11/11/15

9. Homework

Due **11/19/15** at the beginning of the lab

1. **Queue from Stacks (9 points)**

   Assume we are given an implementation of a stack, in which Push and Pop operations take constant time each. We now implement a FIFO queue using two stacks A and B as follows:

   Enqueue(x):
   • Push x onto stack A

   Dequeue():
   • If stack B is nonempty, return B.Pop()
   • Else, pop all elements from A and immediately push them onto B. Return B.Pop()

   (a) (1 point) Show how the following sequence of operations operates on the two stacks. Suppose the stacks are initially empty.
   Enqueue(1), Enqueue(2), Enqueue(3), Enqueue(4), Dequeue(), Enqueue(5), Enqueue(6), Dequeue()

   (b) (1 point) Why do these implementations of Enqueue and Dequeue correctly implement FIFO queue behavior? *(Hint: Which invariant holds for A and B?)*

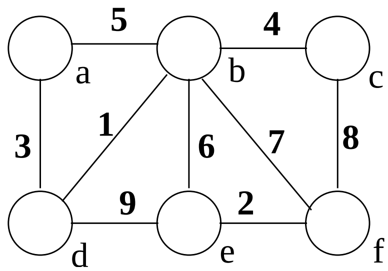
   (c) (1 point) What is the worst-case runtime of a single Enqueue operation? What is the worst-case runtime of a single Dequeue operation?

   (d) (3 points) Prove using the accounting method that the amortized runtime of Enqueue and Dequeue each is $O(1)$. Argue why your account balance is always non-negative.

   (e) (3 points) Use aggregate analysis to show that the amortized runtime of Enqueue and Dequeue each is $O(1)$.

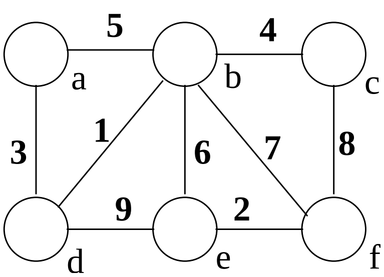
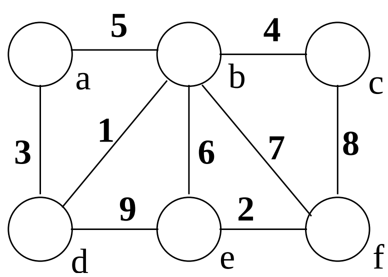
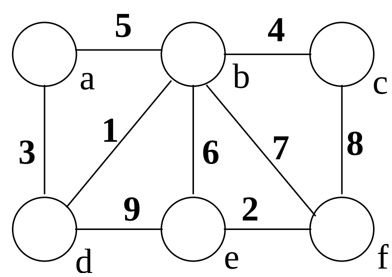
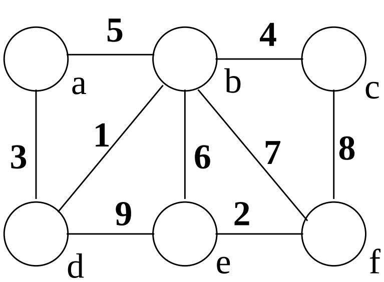
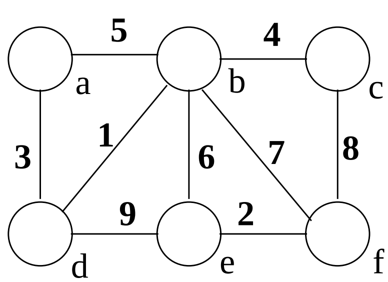
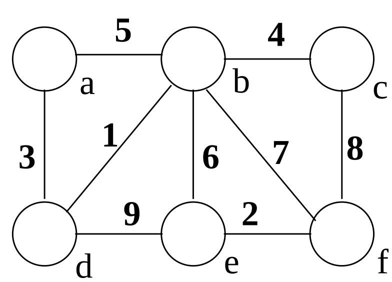
2. **Binary Counter (5 points)**
   Use aggregate analysis to show that, over a sequence of n increment operations on a binary counter, the amortized runtime of one such increment operation is $O(1)$. *(Hint: Study the flipping behavior of every single bit $A[i]$.)*

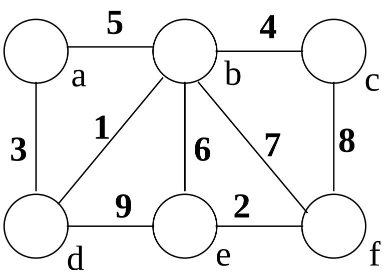
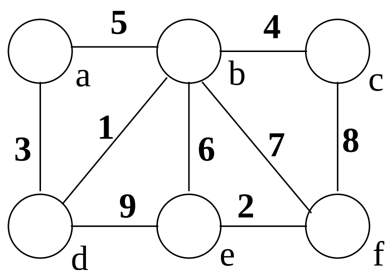
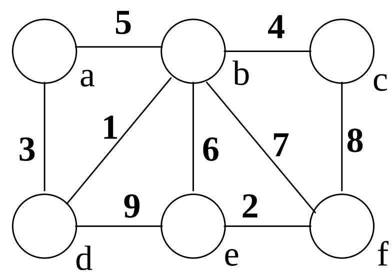
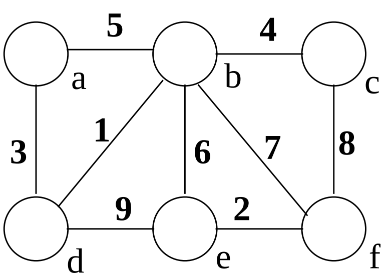
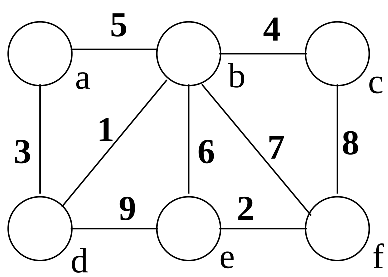
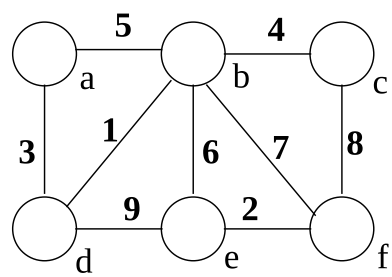
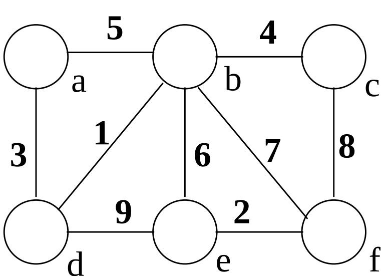
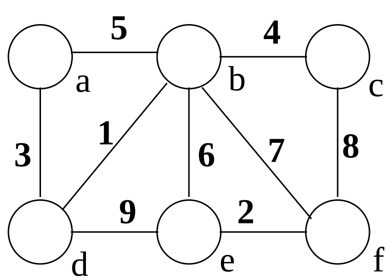
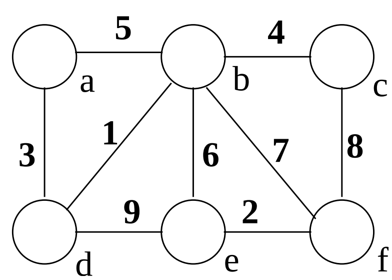
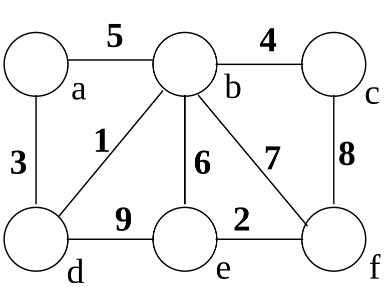
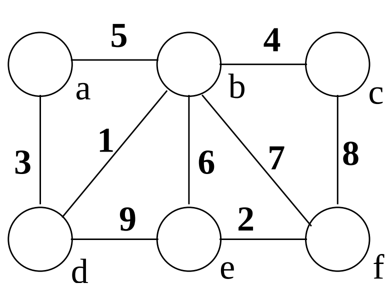
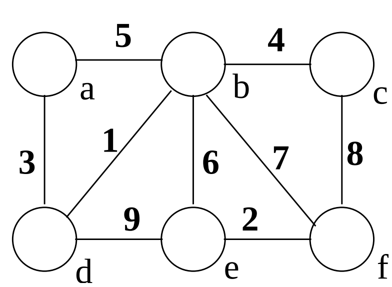
Flip over to next page ⟹



3. **Prim (5 points)**
   Run Prim's algorithm on the graph above, with start vertex a. Assume that vertices are ordered alphabetically. Show all the different stages of the algorithm (vertex weights, tree edges stored in the predecessor array, and the priority queue). You may use a copy of the next page for your convenience.

4. **Kruskal (5 points)**
   Run Kruskal's algorithm on the graph above. Show all the different stages of the algorithm (tree edges and the set of vertex subsets). Clearly indicate the minimum spanning tree. You may use a copy of the next page for your convenience.


5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f

5
4
a
b
c
3
1
6
7
8
9
2
d
e
f