Homework 2

Yu-Chieh Kuo B07611039

Study Group Members:
b06201057 Yu-Chi Hsieh, r08323002 Ze-Wei Chen,b06202004 Han-Wen Chang.

This homework answers the problem set sequentially.

1. **Collaborators: Study Group Members**

Problem	(1)	(2)	(3)	(4)
Bubble Sort	$\theta(n^2)$	Yes	$\theta(n^2)$	$\theta(n^2)$
Merge Sort	$\theta(n \log n)$	Yes	$\theta(n \log n)$	$\theta(n \log n)$
Insertion Sort	$\theta(n^2)$	Yes	$\theta(n)$	$\theta(n^2)$
Quick Sort	$\theta(n^2)$	Yes	$\theta(n \log n)$	$\theta(n^2)$
Heap Sort	$\theta(n \log n)$	No	$\theta(n \log n)$	$\theta(n \log n)$

2. **Collaborators: None**
The pseudo code of algorithm is as below.

Algorithm 1 CyclicShiftFindMin(Array)

```
low ← index of the first element
high ← index of the last element
while Array[low] > Array[high] do
    mid ← ⌊(low+high)/2⌋
    if Array[low] < Array[mid] then
        low ← mid + 1
    else
        high ← mid
    end if
end while
return Array[low]
```

Correctness:
There are three cases for this problem: *mid* at left side, in the middle, and at right side. All three cases can find the minimum after running algorithm since this algorithm will change the searching boundary until finding the minimum.
Running Time:
At iteration 1, the number of total searching elements is n, and it will be halved until finding the minimum element in the array, which costs k iterations and $\frac{n}{2^k} = 1$. Therefore, we have the running time *i.e.* the count of iterations is $k = \log_2 n \in O(\log n)$.

3. (a) **Collaborators: None**
The pseudo code of algorithm is as below.

Algorithm 2 SortAlmostSortedArray(Array, k)

$N \leftarrow$ size of Array
for $i = 1, 2, \ldots \frac{N}{k-1}$ **do**
    sort Array$[ik \ldots ik + 2k]$
end for

Correctness:
Since this array is almost sorted by mismatching at most k spot from its actual location, therefore, sorting $2k$ elements from beginning to end will ensure that in each sorting iteration, the first k's element will be in the right location.
Running Time:
All N elements are sorted by at most k times, the time complexity is $O(n) \in O(n \log k)$.

(b) **Collaborators: None**
We have all possible case is $(k!)^{\frac{n}{k}}$. According to the decision tree, the time complexity is

$$\begin{aligned} O(n) &\geq \log_2 (k!)^{\frac{n}{k}} \\ &\geq \frac{n}{k} \log_2 (k!) \\ &= \frac{n}{k}(k \log_2 k - k \log_2 e + O(\log_2 k)) \\ &\geq n \log_2 k \\ &\in \Omega(n \log k) \end{aligned}$$

4. (a) **Collaborators: None**
Let $c_1 = 1, c_2 = 100000, c_3 = 101, n_1 = 999, n_2 = 1000, n_3 = 1, n = 1000$. Since $\frac{c_1}{n_1} \leq \frac{c_2}{n_2} \leq \frac{c_3}{n_3}$, by greedy algorithm, the total cost is $1 + 100000 = 100001$ but the optimal solution should be $1 + 101 = 102$.

(b) **Collaborators: Study Group Members**
By greedy algorithm, we let it pick machine $T_1 + T_2 + \cdots + T_k$ or T_{k+1} such that we have $\min \sum_i T_i, i \in \textit{the solution set}$, and we denote the greedy solution as G^*. If we let machine be able to produce partial masks instead of producing whole n_i masks, the optimal solution

$$\begin{aligned} OPT \leq G^* &= T_1 + \cdots + T_k + \textit{some of } T_{k+1} \\ &\leq G^* + 4G^* \\ &\leq 5G^* \end{aligned}$$

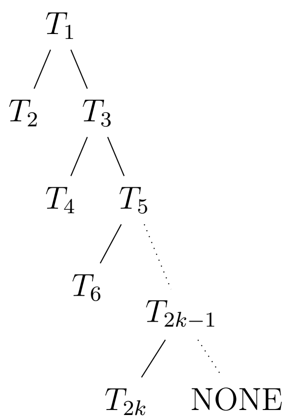
5. (a) **Collaborators: Study Group Members**
The pseudo code of algorithm is as below.

Algorithm 3 DynamicProgramming

Let C[n,m] be the dp table
C[0,k] $\leftarrow 0\ \forall k \geq 0$
C[j,0] $\leftarrow \infty \forall j > 0$
C[j,k] $= 0\ \forall j < 0, k \geq 0$
for $j = 1 \ldots n$ **do**
  for $k = 1 \ldots m$ **do**
    C[j,k] $= \min(c_j + C[j-1, k-n[j]], C[j-1,k])$
    B[j,k] $=$ 0 or 1 dependent on C[j,k]
  end for
end for
return C[m,n]

(b) **Collaborators: None**

Rewrite the relationship of T_i as the following statement. Let T_i be connected with T_{i+1} and T_{i+2}, where $i \in odd$. T_j will be a leaf if $j \in even$. The relation tree might be as below.



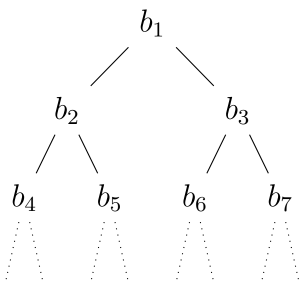
Let $T[i]$ be the strategy for selecting node i, $T'[i]$ be the strategy for not selecting node i, $A[i] = \min \sum c_i$ from $T[i]$, $A'[i] = min \sum c_i$ from $T'[i]$, we have

$$A'[i] = A[i+1] + A[i+2]$$
$$A[i] = \min\{A'[i],\ c_i + A'[i+1] + A'[i+2]\}$$

Note that if i is even, $A'[i] = \{\}$ and $A[i] = c_i$.

6. **Collaborators: None**

The relationship for each bread is as below.



Let $T[i]$ be the strategy for selecting node i, $T'[i]$ be the strategy for not selecting node i, $A[i] = \max \sum P_i$ from $T[i]$, $A'[i] = \max \sum P_i$ from $T'[i]$, we have

$$\begin{aligned} A'[i] &= A[2i] + A[2i+1] \\ A[i] &= \max\{A'[i],\ P_i + A'[2i] + A'[2i+1]\} \end{aligned}$$