1. Algorithm design and complexity

1. Use binary search algorithm.
   (1) Given n building floors indexes from lowest to highest 1, 2, 3, … n
   (2) Set L to 1 and H to n
   (3) If L > H, the search terminates as unsuccessful
   (4) Set m (the position of the middle floor) to the floor of (L+R)/2
   (5) Throw a box on floor m, if the box is not broken, set L to m+1 and go to step (3) and step (4) until the box is broken on floor t, t is the lowest floor
   (6) Throw a box on floor m, if the box is broken, set H to m-1 and go to step (3) and step (4) until the box is not broken on floor t, t+1 is the lowest floor
2. Use binary search algorithm.
   (1) Given n building floors indexes from lowest to highest 1, 2, …, n. Binary divide n floors into 2^{k-1} intervals from the lowest floor to the highest floor. Each interval contains same number of floors.
   (2) Set L to 1 and H to n
   (3) Set m (the position of the middle floor) to the floor of (L+R)/2
   (4) Throw a box on floor m, if the box is not broken, set L to m+1 and go to step (3) until there is only one box left.
   (5) Throw a box on floor m, if the box is broken, set H to m-1 and go to step (3) until there is only one box left.
   (6) Throw the box from the lowest floor to higher floors one by one in the interval which contains m until the box breaks on floor t. Then t-1 is the lowest floor we want.

   For example: If n=16 and k=3, then divide the indexes of floors by 2^{3-1}=4 into {1,2,3,4}, {5,6,7,8}, {9,10,11,12}, {13,14,15,16} and begin throwing a box on floor 8. If the box breaks, then throw it on floor 4. If the box is not break, and now there is only one box left, then throw the box from floor 5 to floor 7 until it breaks.
3.
   (1) Given n building floors indexes from lowest to highest 1, 2, …, n. Equally divide n floors into sqrt(n) intervals from the lowest floor to the highest floor. Each interval contains same number of floors. Set the intervals as $A_1, A_2, \ldots, A_{sqrt(n)}$
   (2) Throw a box from the highest floor in A_1. If the box is not broken, throw it from the highest floor $A_2, A_3, \ldots A_{sqrt(n)}$ until the box gets broken at certain floor.
   (3) When the box is broken at floor k, throw the box from the lowest floor to higher floors in the interval which contains floor k until the box get broken from floor i. i is the index of the lowest floor from which dropping a box will break it.

   For example: If n=16 and k=3, then divide the indexes of floors by sqrt(16)=4 into {1,2,3,4}, {5,6,7,8}, {9,10,11,12}, {13,14,15,16} and begin throwing a box on floor 4. If the box is not broken, then throw it on floor 8. If the box is broken,

then throw the box from floor 5 to floor 7 until it breaks.

2. Greedy 1

Design the algorithm:

Sort leaks in the natural left-to-right order so that the i^{th} leak starting at s(i) and finishing at f(i). i=1, 2…, n.

Place strips from left to right. Assure that each leak part is covered and is not covered by more than one strips. Each strip starts where the leak starts. The start point of each strip mustn't be left to the start point of the corresponding leak to be covered.

The i^{th} strip starting from strip (i). Assume there are m strips.

```
m=1
strip (1) = s (1)
for (i=2; i<n; i++) {
   if(s(i)-strip(m)>9)
      {
         m=m+1;
         strip(m)=s(i)
      }
      else if(f(i)-strip(m)>9) {
         m=m+1;
         strip(m)=strip(m)+9;
      }
      }
```

Proof by induction:

(1) A is a compatible set of strips.

For the purpose of comparison, let O be an optimal set of strips. We will show that |A|=|O|, that is, A contains the same number of strips as O and hence is also an optimal solution.

Let i_1, …, i_k be the set of strips in A in the order they were added to A. Note that |A|=k. Similarly, let the set of strips in O be denoted by j_1, …, j_m. Our goal is to prove that k=m. Assume that the strips in O are also placed in the natural left-to-right order.

Our intuition for the greedy method came from wanting our strip to cover the least "good" part of the pipe (the leak part becomes "good" once it is covered). And indeed, our greedy rule guarantees that stripfinish(i_1) > stripfinish(j_1). This is the sense in which we want to show that our greedy rule "stays right"-that each of its strips places at least as right as the corresponding strip in the set O.

(2) For all indices r<=k, we have stripfinish (i_r)>=stripfinish (j_r).

Proof this by induction.

Induction hypothesis: stripfinish (i_{r-1}) >= stripfinish(j_{r-1})

Since O contains of compatible strips, stripfinish(i_{r-1}) <= leakstart(i_r). So, stripfinish (j_{r-1}) <= leakstart(i_r). Thus, the leak between stripfinish(j_{r-1}) and leakstart(i_r) is at least as much as leak between stripfinish(i_{r-1}) and leakstart(i_r). In the greedy

algorithm, the strip is as right as it can while covering all the leaks. So stripstart(i_r) is at least as right as stripstart(j_r). The length of each strip is the same so we have stripfinish (i_r)>=stripfinish (j_r).

(3) The greedy algorithm returns as an optimal set A.

Proof the statement by contradiction.

If A is not optimal, we must have k>m. Applying (2) with k=m, we have stripfinish(i_m)>=stripfinish(i_m). Since k>m, there is a leak in A after the m_{th} strip, which means there must be a leak after m_{th} strip in O too. Thus, O is not optimal. This is a contradiction. So, m must equal to k.

3. Greedy 2

Design the algorithm:

The ratio of mineral(i) = value(i)/weight(i).

Every time pick up the mineral with the highest ratio among all the minerals left until the total weight reaches the weight limit L.

Proof using an exchange argument:

(1) There is an optimal solution with no extra space in the bag.

If we want to have the maximum value, we can't let extra space in the bag.

(2) All placement with no inversions and no extra space have the same total value.

If two different solutions have neither inversions nor idle time, then they might not produce minerals exactly in the same order, but they can only differ in the order in which minerals with the same ratio. Consider such a ratio r. In both solutions, the jobs with ratio r are all placed consecutively (after all minerals with smaller ratios and before all minerals with larger ratios). Among the minerals with ratio r, the last one makes the max value, and this value does not depend on the order of the minerals.

(3) There is an optimal solution that has no inversions and no extra space.

(a) If O has an inversion, then there is a pair of minerals i and j such that j is placed immediately before i and has ratio(j)<ratio(i).

Indeed, consider an inversion in which a mineral a is placed sometime before a mineral b, and ratio(a)<ratio(b). If we advanced in the placement order of minerals from a to b one at a time, there has to come a point at which the total value we see increases for the first time. This corresponds to a pair of consecutive jobs that form an inversion.

(b) After swapping i and j we get a solution with one less inversion.

(c) The new swapped solution has a total value no smaller than that of O.

If mineral(i) and mineral(j) are both placed whole (we don't take part of them), the swapping them will not decrease the total value of the bag. If before swapping we take the whole mineral(j) and part of mineral(i), after swapping the weight of ratio(i) increases and weight of ratio(j) decreases. Thus, the total value of mineral(i) and mineral(j) in the bag increases.

(4) The solution A produced by the greedy algorithm has optimal maximum value.

Statement (3) proves that an optimal solution with no intervals and extra space

exists. Statement (4) proves that all solutions with no intervals and no extra space have the same maximum value, and so the solution obtained by the greedy algorithm is optimal.