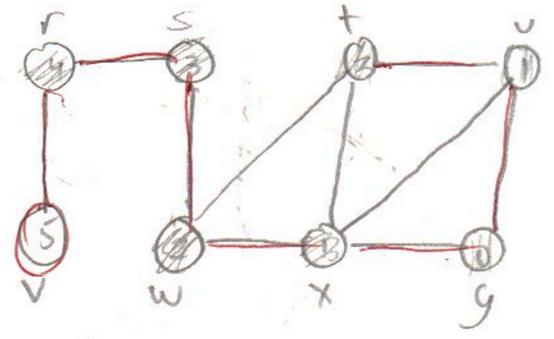


6) Take s out of queue, mark visited. r is only adjacent node, so we add to queue w/ value 4.

Q $\underset{4}{\boxed{r}}$

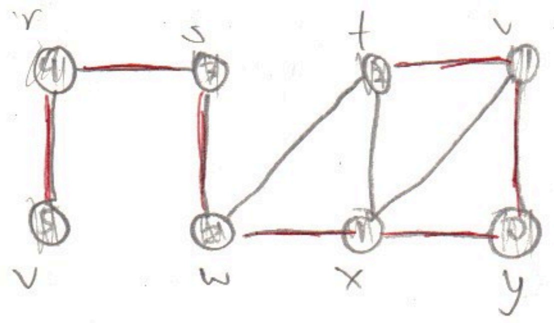


7) Take r out of queue, mark it visited. v is the only adjacent node, so we add it to queue w/ value 5.



Q $\boxed{v}$
5

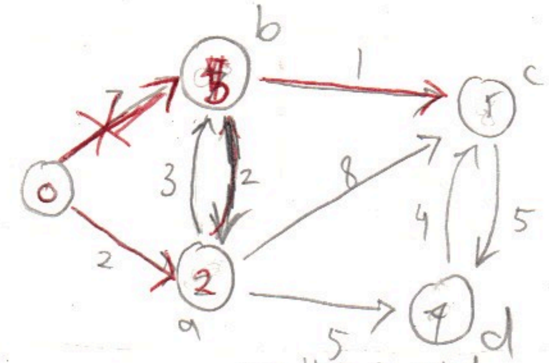
8) Take v out of queue, mark visited. no adjacent node. Check back to queue, it is empty. Algorithm Complete



Q ∅

	r	s	t	u	v	w	x	y
d	4	3	2	1	5	2	1	0
π	s	w	v	y	r	x	y	∅

5)



v	s	b	a	c	d
d[v]	0	∞	∞	∞	∞
π	nil	nil	nil	nil	nil
Color	w	w	w	w	w
Priority	s				
queue	0				

initialization

1) Take s out of queue, mark black for visited Add b and a to queue w/ values 7 and 2 respectively.

v	s	b	a	c	d
d[v]	0	7	2	∞	∞
π	nil	s	s	nil	nil
color	B	w	w	w	w
Priority	b	a			
queue	7	2			

2) Take b out of queue, mark black. b is adjacent to a and c. Traversing to a will give value 9, but a is 2 so I relaxed a. Add c to queue with value 8.

v	s	b	a	c	d
d[v]	0	7	2	8	∞
π	nil	s	s	b	nil
color	B	B	w	w	w
Priority	a	c			
queue	2	8			

3) take a out of queue, mark visited. a is adjacent to be, giving value of 4, which is smaller than 7. I changed the path to b, replaced its value with 5 and put b back into queue. a is adjacent to c, but the value is larger than c's current value, so I relaxed c. d is unvisited, so I put it in the queue w/ value 7.

v	s	b	a	c	d
d[v]	0	5	2	8	7
π	nil	a	s	b	a
color	B	W	B	W	W
priority	c	b	d		
queue	8	4	7		

4) Take c out of queue, mark visited. only adjacent is d, but I relaxed d due to c's path giving larger number. go back to queue and revisit b. since b is now 5, I can change c to 6 and put c back into queue

v	s	b	a	c	d
d[v]	0	5	2	6	7
π	nil	a	s	b	a
color	B	B	B	W	W
priority	d	c			
queue	7	5			

5) take d out of queue, mark visited. only adjacent node is c, but I relaxed c because value will be larger. go to queue and take out c, mark visited. c is only adjacent to d, but value will be larger than d's current value, so I relax d. Queue is empty, algorithm complete

v	s	b	a	c	d
d[V]	0	5	2	6	7
π	nil	a	s	b	a
color	B	B	B	B	B
queue	∅				