Problem 1. There are $\lfloor \log_2 n \rfloor + 1$ powers of 2 between 1 and n, including $1 = 2^0$. For $\lfloor \log_2 n \rfloor$, the operation costs more than 1. All together, these operations cost $\sum_{i=0}^{\lfloor \log_2 n \rfloor} 2^i < 2^{\log_2(n)+1} = 2n$ (the sum of all powers of 2 up to 2^k is $2^{k+1} - 1$). The remaining less than n operations each cost 1. Overall, the cost of n operations is bounded by $3n$, that is, each operation costs no more than 3 on average.

Problem 2. At each power of 2 there needs to be enough credit to pay for the operation. At most, all credit can be used up. In that case, the operations up to the next power of 2 need to build up enough credit to pay for it. For every i which is a power of 2, there are $i/2$ operations (counting the i-th one itself) since the previous power of 2. The total cost of these operations is $i + (i/2 - 1) < 1.5 * i$. To charge a total of $1.5 * i$ over $i/2$ operations, we charge each operation a cost of 3. That is enough to pay for the operations where i is not a power of 2 and by the above explanation, it builds up enough credit to pay for those operations where i is a power of 2.

Problem 3. The number of undirected pairs (2-element sets) of vertices is $n+(n-1)+(n-2)+\cdots+3+2+1 = (n+1)*n/2$. The number of directed pairs is $n*n$, from each vertex to each vertex (it is not quite twice as much as the number of directed pairs because for self-loops there are no two directions). For each of these pairs, an edge can exist or not. Therefore, there are $2^{(n+1)*n/2}$ distinct undirected graphs and 2^{n^2} distinct directed graphs.

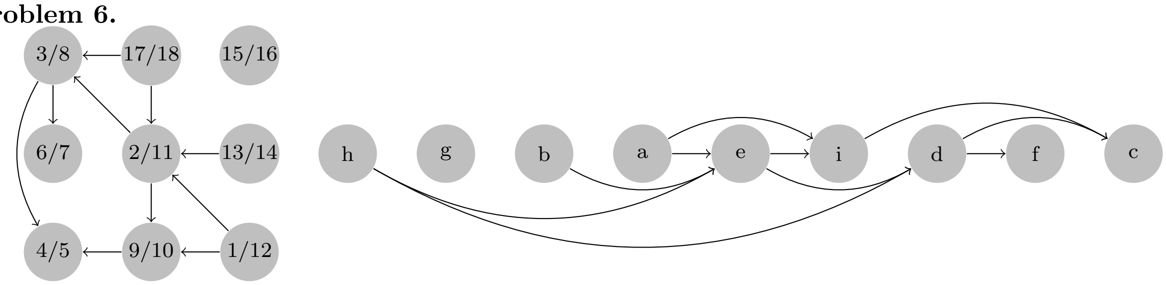
Problem 4.

	Complexity	Explanation	Advantage / Disadv.
Storage	$\Theta(V + E)$	constant time per linked list element	same
Lookup	$O(V)$	find u, then find v in adjl list	slower (larger constant)
Insert edge	$O(V)$	find u, then insert edge	asymptotically slower
Delete edge	$O(V)$	lookup edge, delete in constant time	slower (larger constant)
Insert vertex	$O(1)$	insert vertex at beginning of list	asymptotically faster
Delete vertex	$O(V + E)$	iterate over adj. lists to remove references	faster (smaller constant)

Problem 5.

	adjacency lists	adjacency matrix
single in-degree	$O(V + E)$	$\Theta(V)$
single out-degree	$O(V)$	$\Theta(V)$
all in-degrees	$\Theta(V + E)$	$\Theta(V^2)$
all out-degrees	$\Theta(V + E)$	$\Theta(V^2)$

Problem 6.



Problem 7. Treat each possible configuration of the puzzle as a vertex and connect two vertices via an edge if they can be converted into each other with a single move. This yields 16! vertices and between 16 and $2*16!$ edges (two to four edges per vertex). The task is to find a shortest path, with regard to number of edges, from the vertex representing the starting configuration to the vertex representing the solved puzzle. BFS can be used to find such a path, if it exists. The graph can be "built" during traversal since the adjacencies for each vertex can be computed and need not be given or determined in advance.