Problem#1) In a sequence of n operations there are $\lfloor \log_2 n \rfloor + 1$ exact powers of 2. $(1, 2, 4, \ldots 2^{\lfloor \log_2 n \rfloor})$.

The total cost of all these operations will be

$$\sum_{i=0}^{\lfloor \log_2 n \rfloor} 2^i = 2^{\lfloor \log_2 n \rfloor + 1} - 1 \leq 2^{\log_2 n + 1} = 2n$$

All the other remaining operations are cheap - their cost will be 1\$. There are less than n such operations.

Total cost of all operations is

$$T(n) \leq 2n + n = 3n = O(n)$$

From this we conclude $\underline{O(1)}$ amortized cost per operation.

Problem #2)

We modify every operation with \$3, where \$1 is paid for operations of non-power of 2
\$2 is stored and paid later for operations of power of 2

From this we conclude $\underline{O(1)}$ amortized cost per operation (\$3)

Problem #3) <For undirected graphs>

Let's say $n=1$, then # of edges $=0$
if $n=2$, # of edges $=1$ (max)
if $n=3$, # of edges $=3$ (max)
if $n=4$, # of edges $=6$ (max)

We know that an edge is placed between 2 vertices so total # of edges is a combination of it. $\Rightarrow (n \cdot (n-1))/2$

so total # of graphs is $2^{((n \cdot (n-1))/2)}$

<For directed graphs>

For directed graphs edges can be maximum in 2 directions, so total # of edges is $n \cdot (n-1)$
and total # of graphs is $2^{(n \cdot (n-1))}$

Problem #4 Let's assume, the graph is directed, with V-vertices, and E-edges.

Space complexity: space complexity for the list of Adjacency lists will be $O(3V+2E)=O(V+E)$ and for array of Adjacency lists will be $O(V+2E)=O(V+E)$

Time complexity for lookup operation: in case of the list of Adjacency lists we have to search for the 1st vertex (of any edge(x,y)), which will take $O(V)$ in the worst case, and then to search for the 2nd vertex (y) will take $O(V)$ as well in the worst case. Thus, the asymptotic bound for lookup operation in the list of Adjacency lists will be $O(2V)=O(V)$.

In case of the array of Adjacency lists, it will take $O(1)$ to search for x, but $O(V)$ to search for y, which will take in total $O(V)$ time.

Time complexity for insertion of a new vertex:
in the list of Adjacency lists it will take $O(1)$ time, because we can add a new vertex at the first position. In the array of Adjacency lists it will take $O(1)$, because each vertex is assigned an index-we can just add a new one at the end.

Time complexity for insertion of an edge: In the lists of Adjacency lists we will search for the first vertex, and then we can add vertex to the adjacency list, which will take $O(V)$ in the worst case.
In the array of adjacency list, searching for a vertex will take $O(1)$ time, because all vertices have indices, and to add another vertex will take $O(1)$ time. Total time $O(1)$.

Deletion of the vertex: In case of the lists of adjacency lists, it will take $O(V)$ to search and delete the whole node and then we will have to go to each node and remove that vertex from the adjacency list of each vertex. It will take $O(V+E)$ time. So total time complexity will be: $O(V)+O(V+E)=O(V+E)$
For an array of adjacency lists, to search for a vertex it will take $O(1)$ time, resulting in total $O(1)+\underbrace{O(V+E)}_{\text{deletion}}=O(V+E)$

Deletion of the edge (x,y): In the lists of adjacency lists, we can get to the node with vertex x in $O(V)$ time; then to search for a vertex $y \to O(V)$, resulting in overall time of $O(V+V)=O(V)$.
In the array of adjacency lists deletion will be done in $O(V)$.

Time comlexity for an edge insertion will take faster for an array of adjacency list. And searching is done faster for an array of A.L.

Problem #5 <Adjacency list>

$|V|$ - # of vertices

$|E|$ - # of edges

In order to calculate the outdegree of a vertex in adjacency list, we have to traverse the whole list to find the vertex. Then, we have to count the # of edges, resulting in overall assymptotic bound $O(|V|+E)$.
To calculate the indegree of a vertex of adjacency list, we have to traverse the whole list and search for vertex in the edge set of all vertices, resulting in asymptotic bound $O(|V|\cdot|E|)$.
To calculate the outdegree or indegree of all vertices at once, we have to traverse all the nodes and their edge sets. $\Rightarrow$ asymptotic bound $\Theta(|V|\cdot|E|)$.

< Adjacency matrix >

To calculate the indegree or outerdegree of a vertex, we just have to traverse the row or column of the adjacency matrix respectively. ⇒ asymptotic bound $\Theta(|V|)$

To calculate the indegree and outdegree of all vertices at once, we have to traverse the whole matrix.
⇒ asymptotic notation $\Theta(|V^2|)$.

Problem #6 DFS order is a, e, d, c, f, i, b, g, h

ae; a,e,d; aed,c; a,e,d,c,f; a,e,d,c,f,i; a,e,d,c,f,i,b; a,e,d,c,f,i,b,h; a,e,d,c,f,i,b,h,g

vertex	a	b	c	d	e	f	g	h	i
finishing time	6	7	1	3	5	2	8	9	4

Problem #7 We can state Problem of the 15-puzzle as follows:

1. It can be represented as a problem of moving tokens on a given graph
2. For a given graph G on n vertices, the puzzle (G) is the graph.
3. Nodes - represents all possible placements of n-1 different tokens of G.
4. Adjacency - slides 1 token along edge of G to an empty vertex.
5. It is to be checked that whether two token configurations are in one component of puzzle (G).
6. Is graph puzzle (G) connected?

Solution: The graph puzzle (G) is connected if G is a 2-connected graph except if:
1. G is a cycle with more than 4 vertices
2. G is bipartite different from a cycle.
3. G is an exceptional graph $\theta(0)$
4. G is path with $p \geq 2$
5. G is not connected.