# Burning Ropes (p. 121)

Constraints:

* Length of ropes: different
* Speed of burning of ropes: variable within a rope
* Where can we light the ropes: any of the ends
* Lighting a rope at both ends: possible
* Can we cut the ropes? Yes

Ideas:

* What do we know?
  + We can time 1 hour
  + We can time 2 hours
  + We can time 30 min, if we light one rope at both ends. When the flames meet, 30 min passed
* Procedure:
  + Light rope 1 from both ends, and rope 2 from one end at same time
  + Let’s call the point in time at which the flames of rope 1 meet X
  + At X we know that 30 min passed
  + Thus, we know that rope 2 has 30 min more to burn
  + At this moment, light rope 2 from the other end
  + When the flames of rope 2 meet, we know that exactly 15 min passed since X

# Nine Balls (p. 122)

Constraints.

* How many balls can we put on the scale? Any number

Idea:

* Put 3 balls on each side of the scale
* If the scale indicates equal weights, then the heavy ball must be among the three unused balls
* Else, if the scale indicates not equal weights, the heavy ball must be among the three balls of the heavier side
* So, in any case, we now have three balls, and we know that the heavy ball is among them
* How to find out which of the three balls is the heavy one:
  + Put two of the balls on the scale, one on each side
  + If the scale indicates equal weights, then the heavy ball is the unused one
  + Else, if the scale indicates unequal weights, then the heavy ball is the one of the heavier side on the scale

# The Heavy Pill (6.1)

Constraints:

* How many bottles can we put on the scale? Any number
* Can we take pills out of the bottle? Yes
* Can we put individual pills on the scale? Yes
* How many pills are in each bottle? Any number (not necessarily the same number in each bottle)
* What kind of scale? Electronic, put one object on it, and the scale indicates the weight

Idea:

* What do we know?
  + ~~11 pills from the light bottles have the equal weight of 10 pills of the heavy bottle~~
  + ~~22 pills from the light bottles have the equal weight of 20 pills of the heavy bottle~~
* Solution for if there are only 10 bottles
  + Put one bottle aside
  + Assign an index from 1 to 9 to each of the remaining 9 bottles
  + Of each bottle, put the number of pills equal to its index on the scale
  + If the total weight indicated by the scale is an integer, then the heavy bottle is the one set apart at the beginning
  + Else, if the decimal digit indicated by the scale is X, then the heavy bottle is the one with index X
* Sum of integers 1 to N: N(N+1)/2
  + Sum of integers 1 to 19: (19\*20)/2
* Solution for if there are 20 bottles
  + Assign an index from 0 to 19 to each bottle
  + Of each bottle put the number of pills equal to its index on the scale
  + Note the decimal digit X of the total weight
  + Now we know:
    - If X is 0, the heavy bottle is either #0 or #10
    - If X is 1, the heavy bottle is either #1 or #11
    - …
    - If X is 9, the heavy bottle is either #9 or #19
  + How to find out which of the two options is the heavy bottle:
    - Round the total weight down to the nearest integer (W)
    - If W equals (19\*20)/2, then the smaller of the two options is the heavy bottle
    - Else, if W equals (19\*20)/2+1, then the larger of the two options is the heavy bottle
  + For example, if the decimal digit X is 5, and the rounded down total weight W equals (19\*20)/2, then the heavy bottle is #5
  + Formalisation: #bottle = (total weight - (19\*20/2)) / 0.1

# Basketball (6.2)

Constraints:

* Is the probability p of making each shot the same? Yes
* For Game 2, win only if making exactly 2 shots or if making 3 shots also? Win if making 2 or 3 shots.

Example:

* p = 0.5
* Game 1:
  + Probability to make one shot: 0.5
  + Probability to win the game: 0.5
* Game 2:
  + Probability to make 1st and 2nd shot: 0.5 \* 0.5 \* 0.5 = 0.125
  + Probability to make 1st and 3rd shot: 0.5 \* 0.5 \* 0.5 = 0.125
  + Probability to make 2nd and 3rd shot: 0.5 \* 0.5 \* 0.5 = 0.125
  + Probability to make all 3 shots: 0.5 \* 0.5 \* 0.5 = 0.125
  + Probability to win the game: 0.125 + 0.125 + 0.125 + 0.125 = 0.5

Example:

* p = 0.8
* Game 1
  + Probability to make one shot: 0.8
  + Probability to win the game: 0.8
* Game 2
  + Probability to make 1st and 2nd shot: 0.8 \* 0.8 \* 0.2 = 0.128
  + Probability to make 1st and 3rd shot: 0.8 \* 0.2 \* 0.8 = 0.128
  + Probability to make 2nd and 3rd shot: 0.1 \* 0.8 \* 0.8 = 0.128
  + Probability to make all 3 shots: 0.8 \* 0.8 \* 0.8 = 0.512
  + Probability to win the game: 0.128 + 0.128 + 0.128 + 0.512 = 0.896

Example:

* p = 0.1
* Game 1
  + Probability to make one shot: 0.1
  + Probability to win the game: 0.1
* Game 2
  + Probability to make 1st and 2nd shot: 0.1 \* 0.1 \* 0.9 = 0.009
  + Probability to make 1st and 3rd shot: 0.1 \* 0.9 \* 0.1 = 0.009
  + Probability to make 2nd and 3rd shot: 0.9 \* 0.1 \* 0.1 = 0.009
  + Probability to make all 3 shots: 0.1 \* 0.1 \* 0.1 = 0.001
  + Probability to win the game: 0.009 + 0.009 + 0.009 + 0.001 = 0.028

General:

* p = n
* Probability to win Game 1: n
* Probability to win Game 2: 3 \* n^2 \* (1-n) + n^3 = 3n^2 - 3n^3 + n^3 = 3n^2 - 2n^3

Solution:

* For p = 0.5, the probabilities to win are the same in both games, namely 0.5
* For p > 0.5, the probability is higher to win in Game 2
* For p < 0.5, the probability is higher to win in Game 1
* For p = 1 and p = 0, the probabilities to win are the same in both games, namely 1 and 0

Calculating for which values Game 1 has higher probability to win:

* For which value is
  + p > 3p^2 - 2p^3
  + 1 > 3p - 2p^2
  + 0 > -2p^2 + 3p - 1
  + 0 < 2p^2 - 3p + 1
  + 0 < (2p - 1)(p - 1)
  + ...

# Dominos (6.3)

Constraints:

* Can we put the dominos horizontally and vertically? Yes

Idea:

* If two diagonal corners are cut out, there are either two black squares or two white squares
* Thus, on the resulting boards, there are either 32 black and 30 white squares, or 30 black and 32 white squares
* In whatever way you lay a domino on a chess board, it covers 1 black and 1 white square
* Thus, 31 dominos MUST cover 31 black and 31 white squares
* But the number of each color is 30 and 32, respectively
* Thus, it’s NOT POSSIBLE to cover the board with the dominos

# Ants on a Triangle (6.4)

Constraints:

* Ants start walking at the same time? Yes

Idea:

* Each ant can choose "right" or "left"
* There is a collision in any case, except if all the ants choose "right" or all the ants choose "left"
* We have to find the probability that all the ants choose "right" OR all the ants choose "left"
* What is the probability that all the ants choose "right"?
  + 1/2 \* 1/2 \* 1/2 = 1/8 = 0.125
  + The events are independent (if one ant chooses "right", it has no influence on the probability that another ant chooses "right")
* What is the probability that all the ants choose "left"?
  + 1/2 \* 1/2 \* 1/2 = 1/8 = 0.125
  + The events are independent.
* What is the probability that all the ants choose "right" OR all the ants choose "left"?
  + 0.125 + 0.125 = 0.25
  + The events are mutually exclusive (it cannot occur together that all ants choose "right" AND all ants choose "left")
* So, the probability of NO collision is 0.25
* Consequently, the probability of a collision is 1 - 0.25 = 0.75

Generalisation to n ants on n-vertex polygon:

* Probability that all ants choose "right":
  + (1/2)^n
* Probability that all ants choose "left":
  + (1/2)^n
* Probability that all ants choose "right" OR all ants choose "left":
  + 2\*(1/2)^n = (1/2)^(n-1)
* Probability that there is a collision: 1 - (1/2)^(n-1)

# Jugs of Water (6.5)

Constraints:

* Can we pour water in some accumulator container? No
* We have to end up with 4 quarts in one jug without using any additional jugs

Idea:

* What we can measure:
  + 5 quarts: by filling jug 1
  + 3 quarts: by filling jug 2
  + 2 quarts: by filling jug 1 and pouring into jug 2 until it’s full; the remaining water in jug 1 is 2 quarts
* Solution: in the following way we can end with 4 quarts in jug 1:
  + Fill jug 1 and pour into jug 2 until it’s full. The remaining water in jug 1 is 2 quarts. Empty jug 2 and pour the 2 quarts in jug 1 into jug 2. Now fill jug 1 fully, and pour into jug 2 until it’s full. This will remove 1 quart from jug 1, which results in 4 quarts remaining in jug 1.

Blue-Eyed Island (6.6)

Constraints:

* Can people see who is leaving on the same day? No, only the next day they see who has left the day before

Idea:

* Only one blue-eyed person: the blue-eyed person sees that nobody else has blue eyes, thus she knows that she is the one with blue eyes, and leaves on the first day
* Two blue-eyed persons: everybody can see at least one blue-eyed person, thus nobody knows whether himself has blue eyes or not
  + Thus, on the first day nobody leaves
  + Because nobody left on the first day, both blue-eyed persons know that there are more than one blue-eyed persons. Because both of them see only one blue-eyed person, the know that they themselves must also be a blue-eyed person, and they leave on the second day
* Three blue-eyed persons: the non-blue-eyed persons see three blue-eyed persons, and the blue-eyed persons see two blue-eyed persons. The blue eyed persons don't know if there are two or three blue-eyed persons in total (the latter case meaning that they are blue-eyed themselves)
  + On the first day nobody leaves
  + On the second day: the blue-eyed persons know, if there would be two blue-eyed people, they would have left on the second day
  + Thus, all the blue-eyed persons know that there must be three blue-eyed persons and that they are blue-eyed themselves, so they leave on the third day.
* It takes as many days for the blue-eyed people to leave as there are blue-eyed people
  + All the blue-eyed people leave on the same last day. All the days before, nobody leaves
* It's inductive knowledge: if there are n blue-eyed people, the question for all blue-eyed people is whether there are n-1 or n blue-eyed people (the latter case meaning they are blue-eyed themselves). However, the people know that if there are n-1 blue-eyed people, they would all leave on day n-1. So they wait until this day. If on day n-1 nobody left, they know that the variant of n-1 blue-eyed people is not true, thus the second variant must be true, namely that there are n blue-eyed people, which means that they are blue-eyed themselves. So, they leave on the next day (day n).

# The Apocalypse (6.7)

Idea:

* If the population is X:
  + The number of boys is: X/2 + X/4 + X/8 + X/16
  + The number of girls is: X/2 + X/4 + X/8 + X/16
* Thus, the gender ratio is 1:1.

# The Egg Drop Problem (6.8)

* N is the lowest floor where the egg breaks. We have to find N.
* M is the total number of floor. M is the input size.

Brute force:

* Start at floor 1, then floor 2, etc. Stop when the egg breaks. The current floor is N.
* Complexity: M = O(M) (linear)

Semi brute force:

1. Start at floor M/2
2. If the egg breaks, start at the lowest floor of the lower half, and go up floor by floor until the second egg breaks
3. Else, if the egg doesn’t break, drop it at floor M’/2 of the upper interval, and go to step 2
4. Complexity (worst case): M/2 = O(M) (linear)

Optimised:

1. Start at floor M/10
2. If the egg breaks, start at floor M/10-9 and go up floor by floor
3. Else, if the egg doesn’t break, go to floor 2\*M/10, and go to step 2

* Complexity (worst case): M/10 + 9 = O(M) (linear)

Optimal:

* 10+10+10+10+10+10+10+10+10+10 = 100
* 13+12+11+10+9+8+7+6+5+4+3+2+1 = 91
* 14+13+12+11+10+9+8+7+6+5+4+1 = 100 <<== best setting
* 15+14+13+12+11+10+9+8+7+1 = 100
* Start from floor 14, then go up 13 floors, then go up 12 floors, etc.
* Complexity (worst case): 14

# 100 Lockers (6.9)

Constraints:

* Toggling every nth locker, doesn’t include locker #1? Yes, toggling every nth locker starts at locker #n
* n runs from 1 to 100, and we toggle every nth locker
  + Opening all lockers on the first pass is equivalent to toggling every locker (since all lockers are closed before)
  + Closing every second locker on the second pass is equivalent to toggling every second locker (since all lockers are open before)

Example with 10 lockers:

* Locker numbers to toggle
  + n=1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
  + n=2: 2, 4, 6, 8, 10
  + n=3: 3, 6, 9
  + n=4: 4, 8
  + n=5: 5, 10
  + n=6: 6
  + n=7: 7
  + n=8: 8
  + n=9: 9
  + n=10: 10

Idea:

* We have to find out for each locker, how many times it is toggled
  + Locker X is toggled m times, where m is the number of divisors of X
* If the number of toggles is even, then the locker is closed at the end
* Else, if the number of toggles is odd, then the locker is open at the end
* Thus, we have to find out how many of the 100 lockers have an odd number of toggles
  + **I.e. the problem is: how many of the integers between 1 and 100 have an odd number of divisors?**

Implementation:

public class Lockers {

private static final int N = 100;

public static void main(String[] args) {

int nOdd = 0;

for (int i = 1; i <= 100; i++) {

if (divisors(i) % 2 == 1) nOdd++;

}

System.out.println(nOdd);

}

private int divisors(int n) {

int numDivisors= 0;

for (int i = 1; i <= n; i++) {

if (n % i == 0) numDivisors++;

}

return numDivisors;

}

}

Answer:

* The implementation results in 10 open lockers.

Complexity:

* N^2 = O(N^2)

Solution without implementation:

* All numbers, except perfect squares, have an even number of divisors (a perfect square is 1\*1=1, 2\*2=4, 3\*3=9, 4\*4=16, etc.)
  + For number N
    - If 1 is a divisor, then N/1 is a divisor
    - If 2 is a divisor, then N/2 is a divisor
    - …
  + For perfect squares, and only for perfect squares, this results at some point in:
    - If x is a divisor, then N/x = x is a divisor
    - That is, the pair with x contributes only one divisor, whereas all the other pairs contribute two divisors
* Thus, the problem is reduced to, how many perfect squares are there between 1 and 100?
  + Answer: 10 (because 1\*1=1, 2\*2=4, 3\*3=9, …, 10\*10=100)

# Poison (6.10)

Constraints:

* Can we use multiple test strips at the same time? Yes
* So, we can run 10 tests today, then we have to wait 7 days before we can run another test
* We can put drops of the same bottle on multiple test strips

Brute force:

1. One drop of the next 10 untested bottles on test strips 1 to 10
2. Wait 7 days
3. If one strip is positive, we have the solution
4. Else, if all strips are negative, go to step 1

* Complexity (worst case): 100 \* 7 days = 700 days

With the brute-force solution, we reduce the number of bottles among which the poisonous must be only by 10 each on each step (this is 1/100 of the total number of bottles). We should make this ratio bigger.

Optimised:

* Make 10 groups of 100 bottles each, and put on each test strip a drop of each bottle of a specific group
* **Wait 7 days**
* One test strip must be positive, so we know that the poisonous bottle is among these 100 bottles
* We have 9 test strips left and 100 bottles to test
* Put 11 drops on each strip (one bottle remains untested)
* **Wait 7 days**
* If no strip is positive, then the poisonous bottles must be among the untested (in this case just 1). The max number of untested bottles is #usedStrips - 1, thus we could test which bottle is the poisonous one in a single pass with the strips we have
* Else, if one strip is positive, we know that the poisonous bottle must be among these 11 bottles
* We have 8 strips left to test 11 bottles
* Put one drop on each strip (leave 3 bottles untested)
* **Wait 7 days**
* If no strip is positive, the poisonous bottle must be among the 3 untested bottles, we can find out which one in another **7 days**
* Else, if one strip is positive, we know which bottle is the poisonous one
* Complexity (worst case): 4\*7 days = 28 days

Optimal solution:

* Enumerate the 1000 bottles as binary numbers with 10 digits
  + I.e. 0000000000 (0) to 1111100111 (999)
* Enumerate the test strips from 0 to 9
* Of each bottle, put a drop on those test strips with the index where the binary number has a 1
  + E.g. of bottle 0100101101, put a drop on the test strips 0, 2, 3, 5, 8
  + E.g. of bottle 0000000000, don't put a drop on any test strip
* Wait 7 days
* The indices of the test strips that are positive after 7 days are the indices of the bottle number that contains the poison
* Complexity (any case): 7 days

Why does it work? The content of each bottle is distributed in a unique constellation on the 10 strips. If for example test strips 0, 2, 4 indicate positive, then it can be only one bottle which has been put on exactly the test strips 0, 2, and 4 (namely 0000010101).