

Q1 answer:

The code is copied from the .cpp text in Ubuntu:

```
#include<iostream>
#include<ctime>
#include<limits.h>
using namespace std;

void insertion_sort(int v[], int n)
{
    int value;
    int i, j;
    for (i = 1; i < n; i++)
    {
        value = v[i];
        j = i - 1;
        while (j >= 0 && v[j] > value)
        {
            v[j + 1] = v[j];
            j--;
        }
        v[j + 1] = value;
    }
}
```

```
void merge_sort(int v[], int n)
{
    int A[50];
    int B[50];
    int i, j, k;
    int value;
    for (i = 1; i <= (n / 2); i++)
    {
        A[i] = v[i - 1];
        B[i] = v[i + n / 2 - 1];
    }
    if ((n % 2) == 0)
    {
        B[i] = INT_MAX;
        A[i] = INT_MAX;
    }
    else
```

```

{
    B[i] = v[n - 1];
    B[i + 1] = INT_MAX;
    A[i] = INT_MAX;
}
for (i = 2; i <= (n / 2); i++)
{
    value = A[i];
    j = i - 1;
    while (j >= 0 && A[j] > value)
    {
        A[j + 1] = A[j];
        j--;
    }
    A[j + 1] = value;
}
for (i = 2; i <= ((n+1) / 2); i++)
{
    value = B[i];
    j = i - 1;
    while (j >= 0 && B[j] > value)
    {
        B[j + 1] = B[j];
        j--;
    }
    B[j + 1] = value;
}
i = 1;
j = 1;
for (k = 0; k < n; k++)
{
    if (A[i] <= B[j])
    {
        v[k] = A[i];
        i++;
    }
    else
    {
        v[k] = B[j];
        j++;
    }
}
}

```

```

void print_vector(int v[], int n)
{
    int i;
    cout << "Vector:";
    for (i = 0; i < n; i++)
        cout << " " << v[i];
    cout << endl;
}

int main()
{
    int v[22];
    int u[22];
    int i;
    double n1, n2;
    clock_t start, end, start1, end1;
    for (i = 0; i < 22; i++)
    {
        v[i] = 22 - i;
        u[i] = 22 - i;
    }
    print_vector(v, 22);
    start = clock();
    insertion_sort(v, 22);
    end = clock();
    n1 = (double)(end - start) / (double)(CLOCKS_PER_SEC)*(double)(1000.000000);
    cout << "The running time of the 22 size input in insertion sort is "<<n1<<" ms"<< endl;
    print_vector(v, 22);
    print_vector(u, 22);
    start1=clock();
    merge_sort(u, 22);
    end1=clock();
    n2 = (double)(end1 - start1) / (double)(CLOCKS_PER_SEC)*(double)(1000.000000);
    cout << "The running time of the 22 size input in merge sort is "<<n2<<" ms"<<
endl;
    print_vector(u, 22);
    return 0;
}

```

Through the code and continuing to change some values, I can get when $n=22$, the merge sort will beat the insertion sort. As you can see In the picture below, the size of input indicates the number of n in an

array. I can change the value of n in the main function. When $n > 22$, the running time of insertion sort will be more than that of merge sort.

At first, I would do the experiment in Windows 10. After some trials, I found that the base of CPU time is 'ms' which is not precise enough. So at last I use ubuntu to show it.

As you see, when $n=22$, the running time of insertion sort is 0.002 ms. The running time of merge sort is 0.001 ms. All inputs are set in the worst-case.

```
Activities Terminal
eece@ubuntu: ~/source
eece@ubuntu:~$ cd source
eece@ubuntu:~/source$ ls
a.out netanim-3.108 ns-3.31 pybindgen test2.cpp
eece@ubuntu:~/source$ g++ test2.cpp
eece@ubuntu:~/source$ ./a.out
Vector: 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
The running time of the 19 size input in insertion sort is 0.002 ms
Vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Vector: 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
The running time of the 19 size input in merge sort is 0.002 ms
Vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
eece@ubuntu:~/source$ ls
a.out netanim-3.108 ns-3.31 pybindgen test2.cpp
eece@ubuntu:~/source$ g++ test2.cpp
eece@ubuntu:~/source$ ./a.out
Vector: 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
The running time of the 20 size input in insertion sort is 0.002 ms
Vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Vector: 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
The running time of the 20 size input in merge sort is 0.002 ms
Vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
eece@ubuntu:~/source$
```

```
Activities Terminal
eece@ubuntu: ~/source
eece@ubuntu:~$ cd source
eece@ubuntu:~/source$ ls
a.out netanim-3.108 ns-3.31 pybindgen test2.cpp
eece@ubuntu:~/source$ g++ test2.cpp
eece@ubuntu:~/source$ ./a.out
Vector: 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
The running time of the 21 size input in insertion sort is 0.002 ms
Vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
Vector: 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
The running time of the 21 size input in merge sort is 0.002 ms
Vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
eece@ubuntu:~/source$ ls
a.out netanim-3.108 ns-3.31 pybindgen test2.cpp
eece@ubuntu:~/source$ g++ test2.cpp
eece@ubuntu:~/source$ ./a.out
Vector: 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
The running time of the 22 size input in insertion sort is 0.002 ms
Vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Vector: 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
The running time of the 22 size input in merge sort is 0.001 ms
Vector: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
eece@ubuntu:~/source$
```

Q2 answer:

Initial array: 10,5,7,9,8,3

Iteration of insertion sort:

5,10,7,9,8,3

5,7,10,9,8,3

5,7,9,10,8,3

5,7,8,9,10,3

3,5,7,8,9,10

The last array is the result of insertion sort.

Initial array:10,5,7,9,8,3

So, the iteration of quick sort is:

10,5,7,9,8,3

3,5,7,9,8,10

3,5,7,9,8,10

3,5,7,9,8,10

3,5,7,8,9,10

Q3 answer:

$n + 3 \in \Omega(n)$ is true.

$n + 3 \in O(n^2)$ is true.

$n + 3 \in \Theta(n^2)$ is false, $n + 3 \in O(n^2)$

$2^{n+1} \in O(n + 1)$ is false, $2^{n+1} \in \Omega(n + 1)$

$2^{n+1} \in \Theta(2^n)$ is true.

Q4 answer:

$T(n) = 8T\left(\frac{n}{2}\right) + n$, for some $\varepsilon > 0$, $n = O(n^{3-\varepsilon})$, so $T(n) = \theta(n^3)$.

$T(n) = 8T\left(\frac{n}{2}\right) + n^2$, for some $\varepsilon > 0$, $n^2 = O(n^{3-\varepsilon})$, so $T(n) = \theta(n^3)$.

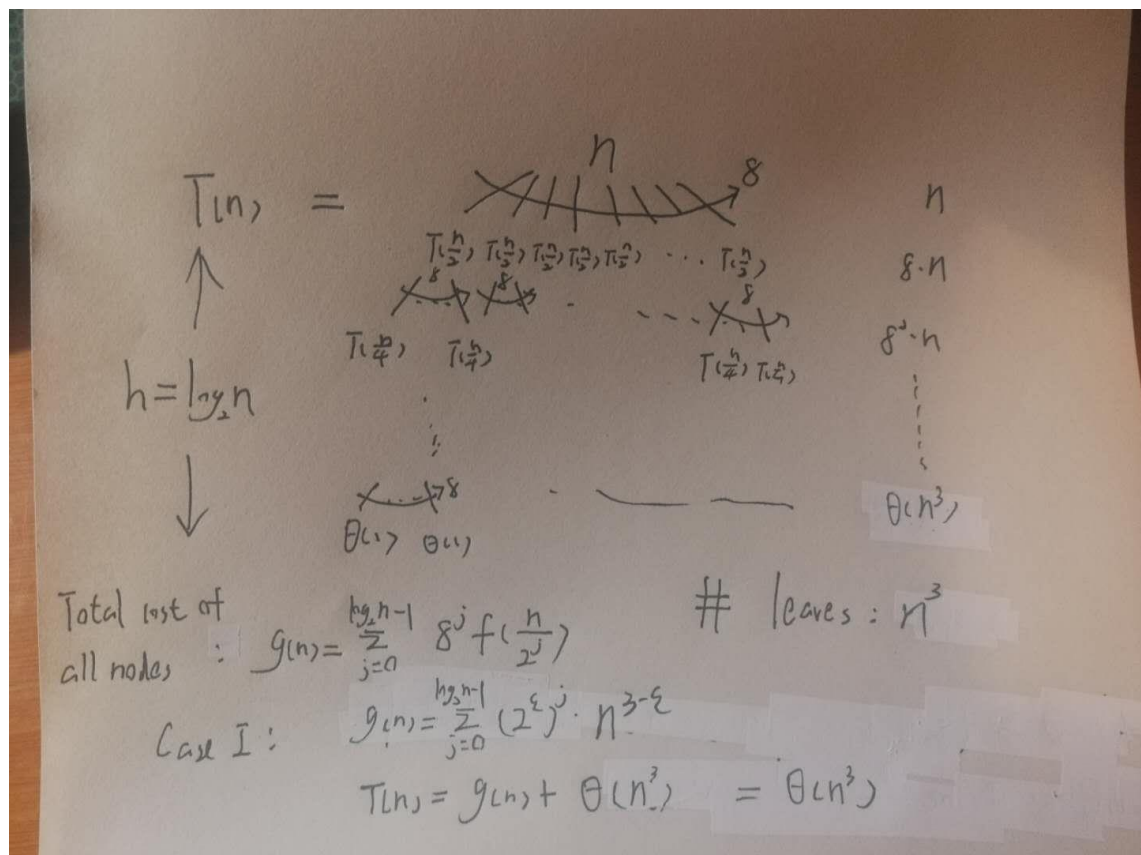
$T(n) = 8T\left(\frac{n}{2}\right) + n^3$, if $k = 0$, $n^3 = \theta(n^3)$, so $T(n) = \theta(n^3 \log n)$.

$T(n) = 8T\left(\frac{n}{2}\right) + n^4$, for some $\varepsilon > 0$, $n^4 = \Omega(n^{3+\varepsilon})$, and for some

$\varepsilon^l > 0$, $\frac{n^4}{2} \leq (1 - \varepsilon^l)n^4$, so $T(n) = \theta(n^4)$

Q5 answer:

The recursion tree is as the picture below:



Prove:

Guess $T(n) = \theta(n^3)$;

Assume $T(n) \leq c_1 n^3 - c_2 n^2$ for all $n > n_0$ and $T(n) \geq c_3 n^3 + c_4 n^2$ for all $n > n_0$;

For the former $T(n) = 8T\left(\frac{n}{2}\right) + n \leq 8 * c_1 * \left(\frac{n}{2}\right)^3 - 8 * c_2 * \left(\frac{n}{2}\right)^2 + n$
 $= c_1 n^3 - c_2 n^2 - n(c_2 n - 1)$ and if $c_2 = 1, n_0 = 1, n(c_2 n - 1) > 0$, so
 $T(n) \leq c_1 n^3 - c_2 n^2, T(n) = O(n^3)$

For the latter $T(n) = 8T\left(\frac{n}{2}\right) + n \geq 8 * c_3 * \left(\frac{n}{2}\right)^3 + 8 * c_4 * \left(\frac{n}{2}\right)^2 + n$
 $= c_3 n^3 + c_4 n^2 + n(c_4 n + 1)$ and if $c_4 = 1, n_0 = 1, n(c_4 n + 1) > 0$, so

$$T(n) \geq c_3 n^3 + c_4 n^2, \quad T(n) = \Omega(n^3)$$

For base case, $T(1) \leq c_1 - c_2$ and $T(1) \geq c_3 + c_4$ for that c_1 is much larger than c_2 and c_3, c_4 are very small.

In summary, $T(n) = \theta(n^3)$