**Problem 1**

1. **Aggregate method**

Ci = cost of i th operation

We have: **Ci = (if i is an exact power of 2) ? i : 1;**

|  |  |
| --- | --- |
| **i th operation** | **Cost** |
| 1 | 1 |
| 2 | 2 |
| 3 | 1 |
| 4 | 4 |
| 5 | 1 |
| 6 | 1 |
| 7 | 1 |
| 8 | 8 |
| … | … |
| … | … |
| n |  |

Average cost of operation = Total cost / n < 3

By aggregate analysis, cost per operation is O(1)

**Amortized method**

Ci = cost of Ith operation

We have: **Ci = (if i is an exact power of 2) ? i : 1;**

We also have amortized cost ^Ci = charge each operation = $3

* If i is not an exact power of 2, pay $1 and have $2 as credit
* If i is an exact power of 2, pay $i and using stored credit

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Cost | Actual cost | Credit remaining = credit remaining of (ith-1) + (cost - actual cost) |
| 1 | 3 | 1 | 2 |
| 2 | 3 | 2 | 2 + (3 - 2) = 3 |
| 3 | 3 | 1 | 3 + (3 - 1) = 5 |
| 4 | 3 | 4 | 5 + (4 - 3) = 4 |
| 5 | 3 | 1 | 4 + (3 - 2) = 6 |
| 6 | 3 | 1 | 6 + (3 - 1) = 8 |
| 7 | 3 | 1 | 8 + (3 - 1) = 10 |
| 8 | 3 | 8 | 10 + (3 - 8) = 5 |
| 9 | 3 | 1 | 5 + (3 -1) = 7 |
| … | 3 | … | … |
| … | 3 | … | … |

The amortized cost is $3 per operation,

We have that

Then we have => credit = amortized cost – actual cost >= 0

Because the amortized cost of each operation is O(1) and the amount of credit never goes negative, the total cost of n operation is O(n)

**Problem 2**

Please refer to **Prob2\_BubbleSort1.java** file

**Problem 3**

Please refer to **Prob4\_BubbleSort2.java** file

**Problem 4**

Please refer to **Prob4\_MySort.java** file

**Proof**

Just one FOR loop so it is O(n)