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.

Solution:

1. (4 points) The input set I is $(\cup_{n=1}^{\infty} \mathbb{Z}^n) \times \mathbb{Z}$. The output set O is {TRUE,FALSE}. The relation is

$$\begin{aligned} R \quad = \quad & \left\{ ((\sigma, x), \text{TRUE}) \;\middle|\; \exists\, 0 \le i < j < |\sigma| \colon\; \sigma_i + \sigma_j = x \right\} \cup \\ & \left\{ ((\sigma, x), \text{FALSE}) \;\middle|\; \forall\, 0 \le i < j < |\sigma| \colon\; \sigma_i + \sigma_j \neq x \right\} \end{aligned}$$

   where $|\sigma|$ is the length of the sequence σ.

2. (6 points) Given an array σ consisting of n elements, and the integer x, we first sort σ using $O(n \log n)$ operations. Next, we initialize two indices i and j to $i = 0$ and $j = n - 1$. As long as $i < j$, we perform the following: if $\sigma_i + \sigma_j > x$, then we decrease j by one. If $\sigma_i + \sigma_j < x$, then we increase i by one. We stop and return the value TRUE if we hit (i, j) with $\sigma_i + \sigma_j$. If during the course of the algorithm $i \ge j$, then we return the value FALSE. The running time of this algorithm is obviously $O(n)$, so the total running time is $O(n \log n)$.

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.

Solution: The key is the following realization: if $a_i < i$, then for all $\ell \le i$ we have $a_\ell < \ell$, and if $a_i > i$, then for all $\ell \ge i$ we have $a_\ell > \ell$. To see this, note that in the former case, $a_{i-\ell} \le a_i - \ell \le i - \ell$ since the a_i's are distinct and integers. In the latter case, $a_\ell \ge a_i + (\ell - i) \ge \ell$, again since the a_i's are distinct.

Now we use binary search on the sequence $(a_i - i | 1 \le i \le n)$ to find a zero of this sequence if it exists.

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.

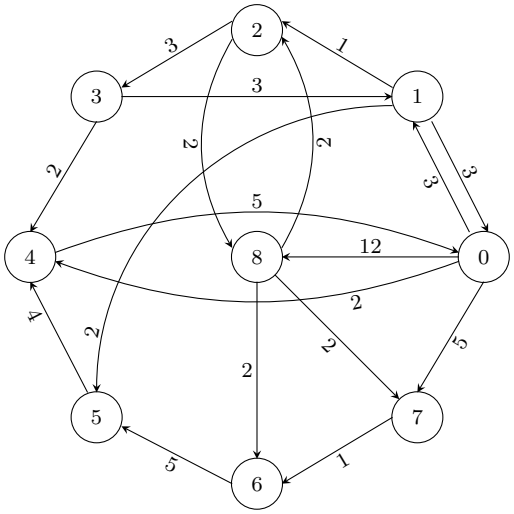
Hint: Use a heap of size at most k.

Solution: Let the arrays be $A_1, \ldots, A_k$ and assume that they are sorted in an ascending way. We will create sorted array S consisting of the union of the A_i's.

```
Create min-heap for elements (1,A_1[0]),...,(k,A_k[0]) where
the minimization is with respect to the second variable.
i = 0;
while ( i < n ) do
  (j,A_j[l]) = DeleteMin of the Heap;
  S[i] = A_j[l];
  i = i+1;
  if (l is not the length of A_j ) then
     Insert A_j[l+1] into the min-heap.
```

For every entry of S we need to do one deletemin operation and at most one insertion operation into the heap. The heap size is at most k, so these operations cost $O(\log(k))$. In total we will have an $O(n \log(k))$-algorithm.

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	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Solution: (12 points for the trace, 3 points for each step. 8 points for the shortest paths, 1 point for each path)

The trace of the MBF algorithm is the following:

	0	1	2	3	4	5	6	7	8
Step 0	0	∞	∞	∞	∞	∞	∞	∞	∞
Step 1	0	3	∞	∞	2	∞	∞	5	12
Step 2	0	3	4	∞	2	5	6	5	12
Step 3	0	3	4	7	2	5	6	5	6
Step 4	0	3	4	7	2	5	6	5	6

The shortest paths are the following:

$0 \to 1$
$0 \to 1 \to 2$
$0 \to 1 \to 2 \to 3$
$0 \to 4$
$0 \to 1 \to 5$
$0 \to 7 \to 6$
$0 \to 7$
$0 \to 1 \to 8$

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.

Solution: Here is the algorithm:

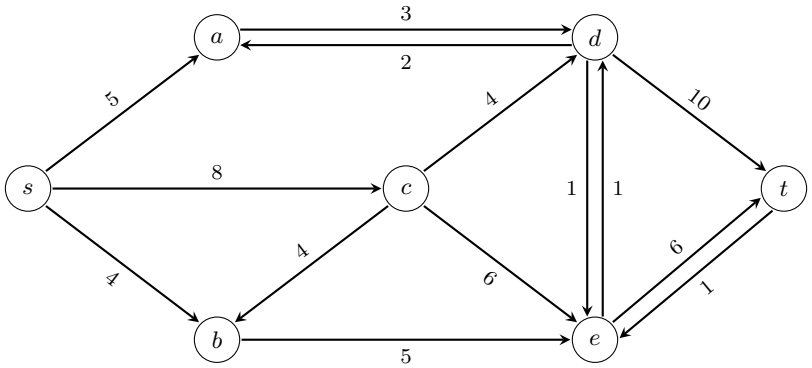
```
A is an array of n integers
Max-Min(A) returns [max,min], the maximum and the minimum of A

if ( n == 1 ) then
    return max=min=A[0]
l = ceil(n/2), m = n-l
[max_1, min_1] = Max-Min(A[0:l-1])
[max_2,min_2] = Max-Min(A[l:n-1])
Return [max(max_1,max_2), min(min_1,min_2)]
```

It is clear that $T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 2$, and $T(1) = 0$. We will now show by induction that $T(n) = \lceil 3n/2 \rceil - 2$. The assertion is true for $n = 1$.

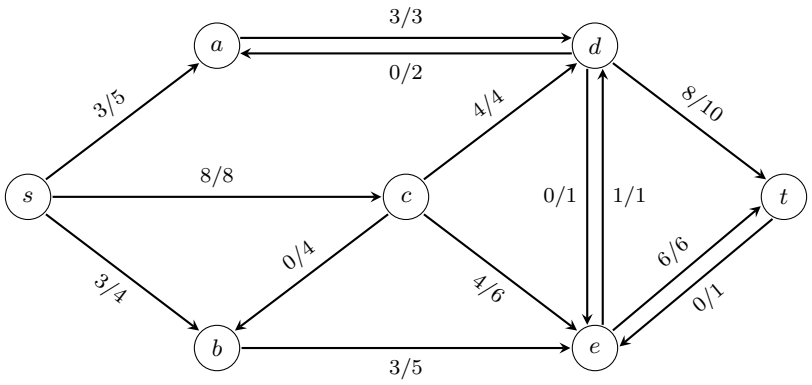
Assume now that the assertion is true for all $m \leq n-1$. We want to prove it for n.
Case 1. $n = 2m$, m odd. In this case, we write $T(n) = T(m-1)+T(m+1)+2 = 3(m-1)/2 + 3(m+1)/2 - 2 = 3n/2 - 2$.
Case 2. $n = 2m$, m even. In this case, we write $T(n) = 2T(m)+2 = 2 \cdot 3m/2 - 2 = 3n/2 - 2$.
Case 3. $n = 2m+1$. In this case, we write $T(n) = T(m) + T(m+1) + 2 = \lceil 3m/2 \rceil + \lceil 3(m+1)/2 \rceil - 2$. Let us assume that m is even. In this case the expression is equal to $3m/2 + 3(m+1)/2 + 1/2 - 2 = \lceil 3n/2 \rceil - 2$. The case where m is odd is handled similarly.

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



Solution: (15 points for the max-flow, 5 points for the min-cut)

The value of the maximal flow is 14. The following is a possible maximal flow:



A possible minimum cut is given by the subsets $\{s, a, b, c, e\}$ and $\{d, t\}$.

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.

Solution: Starting from the root of the tree, traverse the children of a node until the value of the node is strictly smaller than x. Put the index of this node into an array of length $2k$. If there is an overflow, then state that the kth largest element is strictly larger than x. *Claim: if the k-th largest element is smaller than or equal to x, then the array will contain at most $2k$ entries.* Proof. For every node in the array, the value of the ancestor is greater than or equal to x. There are thus at most k such ancestors. Each ancestor has at most 2 children, hence the number of entries in the array is at most $2k$. *Claim: running time is $O(k)$.* Proof. The number of nodes traversed is at most the number of direct descendants of nodes whose values are greater than or equal to x, hence at most $2k$.

Now we will use the new array to see whether the kth largest element of the array is at least x.