Assignment 1

Zimin Guo

Spring 2016 Professor: P Thananjeyan

Written Assignments

1. Analysis of algorithms

(a) Order the following functions by growth rate: N, $\sqrt{N}$, $N^{1.5}$, N^2, $N \log N$, $N \log \log N$, $N \log (N^2)$, $2/N$, 2^N, $2^{N/2}$, 37, $N^2 \log N$, N^3. Indicate which grow at the same rate.

Solution: From high to low: 2^N, $2^{N/2}$, N^3, $N^2 \log N$, N^2, $N^{1.5}$, $N \log (N^2)$, $N \log N$, $N \log \log N$, N, $\sqrt{N}$, 37, $2/N$. $N \log (N^2)$ and $N \log N$ grow at the same rate.

(b) For each of the following program fragments, give an analysis of the running time.

```
a. sum = 0;
    for (i = 0, i < n; ++i)
         ++sum;

b. sum = 0;
    for (i = 0; i < n; ++i)
            for (j = 0; j < n; ++j)
                ++sum;

c. sum = 0;
    for (i = 0; i < n; ++i)
            for (j = 0; j < n * n; ++j)
                ++sum;

d. sum = 0;
    for (i = 0; i < n; ++i)
            for (j = 0; j < i; ++j)
                ++sum;

e. sum = 0;
    for (i = 0; i < n; ++i)
            for (j = 0; j < i * i; ++j)
                for (k = 0; k < j; ++k)
                ++sum;
```

Solution:

a. $O(n)$
b. $O(n^2)$
c. $O(n^3)$
d. $O(n^2)$
e. $O(n^5)$

(c) An algorithm takes 0.5 ms for an input size of 100. How large will it take for input sizes of 500, 1,000, 10,000, 100,000, if the running time is the following:

a. Linear

b. $O(N \log N)$

c. Quadratic

d. Cubic

e. Exponential

Solution:

a. Linear $N = 500/100 \cdot 0.5 = 2.5ms$, $N = 10 \cdot 0.5 = 5ms$, $N = 100 \cdot 0.5 = 50ms$, $N = 1000 \cdot 0.5 = 500ms$

b. $O(N \log N)$ $N = \frac{500 \log 500}{100 \log 100} \cdot 0.5 = 3.3737ms$, $N = \frac{1000 \log 1000}{100 \log 100} \cdot 0.5 = 7.5ms$, $N = \frac{10000 \log 10000}{100 \log 100} \cdot 0.5 = 100ms$, $N = \frac{100000 \log 100000}{100 \log 100} \cdot 0.5 = 1250ms$,

c. Quadratic $N = \frac{500^2}{100^2} \cdot 0.5 = 12.5ms$, $N = \frac{1000^2}{100^2} \cdot 0.5 = 50ms$, $N = \frac{10000^2}{100^2} \cdot 0.5 = 5000ms$, $N = \frac{100000^2}{100^2} \cdot 0.5 = 5 \times 10^5 ms$,

d. Cubic $N = \frac{500^3}{100^3} \cdot 0.5 = 62.5ms$, $N = \frac{1000^3}{100^3} \cdot 0.5 = 500ms$, $N = \frac{10000^3}{100^3} \cdot 0.5 = 5 \times 10^5 ms$, $N = \frac{100000^3}{100^3} \cdot 0.5 = 5 \times 10^8 ms$,

e. Exponential $N = \frac{2^{500}}{2^{100}} \div 2 = 2^{399} ms$, $N = \frac{2^{1000}}{2^{100}} \div 2 = 2^{899} ms$, $N = \frac{2^{10000}}{2^{100}} \div 2 = 2^{9899} ms$, $N = \frac{2^{100000}}{2^{100}} \div 2 = 2^{99899} ms$,

2. Union-Find

Show the contents of the id[] array and the number of times array is accessed for each input pair from the following sequence of instructions: union(1, 2), union(3, 4), union(1, 7), union(3, 6), union(8, 9), union(1, 8), union(3, 10), union(3, 11), union(3, 12), union(3, 13), union(14, 15), union(16, 0), union(14, 16), union(1, 3), union(1, 14) when the union are

a. Quick-Find

b. Quick-Union

c. Weighted Quick-Union

Solution:

a. Quick-Find

union	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(1, 2)	0	2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(3, 4)	0	2	2	4	4	5	6	7	8	9	10	11	12	13	14	15	16
(1, 7)	0	7	7	4	4	5	6	7	8	9	10	11	12	13	14	15	16
(3, 6)	0	7	7	6	6	5	6	7	8	9	10	11	12	13	14	15	16
(8, 9)	0	7	7	6	6	5	6	7	9	9	10	11	12	13	14	15	16
(1, 8)	0	9	9	6	6	5	6	9	9	9	10	11	12	13	14	15	16
(3, 10)	0	9	9	10	10	5	10	9	9	9	10	11	12	13	14	15	16
(3, 11)	0	9	9	11	11	5	11	9	9	9	11	11	12	13	14	15	16
(3, 12)	0	9	9	12	12	5	12	9	9	9	12	12	12	13	14	15	16
(3, 13)	0	9	9	13	13	5	13	9	9	9	13	13	13	13	14	15	16
(14, 15)	0	9	9	13	13	5	13	9	9	9	13	13	13	13	15	15	16
(16, 0)	0	9	9	13	13	5	13	9	9	9	13	13	13	13	15	15	0
(14, 16)	0	9	9	13	13	5	13	9	9	9	13	13	13	13	0	0	0
(1, 3)	0	13	13	13	13	5	13	13	13	13	13	13	13	13	0	0	0
(1, 14)	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0

Array access: 20, 20, 21, 21, 20, 22, 22, 23, 24, 25, 20, 20, 21, 24, 31

b. Quick-Union

union	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(1, 2)	0	2	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(3, 4)	0	2	2	4	4	5	6	7	8	9	10	11	12	13	14	15	16
(1, 7)	0	2	7	4	4	5	6	7	8	9	10	11	12	13	14	15	16
(3, 6)	0	2	7	4	6	5	6	7	8	9	10	11	12	13	14	15	16
(8, 9)	0	2	7	4	6	5	6	7	9	9	10	11	12	13	14	15	16
(1, 8)	0	2	7	4	6	5	6	9	9	9	10	11	12	13	14	15	16
(3, 10)	0	2	7	4	6	5	10	9	9	9	10	11	12	13	14	15	16
(3, 11)	0	2	7	4	6	5	10	9	9	9	11	11	12	13	14	15	16
(3, 12)	0	2	7	4	6	5	10	9	9	9	11	12	12	13	14	15	16
(3, 13)	0	2	7	4	6	5	10	9	9	9	11	12	13	13	14	15	16
(14, 15)	0	2	7	4	6	5	10	9	9	9	11	12	13	13	15	15	16
(16, 0)	0	2	7	4	6	5	10	9	9	9	11	12	13	13	15	15	0
(14, 16)	0	2	7	4	6	5	10	9	9	9	11	12	13	13	15	0	0
(1, 3)	0	2	7	4	6	5	10	9	9	13	11	12	13	13	15	0	0
(1, 14)	0	2	7	4	6	5	10	9	9	13	11	12	13	0	15	0	0

Array access: 3, 3, 5, 5, 3, 9, 7, 9, 11, 13, 3, 3, 7, 21, 15

c. Weighted Quick-Union

union	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(1, 2)	0	1	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(3, 4)	0	1	1	3	3	5	6	7	8	9	10	11	12	13	14	15	16
(1, 7)	0	1	1	3	3	5	6	1	8	9	10	11	12	13	14	15	16
(3, 6)	0	1	1	3	3	5	3	1	8	9	10	11	12	13	14	15	16
(8, 9)	0	1	1	3	3	5	3	1	8	8	10	11	12	13	14	15	16
(1, 8)	0	1	1	3	3	5	3	1	1	8	10	11	12	13	14	15	16
(3, 10)	0	1	1	3	3	5	3	1	1	8	3	11	12	13	14	15	16
(3, 11)	0	1	1	3	3	5	3	1	1	8	3	3	12	13	14	15	16
(3, 12)	0	1	1	3	3	5	3	1	1	8	3	3	3	13	14	15	16
(3, 13)	0	1	1	3	3	5	3	1	1	8	3	3	3	3	14	15	16
(14, 15)	0	1	1	3	3	5	3	1	1	8	3	3	3	3	14	14	16
(16, 0)	16	1	1	3	3	5	3	1	1	8	3	3	3	3	14	14	16
(14, 16)	16	1	1	3	3	5	3	1	1	8	3	3	3	3	14	14	14
(1, 3)	16	3	1	3	3	5	3	1	1	8	3	3	3	3	14	14	14
(1, 14)	16	3	1	3	3	5	3	1	1	8	3	3	3	3	3	14	14

Array access: 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8