① $T(n) = 16T\left(\frac{n}{4}\right) + n!$

$a=16 \quad b=4 \quad f(n)=n!$

$n^{\log_b a} = n^{\log_4 16} = n^2$ $\quad f(n) \in \Omega(n^2)$ $\quad$ so. $T(n) \in \Theta(f(n))$

$\boxed{T(n) \in \Theta(n!)}$

---

b) $T(n) = \sqrt{2}\,T\left(\frac{n}{4}\right) + \log n$

$a=\sqrt{2} \quad b=4 \quad n^{\log_b a} = n^{\log_4 \sqrt{2}} = n^{0,25}$

$\log n = n^{0,25-\varepsilon}$

$T(n) = \Theta(n^{\log_b a}) \Rightarrow$

$\boxed{T(n) \in \Theta(n^{0,25})}$

---

c) $T(n) = 8T\left(\frac{n}{2}\right) + 4n^3$

$a=8 \quad b=2 \quad d=3 \quad 8 = 2^3 \quad T(n) = \Theta n^d \log n$

$\boxed{T(n) \in \Theta n^3 \log n}$

---

d) $T(n) = 64\,T\left(\frac{n}{8}\right) - n^2 \log n$

Cannot be solveable because $f(n)$ is not non-decreasing

---

e) $T(n) = 3T\left(\frac{n}{3}\right) + \sqrt{n}$

$a=3 \quad b=3 \quad d=\frac{1}{2} \quad 3 > \sqrt{3} \quad T(n) = \Theta(n^{\log_b a})$

$\boxed{T(n) = \Theta(n^{\log_3 3}) = \Theta(n)}$

---

f) $T(n) = 2^n T\left(\frac{n}{2}\right) - n^n$

d is not polynomial & a is not constant & $f(n)$ is not positive

$\varnothing$

---

g) $T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{\log n} \Rightarrow 3T(n/3) + (n \cdot \log^{-1} n)$

$a=3 \quad b=3 \quad d=1 \quad k=-1$

If $f(n) \in \Theta(n^{\log_b a} \log^k n)$; $k \geq 0$ $\quad k<0$

$T(n) \in \Theta(n^{\log_b a} \log^{k+1} n)$ $\quad$ so doesn't apply theorem

so non-polynomial difference between $f(n)$ and $n^{\log_b a}$

2)

a) $T(n) = 9T(\frac{n}{3}) + n^2$ $\quad a=9,\ b=3,\ d=2$ $\quad 9 = 3^2$

$T(n) = \Theta(n^d \log n)$

$T(n) \in \Theta(n^2 \log n)$

b) $T(n) = 8T(\frac{n}{2}) + n^3$ $\quad a=8,\ b=2,\ d=3$ $\quad 8 = 2^3$

$T(n) = \Theta(n^d \log n)$

$T(n) \in \Theta(n^3 \log n)$

c) $T(n) = 2T(\frac{n}{4}) + \sqrt{n}$ $\quad a=2,\ b=4,\ d=\frac{1}{2}$ $\quad 2 = \sqrt{4}$

$T(n) = \Theta(n^d \log n)$

$T(n) \in \Theta(\sqrt{n} \log n)$

I prefer to choose third (c) algorithm because it is faster and clear algorithm than others.

3)

a) 

Merge sort is a divide an conquer algoritm. We are not doing any comparition until dividing array size equal to 1 from branch to branch.

i)

5	1	7	3	6	2	8	4

5	1	7	3

5	1

5

$\frac{n}{2^k} = 1$

$k = \log_2 n$

there are $\log_2 n$ level

$T(n) = 2(\frac{T}{2}) + n$

$a=2 \quad b=2 \quad d=1 \quad 2 = 2^1$ ✓

$T(n) \in \Theta(n \log n)$

```
if size > 1
    hsize = size/2
    ltable = copyAr(0, hsize-1)
    rtable = copyAr(hsize, size)
2 { merge (ltable)
  { merge (rtable)
    concanate (ltable, rtable)
```

1	5

3	7

- insertion take n time when two list concanate.

ii)

1	2	3	4	5	6	7	8

1	2	3	4

1	2

1

$\log_2 n$ level

- concatanate just using 1 comparison so its constan time

$T(n) = 2(\frac{T}{2}) + n^0 + 1$

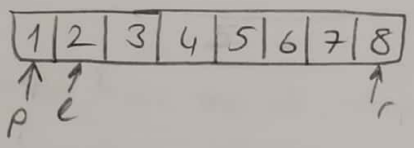
$a=2 \quad b=2 \quad d=0 \quad 2 > 2^0$

$T(n) = (n^{\log_b a})$

$T(n) \in \Theta(n^{\log_2 2}) \Rightarrow T(n) \in \Theta(n)$

b)

i)



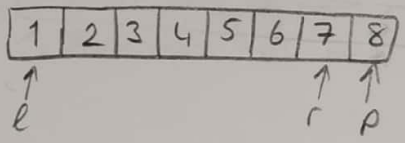
max swap = ?
pivot = first element

Other swapping operation does not require this much swapping because the partition function moving pivot point very fastly then it is realize the list sorted. And also this case is the worst case for the quick sort algorithm

swap number = 7

L[0] swap L[0]
L[1] " L[1]
L[2] " L[2]
⋮ ⋮
L[6] " L[6]

ii)



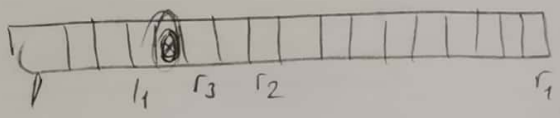
min swap = ?
pivot = last element

When the pivot is the last point that means every item in the list less than pivot so there will less swapping other list permutation. There will be no again swapping operation.

swap number = 7

L[7] swap L[7]
L[6] " L[6]
⋮ ⋮
L[1] " L[1]

4)



- binary search 0 in a sorted list

$T(1) = 1$

(I am consider the '0' will be always in my list)

$\rightarrow T(n) = 1 \cdot T\left(\frac{n}{2}\right) + n^0 \cdot 1$

$\vdots$

$T\left(\frac{n}{2^{k-1}}\right) = T\left(\frac{n}{2^k}\right) + 1 \cdot k \rightarrow$ final step

log/ $\frac{n}{2^k} = 1 \Rightarrow n = 2^k \quad \log n = k$

$T(n) = T\left(\frac{n}{2^{\log_2 n}}\right) + 1 \cdot \log n$

$T(n) = \underbrace{T(1)}_{1} + \log n$

$T(n) \in \Theta(\log n)$

---

5) a)

boxes

a	b	c	d	e	f	...

gifts

c	a	e	b	f	...

```
function partition ( boxes, gifts, start, end):
    while 0<= end & gifts [end] != boxes [start]
        end -= 1
    end while
    swap (gifts, start, end)
    return end
end function

function quick (boxes, gifts, start, end):
    if start <= end
        p = partition (boxes, gifts, start, end)
        quick (boxes, gifts, start, p-1)
        quick (boxes, gifts, p+1, end)
    return gifts
end function
```

b) Master:

$T(n) = 2\left(\frac{T}{2}\right) + n$

$a=2 \quad b=2 \quad d=1$

$a ? b^d$

$2 = 2^1$

$?=$

$T(n) = (n^d \log n)$

$\boxed{T(n) \in \Theta(n \log n)}$

Reccurence Relation)

$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n$

$T(n) = 2 \cdot \left[2 \cdot T\left(\frac{n}{4}\right) + n\right] + n$

$T(1) = 1$

$\frac{n}{2^k} = 1 \qquad 2^k = n$

$\boxed{k = \log_2 n}$

$T(k) = 2^k T\left(\frac{n}{2^k}\right) + k \cdot n$

$\Rightarrow 2^{\log_2 n} \cdot T(1) + n \log_2 n \Rightarrow T(n) = n + n \log n$

$\boxed{T(n) \in \Theta(n \log n)}$

My algorithm boxes are stabil change the position of gifts according to the boxes which is fit the sift.

as an example:

boxes size = [3, 5, 2, 1, 4]

gifts before = [5, 3, 2, 4, 1]

search the sift for the fit the box than swap the corresponding place.

after quick sort gift list = [3, 5, 2, 1, 4]

```
mbulucay@DESKTOP-P69IKKT: /mnt/c/Users/M Bedir ULUCAY/Desktop/algo/hws/hw2
mbulucay@DESKTOP-P69IKKT:/mnt/c/Users/M Bedir ULUCAY/Desktop/algo/hws/hw2$ python3 matchGiftBox_1901042697.py
boxes list : [4, 3, 1, 2, 5]
gifts list (before sorted according to the boxes): [3, 4, 2, 1, 5]
gifts list (after sorted according to the boxes): [4, 3, 1, 2, 5]
mbulucay@DESKTOP-P69IKKT:/mnt/c/Users/M Bedir ULUCAY/Desktop/algo/hws/hw2$
```