H1.

Yingzhu (Jacqueline) Zhang

Students I have consulte with for this assignment:
1. William Bench
2. Nathanel Marshall
3. Yussre ElBardicy
4. William Theuer
5. Maurice Hayward
6. Stephanie Bramlett
7. Diego Lanao

1. 1.4.4 **Solution:**

```
public class twoSum
{
    public static in count(int[] a)
    {
    int N = a.length;
    int cnt = 0;

    for (int i = 0; i ¡N; i ++)
        for (int j = i+1; j ¡ N; j++)
            if (a[i] + a[j] == 0)
                cnt ++;

    return cnt;
    }

    public static void main(String[] args)
    {
        int[] a = In.readInts(args[0]);
        StdOut.println(count(a))
    }
}
```

statement block	time in seconds	frequency	total time
D	t_0	x(depends on input)	$t_0 x$
C	t_1	$N^2/2 - N/2$	$t_1(N^2/2 - N/2)$
B	t_2	N	$t_2 N$
A	t_3	1	t_3

grand total	$(t_1/2)N^2$ $(-t_1/2 + t_2)N$ $t_3 + t_0 x$
tilde approximation	$\sim (t_1/2)N^2$ (assuming x is small)
order of growth	N^2

2. 1.4.5 **Solution:**

   (a) $\sim N$

   (b) ~ 1

   (c) $1 + 2/N + 1/N + 2/N^2 = 1 + 1/N + 2/N + 2/N^2 \quad \sim 1$

   (d) $\sim 2N^3$

   (e) $lg2N - lgN = (lg2 + lgN)/lgN = 1/lgN + 1 \quad \sim 1$

   (f) $lg(N^2+1)/lgN = lg(1(N^2+1))/lgN$
   $= (lg(N^2(1 + 1/N^2)))/lgN$
   $= (2lgN + lg(1 + 1/1/N^2)))/lgN$
   $= 2 + ((lg(1 + 1/N^2)/lgN)) \quad \sim 2$

   (g) $\sim N^{100}/2^N$

3. 1.4.9 **Solution:**

   According to Doubling Ratio Proposition:
   If $T(N) \sim aN^b lgN$, then $T(2N)/T(N) =$
   $(a(2N)^b lg(2N))/aN^b lgN$
   $(2^b N^b lg(2N))/N^b lgN$
   $(2^b lg(2N))/lgN$
   $(2^b(lg2lgN))/lgN = 2^b(1)$
   $\therefore \sim 2^b$

   Because the running time for problems of size $N_0 = T$, and the doubling factor converges to 2^b then
   According to Doubling Ratio Proposition,
   $aN_0^b lgN_0 = T$, $a = T/(N_0^b lgN_0)$
   $\therefore$ the running time of a program for a problem of size N: $\sim T(N^b lgN)/(N_0^b lgN_0)$

4. 1.5.1 **Solution:**

   Through quick-find, we get:

p-q	0	1	2	3	4	5	6	7	8	9	number of times the array is accessed
9-0	0	1	2	3	4	5	6	7	8	0	2+10+1 = 13
3-4	0	1	2	4	4	5	6	7	8	0	2+10+1 = 13
5-8	0	1	2	4	4	8	6	7	8	0	2+10+1 = 13
7-2	0	1	2	4	4	8	6	2	8	0	2+10+1 = 13
2-1	0	1	1	4	4	8	6	1	8	0	2+10+1+1 = 14
5-7	0	1	1	4	4	1	6	1	1	0	2+10+1+1 = 14
0- 3	4	1	1	4	4	1	6	1	1	4	2+10+1+1 = 14
4- 2	4	1	1	1	1	1	6	1	1	1	2+10+1+2 = 15

See attached sheet of paper for demonstrative diagrams.

5. 1.5.2 **Solution:**

   Through quick-union, we get:

	p-q	0	1	2	3	4	5	6	7	8	9	number of times the array is accessed
1)	9-0	0	1	2	3	4	5	6	7	8	0	2+1 = 3
2)	3-4	0	1	2	4	4	5	6	7	8	0	2+1 = 3
3)	5-8	0	1	2	4	4	8	6	7	8	0	2+1 = 3
4)	7-2	0	1	2	4	4	8	6	2	8	0	2+1 = 3
5)	2-1	0	1	1	4	4	8	6	2	8	0	2+1 = 3
6)	5-7	0	1	1	4	4	2	6	2	8	0	2+2+1 = 5
7)	0-3	4	1	1	4	4	2	6	2	8	0	1+2+1 = 4
8)	4-2	4	1	1	4	1	2	6	2	8	0	1+2+1 = 4

   See attached sheet of paper for demonstrative diagrams.

6. 1.5.3 **Solution:**

   Through weighted quick-union, we get:

	p-q	0	1	2	3	4	5	6	7	8	9	number of times the array is accessed
1)	9-0	9	1	2	3	4	5	6	7	8	9	2+1+1 = 4
2)	3-4	9	1	2	3	3	5	6	7	8	9	2+1+1 = 4
3)	5-8	9	1	2	3	3	5	6	7	5	9	2+1+1 = 4
4)	7-2	9	1	7	3	3	5	6	7	5	9	2+1+1 = 4
5)	2-1	9	7	7	3	3	5	6	7	5	9	2+1+1+1 = 5
6)	5-7	9	7	7	3	3	5	6	5	5	9	2+1+1 = 4
7)	0-3	9	7	7	9	3	5	6	5	5	9	2+2+1+1 = 6
8)	4-2	9	7	7	9	7	5	6	5	5	9	2+2+1+1 = 6

   See attached sheet of paper for demonstrative diagrams.

7. 1.5.5 10^9 sites–array size/number of components),
   10^6 input pairs,
   10^9 instructions per second–array accesses persecond
   Assume:each iteration of the inner for loop requires 10 machine instruction.
   We can find the minimus amount of time (in days) that would be required for quick-find to solve this dynamic connectivity problem by:

Solution:

For each input pair:
$find() : 2 machineinstructions$
$union() : 10 * (10^9 + 1) machineinstructions$
Then, for 10^6 pair, the total machine instructions = $2 * 10^6 + 10^7(10^9 + 1)$
Also, a computer is capable of executing $24 * 60^2 * 10^9 = 8.64 * 10^13$ machine instructions per day
$(2 * 10^6 + 10^7(10^9 + 1))/8.64 * 10^13 \approx 115.7407$
$\therefore \approx 116$ days

8. 1.5.9 **Solution:**

According to the proposition that the depth of any node in a forest built by weighted quick-union for N sites is at most lgN,
When $N = 10$, depth = $lg10(3 < lg10 < 4)$
But according to the diagram drawn in the attachement page, the depth of this problem is 4,
$\therefore$ it is impossible to ge the result by running weighted quick-union