Problem 1 (10 points)

1. Give a formal specification of the following problem: given an array of integers, and another integer x, determine whether there are two elements in the array that sum up to x.

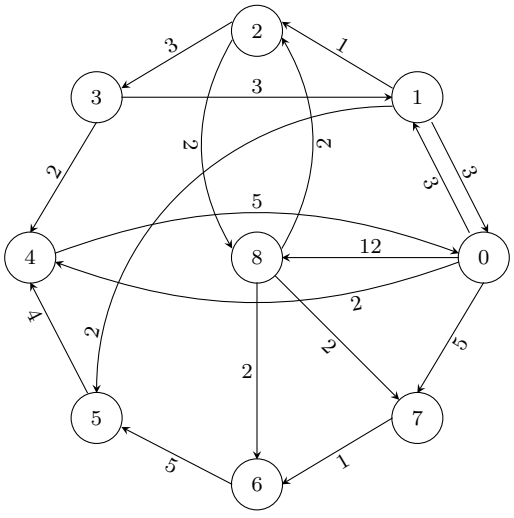
2. Design an algorithm that solves the problem of the previous part in $O(n \log n)$ steps, where n is the length of the array, and a step is either an addition or a comparison of integers.

Problem 2 (15 points) Suppose that you are given a sorted sequence of n *distinct* integers $\{a_1, \ldots, a_n\}$. Give an algorithm to determine whether there exists an index i such that $a_i = i$, outputting one i if such an i exists, and which uses $O(\log(n))$ steps. For example, in $\{-10, -3, 3, 5, 7\}$, $a_3 = 3$, whereas $\{2, 3, 4, 5, 6, 7\}$ has no such i.

Problem 3 (15 points) Given an $O(n \log(k))$-algorithm that merges k sorted list of integers with a total of n elements into one sorted list.

Hint: Use a heap of size at most k.

Problem 4 (20 points) Show the successive node values computed in the execution of the Moore-Bellman-Ford algorithm on this graph, assuming that node 0 is the starting node s. Moreover, for every node v, determine a shortest path from s to v.



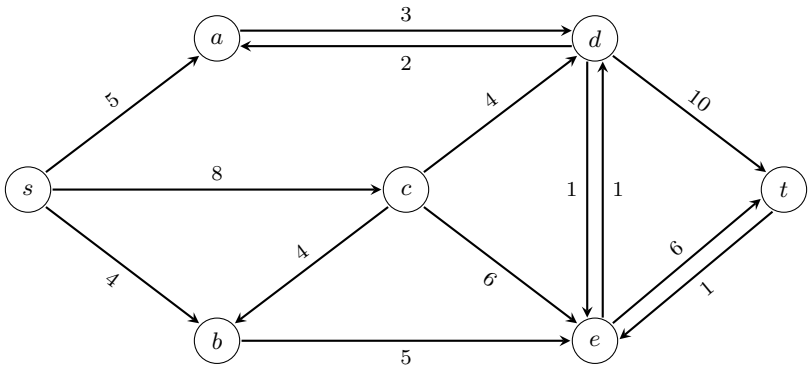
Your output should look like this:

	0	1	2	3	4	5	6	7	8
Step 0	0	∞	∞	∞	∞	∞	∞	∞	∞
Step 1	0								
Step 2	0								
$\cdots$	0	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Problem 5 (20 points) A max-min algorithm finds both the largest and the smallest elements in an array of n integers. Design and analyze a divide-and-conquer max-min algorithm that uses $\lceil 3n/2 \rceil - 2$ comparisons for any integer n. (You will receive 10 points if you can show this for the case when n is a power of 2, and an additional 10 points if you can prove it for general n.)

Hint: If $T(n)$ denotes the number of comparisons of your algorithm, try to find a recursion relating $T(n+m)$ to $T(n)$ and $T(m)$. Then study first the case where n is a power of 2.

Problem 6 (20 points) Calculate a maximal flow and a minimal cut on the following network[1] using the Ford-Fulkerson algorithm. At every step, draw the residual graph corresponding to the current flow.



Bonus Problem (20 points) Consider a binary heap containing n numbers where the root stores the largest number. Let $k < n$ be a positive integer, and x be another integer. Design an algorithm that determines whether the kth largest element of the heap is greater than x or not. The algorithm should take $O(k)$ time and may use $O(k)$ additional storage.

Hint: don't try to *find* the kth largest element.

[1] In the 2011 version of the course we did not assume that the graph was free from anti-parallel edges. You can obtain a graph without the anti-parallel edges by using the standard gadget trick that we explained in class and that is explained in the book.