Danny Tan

Dt1462

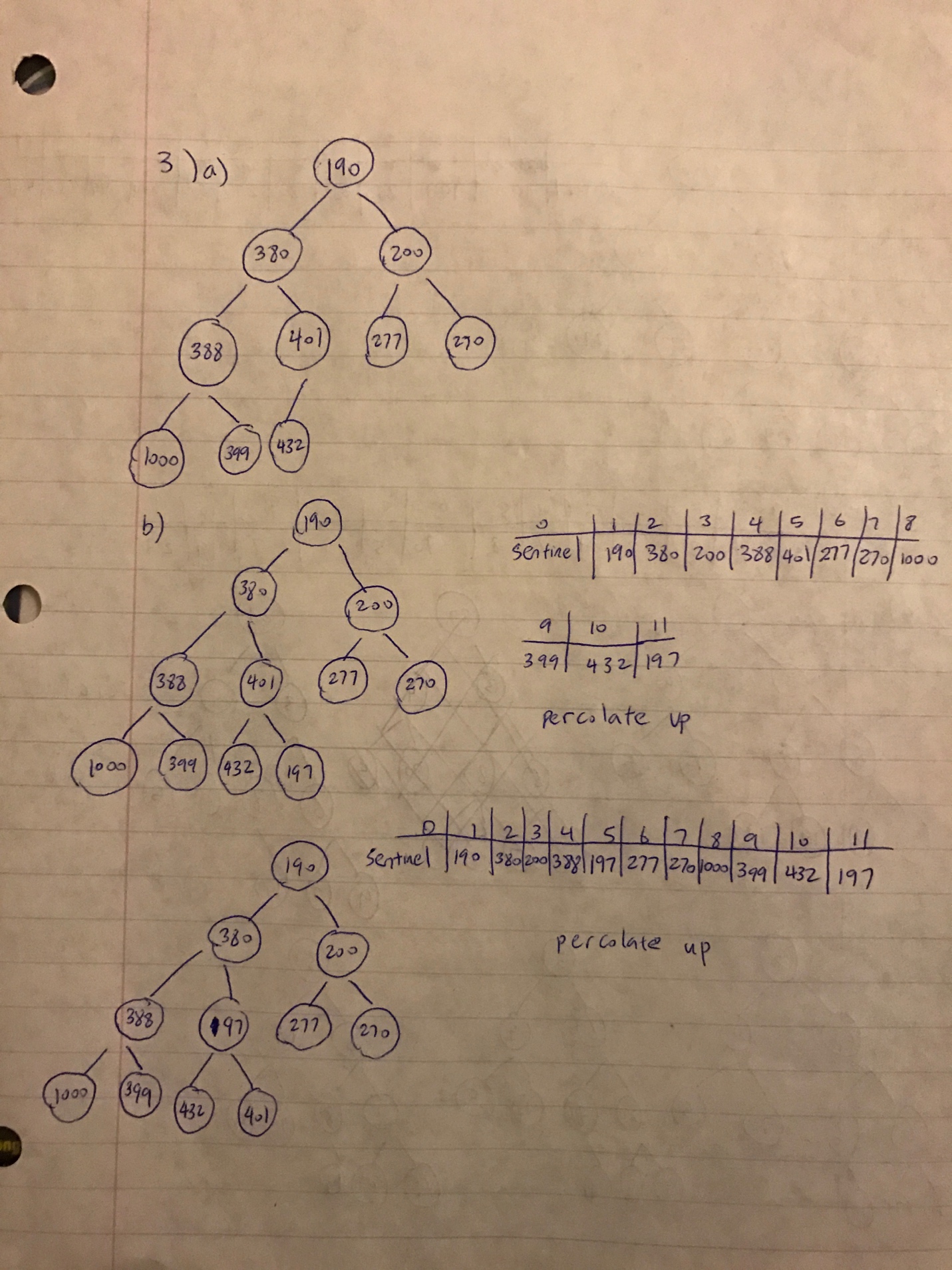
HW#11

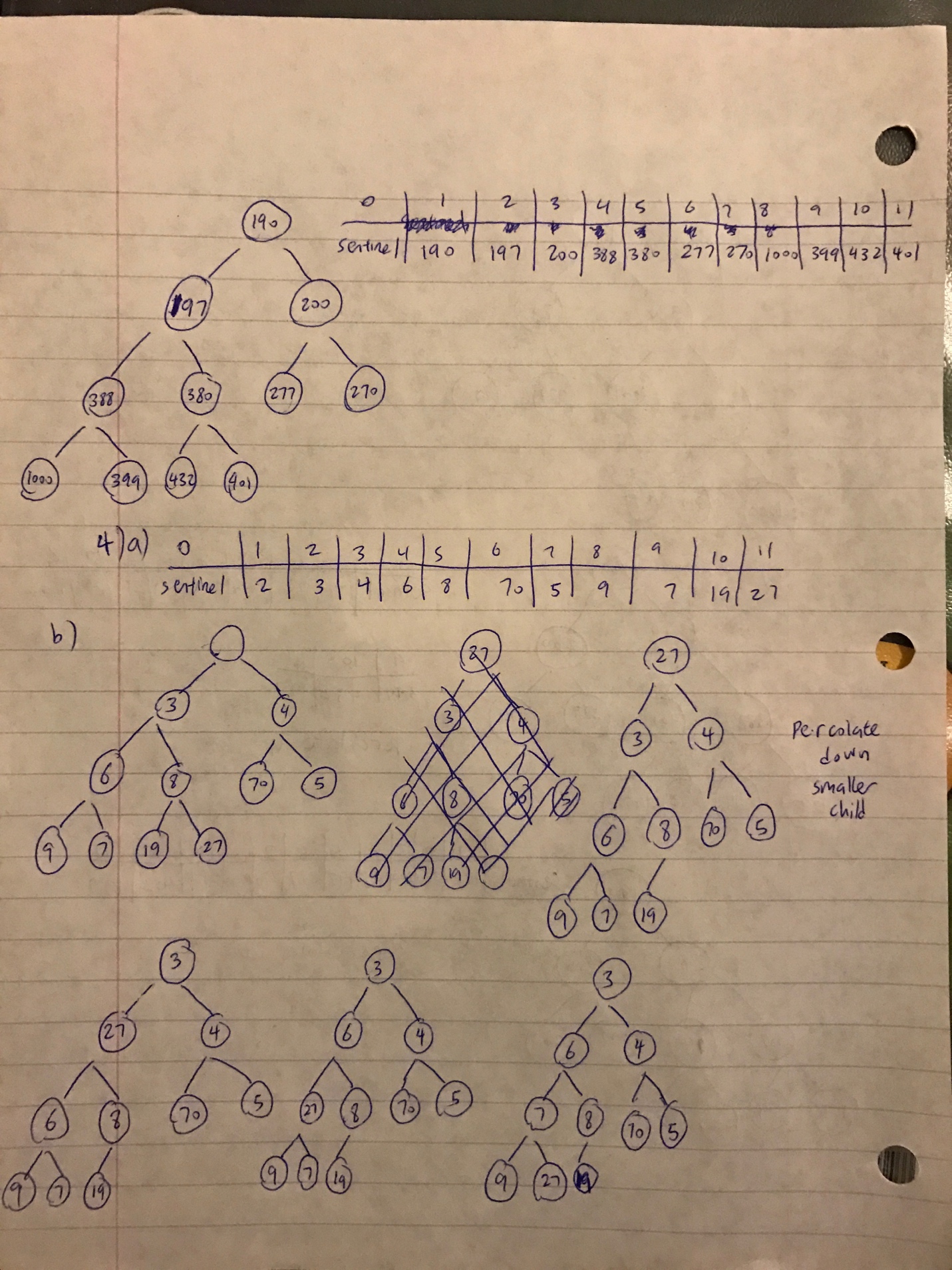
1) O(n+k)

2) Insert (x) : O(1)

deleteMin(): O(log(n))

findMin() O(1)





5)

template <class Comparable>

void BinaryHeap::insert( const Comparable & x ) {

if( theSize + 1 == array.size( ) )

array.resize( array.size( ) \* 2 + 1 ); // Percolate up int hole = ++theSize;

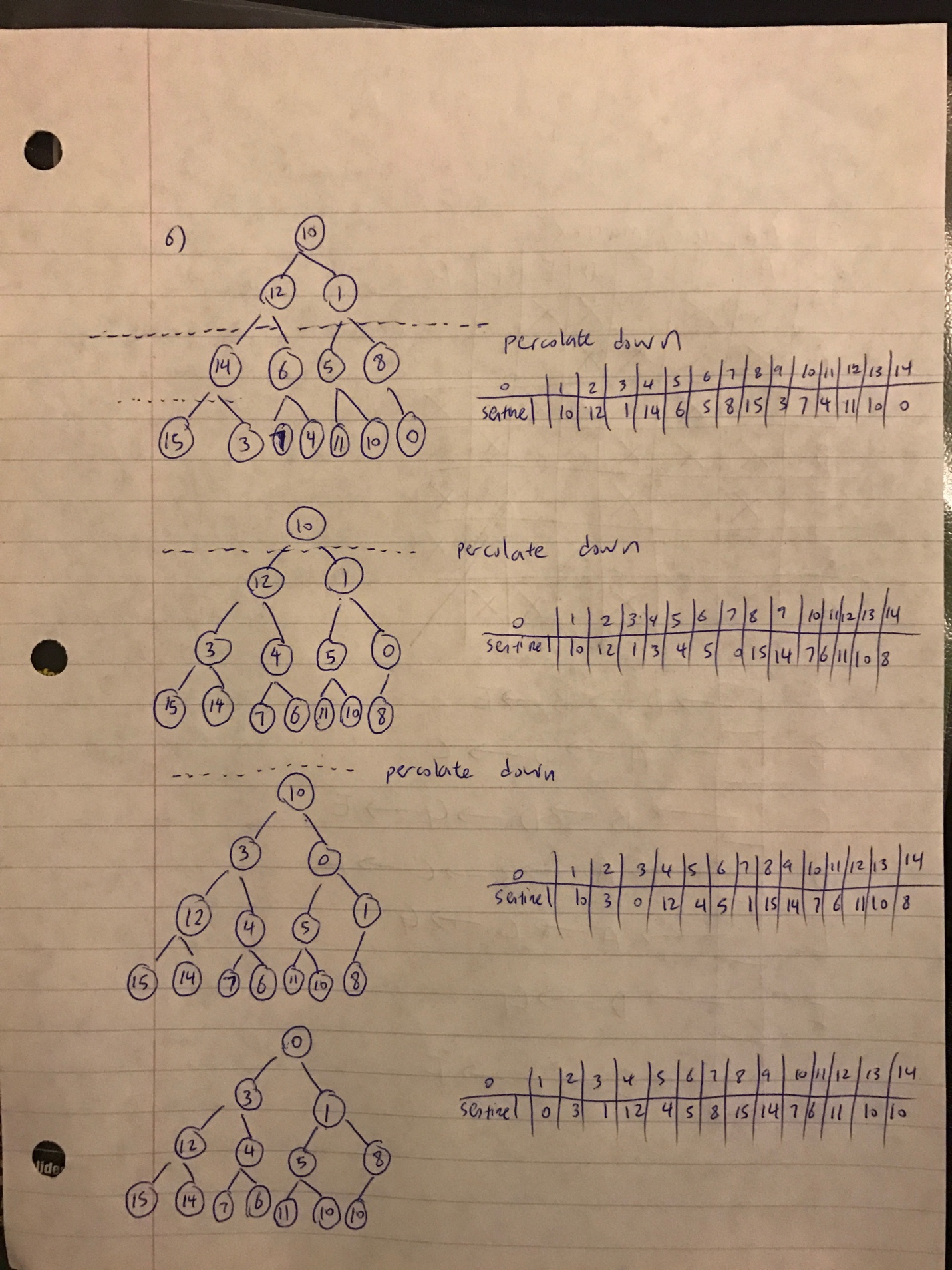
int hole = theSize ++;

for( ; x < array[ hole / 2 ]; hole /= 2 )

array[ hole ] = std::swap ( array[ hole / 2 ] );

array[ hole ] = x;

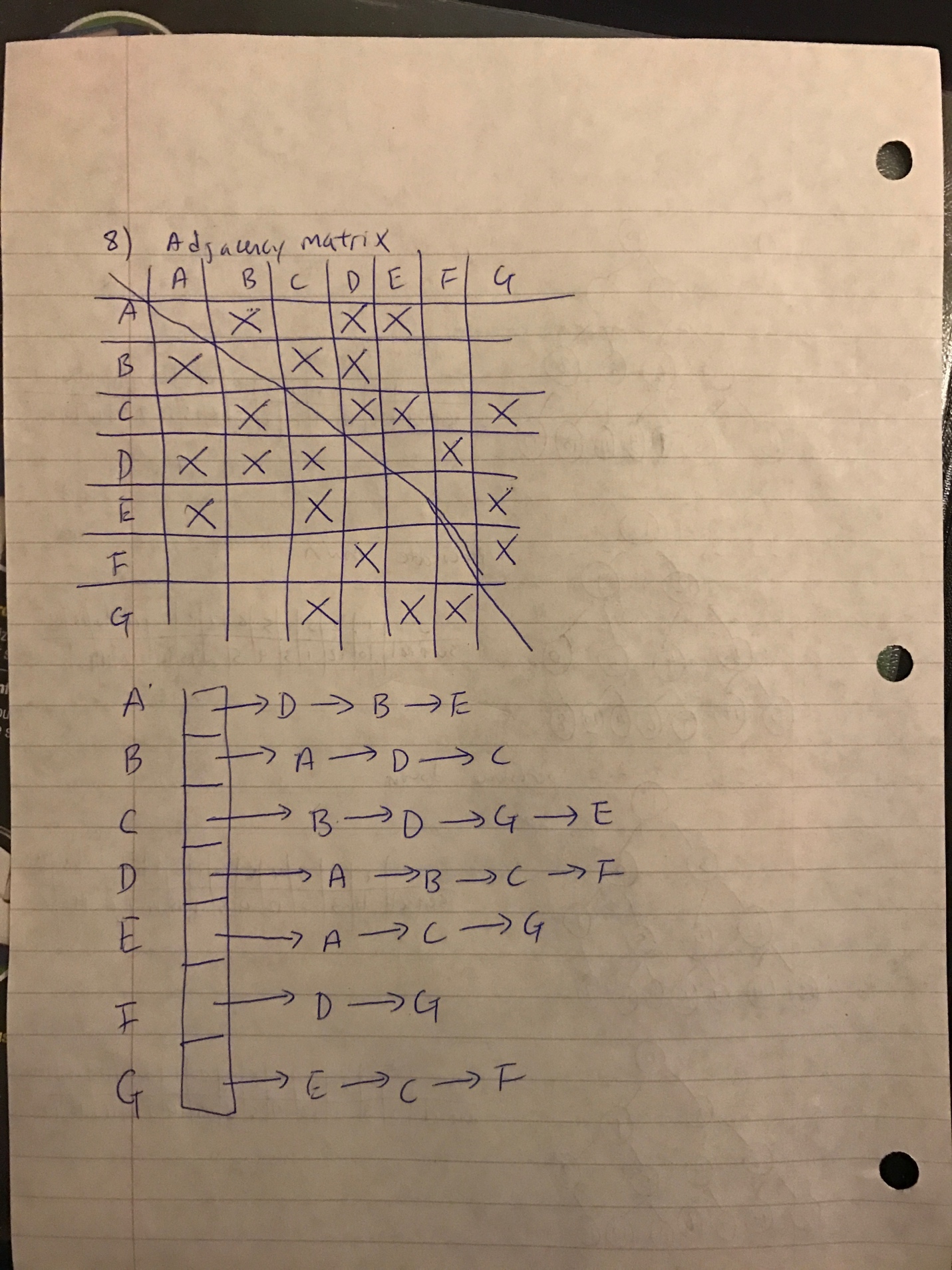
}



7)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| phase | distance/predecessor | | | | | | | visiting | queue |
|  | G | A | B | C | D | E | F |  |  |
| init | 0/- |  |  |  |  |  |  |  | G |
| 1 | 0/- | 1/G |  |  |  |  | 1/G | G | F A |
| 2 | 0/- | 1/G |  |  |  | 2/F | 1/G | F | A E |
| 3 | 0/- | 1/G |  |  |  | 2/F | 1/G | A | E |
| 4 | 0/- | 1/G |  |  | 3/E | 2/F | 1/G | E | D |
| 5 | 0/- | 1/G | 4/D | 4/D | 3/E | 2/F | 1/G | D | B C |
| 6 | 0/- | 1/G | 4/D | 4/D | 3/E | 2/F | 1/G | B | C |
| 7 | 0/- | 1/G | 4/D | 4/D | 3/E | 2/F | 1/G | C |  |

Shortest path from G to C is 4 (GFEDC)



9)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| phase | distance/predecessor | | | | | | | visiting | discovered |
|  | A | B | C | D | E | F | G |  |  |
| init | 0/-1 | Inf/ -1 | Inf/-1 | Inf/-1 | Inf/-1 | Inf/-1 | Inf/-1 |  | A |
| 1 | 0/-1 | 6/A | Inf/-1 | 3/A | 1/A | Inf/-1 | Inf/-1 | A | B D E |
| 2 | 0/-1 | 6/A | 19/E | 3/A | 1/A | Inf/-1 | 20/E | E | B D C G |
| 3 | 0/-1 | 3/D | 15/D | 3/A | 1/A | 6/D | 20/E | D | B C G F |
| 4 | 0/-1 | 3/D | 8/B | 3/A | 1/A | 6/D | 20/E | B | C G F |
| 5 | 0/-1 | 3/D | 8/B | 3/A | 1/A | 6/D | 20/E | F | C G |
| 6 | 0/-1 | 3/D | 8/B | 3/A | 1/A | 6/D | 13/C | C | G |
| 7 | 0/-1 | 3/D | 8/B | 3/A | 1/A | 6/D | 13/C | G |  |

Shortest path from A to G is 13 (ADBCG)

10)

a->b 3

a->c 2

b->c -3

In this example, since c has the smallest distance it is deleted from priority queue first by using the greedy algorithm. Therefore the final distance from a to c is 2. However, we can do better by going through b which will give a distance of 0.

11)

We can create an array which keeps track the number of different minimum paths from one point to another point. If the dijkstra finds a smaller path, then the count of that point resets to 0. If there is an equivalent minimum path, the count is increment by 1.