Homework 4

Problem 5

1. {3,2,1,6,4,5} → {3,2,1,6,5,4} → {3,2,4,1,5,6} → {3,2,4,1,6,5}
2. **Pseudocode**(s):

```
1. $n := |s|$.
2. $l$ := the largest index such that $s_l < s_{l+1}$.
3. $r$ := the largest index such that $s_l < s_r$.
4. Swap $s_l$ and $s_r$.
5. For $i = 0 \dots \lfloor (n-l)/2 \rfloor$: Swap $s_{l+1+i}$ and $s_{n-i}$.
6. Return $s$.
```

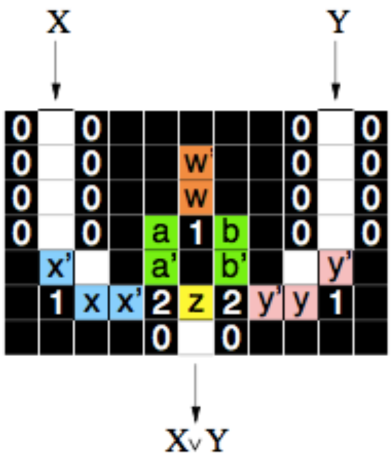
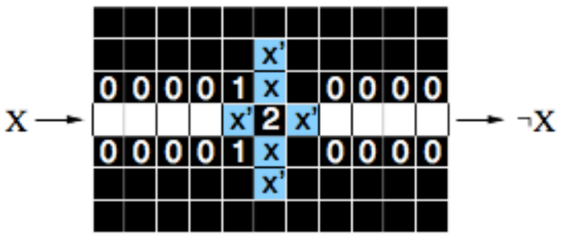
3. Aggregate method.
   - **Claim 1**: The sequence from s_{l+1} to s_n is in decreasing order and represents a sequence that finishes permutation. **Proof**: l is the largest index such that $s_l < s_{l+1}$, so s_{l+1} to s_n is in decreasing order. A decreasing sequence up to s_n means that no more permutation is allowed from s_{l+1} to s_n; otherwise, the lexicographical order is violated.
   - **Claim 2**: s_r is the next number to be in the l-th index where $l < r \leq n$. **Proof**: $s_l < s_{l+1}$, so the largest index r such that $s_l < s_r$ is at least $i+1$, i.e., $l < r \leq n$. The next number to be in the l-th index should be larger than s_l but the smallest from s_{l+1} to s_n; otherwise, the lexicographical order is violated.
   - **Claim 3**: After Step 4 swaps s_l and s_r, the decreasing property from s_{l+1} to s_n is still preserved. After Step 5, the sequence from s_{l+1} to s_n turns to an increasing sequence. **Proof**: $s_{r-1} > s_r > s_l > s_{r+1}$, so replacing s_r with s_l preserves the decreasing order. Step 5 reverse the sequence from s_{l+1} to s_n, hence, make a decreasing sequence be increasing.
   - **Claim 4**: A change of s_k occurs every $(n-k)!$ operations of $f(s)$. **Proof**: s_k is changed if and only if $k \geq l$. And $k \geq l$ if and only if the sequence from s_{l+1} to s_n finishes permutation, which occurs every $(n-k)!$ operations of $f(s)$. Hence, in a total of $n!$ operations, a change of s_k occurs $n!/(n-k)!$ times.
   - **Conclusion**: Suppose there are a total of N operations on $f(s)$. The total cost is $\sum_{k=1}^{n} N/(n-k)! \leq N \sum_{k=0}^{\infty} 1/k! = eN$. Hence, the amortized cost is $eN/N = e = O(1)$.

Problem 6

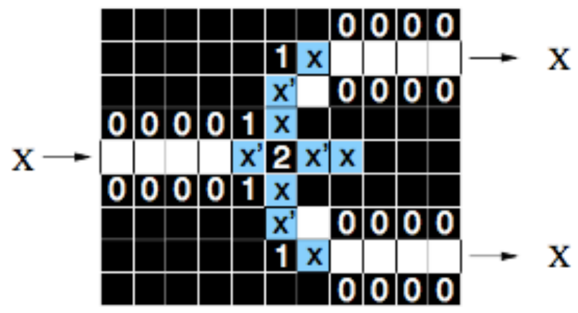
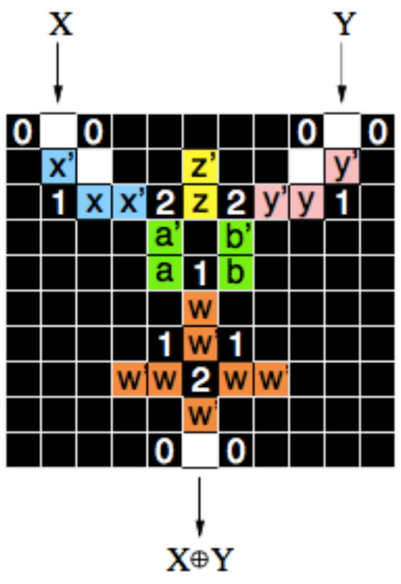
1. - Problem description:
     - **Vertex Cover**: $\{\langle G, k\rangle$: Graph G has a vertex cover of size $k\}$
     - **Halting Problem**: $\{\langle M, w\rangle$: Turing machine M halts on input word $w\}$
   - **Claim 1**: **Vertex Cover** is a known NP-complete problem.
   - **Claim 2**: There is a reduction from an instance of $\langle G, k\rangle$ for **Vertex Cover** to an instance of $\langle M, w\rangle$ for **Halting Problem**.

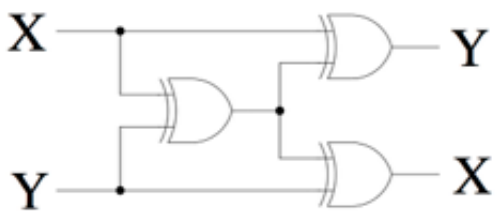
        1. **Vertex Cover** is a known NP-complete problem, so there is a non-deterministic Turing machine V that solves **Vertex Cover**.
        2. Define Turing machine H as follows: On input $\langle V, \langle G, k \rangle \rangle$: Run V on $\langle G, k \rangle$. Return if V accepts. Loop if V rejects.
        3. The input word for **Halting Problem** is constructed as $\langle H, \langle V, \langle G, k \rangle \rangle \rangle$. **Halting Problem** returns **true** if H halts on $\langle V, \langle G, k \rangle \rangle$, otherwise, returns **false**.
    - **Claim 3**: G has a vertex cover of size k iff H halts on $\langle V, \langle G, k \rangle \rangle$.
        - $\Rightarrow$: Suppose G has a vertex cover of size k. Then, V accepts $\langle G, k \rangle$. Then, H halts on $\langle V, \langle G, k \rangle \rangle$.
        - $\Leftarrow$: Suppose H halts on $\langle V, \langle G, k \rangle \rangle$. Then, V accepts $\langle G, k \rangle$. Then, G has a vertex cover of size k.
    - **Claim 4**: The machine description of V and H can be constructed in constant time, regardless of G and k. Hence, there is polynomial time reduction from $\langle G, k \rangle$ to $\langle M, w \rangle = \langle H, \langle V, \langle G, k \rangle \rangle \rangle$.
    - **Conclusion**: **Halting Problem** is NP-hard.

2.
    1. Each solution is expressed as a set of position of light bulbs.
        - Solutions to puzzle A: $\{A_{2,1}\}$, $\{A_{2,4}\}$
        - Solutions to puzzle B: $\{B_{2,3}, B_{2,6}, B_{5,2}, B_{6,4}, B_{6,7}, B_{4,7}\}$, $\{B_{2,4}, B_{2,7}, B_{3,2}, B_{6,3}, B_{6,6}, B_{4,7}\}$
    2.
        - Solutions to puzzle C: $(a, b, c) = (0, 0, 1), (0, 1, 1), (1, 1, 0), (1, 0, 1)$.
        - Solutions to puzzle D: $(a, b, c) = (0, 0, 1), (1, 1, 0)$.
    3. There has to be a light bulb in z in order for z to light up. But the black cell right of z is 0.
    4. Reference: Brandon McPhail. Light Up is NP-complete.
        - **Claim 1**: The circuit satisfiability problem (Circuit-SAT) is a known NP-complete problem.
        - **Claim 2**: There is a reduction from Circuit-SAT to the puzzle.
            - Every circuit has an equivalent circuit with only NOR or NAND gates. The transformation can be done in polynomial time and is beyond the scope of this proof. The construction here focuses on the NOR gate.
            - Inputs and outputs of a gate corresponds to a formula. The NOR gate is reduced to the corresponding puzzle as below. Connecting the output of the OR gate/puzzle and the input of the NOT gate/puzzle results in a full NOR gate/puzzle.

OR gate/puzzle	**NOT gate/puzzle**
	

- Branches and crosses in a circuit remain to be handled. Branches are given as below. Crosses can be taken care of by 3 branches and 3 XOR gate/puzzle. XOR gate/puzzle is also given as below.

Branch	XOR gate/puzzle
	



- **Claim 3**: A circuit is satisfiable iff the puzzle is solvable.
  - ⇒: Suppose a circuit is satisfiable, there is an assignment of variables such that the last output evaluates to true. No contradiction of assignments occurs along the gates. Hence, there is always a way to put a light bulb on a row/column of white cells. The puzzle is solvable.
  - ⇐: Suppose the puzzle is solvable, there is no contradiction of assignments occurs along the white cells. Hence, there is an assignment of variables such that the last output evaluates to true. The circuit is satisfiable.
- **Claim 4**: The reduction is done in polynomial-time since every gate and branch/cross of a circuit reduces to a fixed number of puzzle grids.
- **Conclusion**: The decision problem of Light-Up is NP-hard.