1. 당연히 적어도 하나의 불량동전은 있어야한다.

- 불량동전이 100개인지 정상동전이 100개인지 판별할 수 없기 때문이다.

2. 분석하는데 oblivious decision tree / adaptive decision tree를 쓰는데 뭔소린지 모르겠다

3. 문제의 종류

Lower Bounds for Coin-Weighing Problems(ERIC PURDY)의 논문 참고 - 시온이가 올림

Problem	Info	Adaptive		Oblivious	
		Lower	Upper	Lower	Upper
Counting	$\log_3 n$	$\Omega(\log n)$	$O((\log n)^2)$ [F, Alg 3]	$\Omega(\sqrt{n})$ [Thm 1.1]	n
Parity	1	$\Omega(\log n)$ [Cor 1.4]	$O(\log n)$ [F, Alg 2]	$\Omega(\sqrt{n})$ [Thm 1.1]	n
Location	$\log_3 n$	$\Theta(\log n)$ [A]		$\Theta(\log n)$ [B]	
Diagnosis	$\log_3 \binom{n}{c}$ $(\log_3 2)n$	$\Theta\left(\log_3 \binom{n}{c}\right)$ [C]		$\Theta(n)$	
All Equal	1	$\Theta\left(\frac{\log n}{\log\log n}\right)$ [D]		$\Omega\left(\frac{\log n}{\log\log n}\right)$ [E]	?
Finding (Sec. 7.1)	$\geq \log_3 n$	$\Omega(\log n)$	$O(\log n)$ [F, Alg 1]	$\Omega(\log n)$	?
Splitting (Sec. 7.2)	$\geq \log_3 n$	$\Omega(\log n)$	$O(\log n)$ [F, Alg 2]	$\Omega(\log n)$	?

Bad Coin Counting Problem	c개의 불량 동전의 개수를 센다.
Bad Coin Parity problem	c개의 불량 동전이 홀수인지 짝수인지 판별
Bad Coin Location problem	무게를 모르는 1개의 불량 동전을 찾는다.
Bad Coin Finding Problem	무게를 아는 1개의 불량 동전을 찾는다.
Bad Coin Splitting Problem	정상 동전과 불량 동전의 개수가 같은 두 집합으로 나눈다.
All Equal	모든 동전의 무게가 같은지 판별한다.
Bad Coin Diagnosis problem	c개의 불량 동전을 찾는다. (c는 known or unknown)

우리가 푸는 문제는 Bad Coin Diagnosis problem이다. 제일 어렵다.

1994~1997년에 나온 논문에 의하면(1997년보다 더 최근 논문이 있는지 모르겠다.)

$$a\log_3\binom{n}{c}, a \leq 2\log_2 3$$

의 비교횟수를 가진다.

이와 관련된 논문은

- ✓ A new competitive algorithm for the counterfeit coin problem. (학교 로그인해도 유료다)
- ✓ A ($\log_2 3 + ½$)-competitive algorithm for the counterfeit coin problem
- ✓ A 3/2log3-competitive algorithm for the counterfeit coin problem

4. 만약 평균 $a\log_3\binom{n}{c}$이라면 좋은 비교 횟수일까?

a = 1	n = 100	n = 1000	n = 10000
10%	27	293	2955
20%	43	452	4550
30%	53	552	5556
40%	58	609	6121
50%	60	627	6304

a = 3/2log$_2$3	n = 100	n = 1000	n = 10000
10%	65	696	7025
20%	**103**	**1075**	**10818**
30%	**126**	**1314**	**13209**
40%	**140**	**1448**	**14553**
50%	**144**	**1492**	**14989**

왼쪽은 불량 동전(c)의 비율 / n은 전체 동전의 수

계수가 크면 불량동전의 개수가 약 20%보다 많으면 n번을 넘는다.

불량동전의 개수가 많아지면 n번을 탐색하는게 더 나은 것이다..아마도

5. 우리는 과제의 특성을 잘 파악하는게 중요하다.

첫번째 n이 정말 작다.

$\log_2 100 = 7$이다. 로그를 씌워도 그렇게까지 많이 줄으는건 아니다. ($\log_2 1000 = 10$인데..)

알고리즘이 로그 복잡도를 가져도 상수가 커버리면 의미가 적어진다는 것

심지어 $(\log_2 100)^2 = n/2$ 와 같다.

두번째 각각 다른 확률에 대해서 10번 시행할 것이다.

(일단 운을 심하게 탈 것은 뻔하고 뻔하고 뻔하다. 동전 1000개로 10000번정도 시행해야..)

(내 생각엔) 확률은 작은 것부터 큰 것까지 시행할 가능성이 높다.

예를 들면 10% 20% … 80% 90% 이렇게

세번째 우리가 개선해야 하는건 최악의 경우가 아니다.

우리가 개선해야하는 것은 10번의 시행의 합이다.

아무리 최악을 개선해서 모든 경우에 70번 안에 해결한다고 해보자.

A1. 모든 경우 70번, 불량 동전의 개수가 작거나 많아도 같다.

A2. 최악일 때 90번, 하지만 불량 동전의 개수가 작거나 많으면 더 짧다.

	c = 1	c = 10	c = 20	c = 30	c = 40	c = 50	c = 60	c = 70	c = 80	c = 90	sum
A1	70	70	70	70	70	70	70	70	70	70	700
A2	**30**	**40**	70	80	85	90	85	80	70	**40**	670

근데 대부분의 알고리즘은 불량 동전의 개수가 적으면 시행 횟수는 줄어들것이다… 위는 그냥 예시

내가 말하고자 하는건

> 1. 최악은 나쁘지만 최악이 걸릴 확률이 상당히 낮은 알고리즘(예를 들면 010101…) // 1. 10번 시행
>
> 2. 불량 동전의 개수가 많으면 횟수가 많지만 적으면 훨씬 줄어드는 알고리즘 // 2. 각각 다른 확률

이 유리할지도 모른다.

6. 앞으로 해볼 것

1. A 3/2log3-competitive algorithm for the counterfeit coin problem 구현(내가 해볼...게)

Algorithm A

$X := \{c\}$ *where* c *is any coin in* C, $U := C - X$;
repeat
  Found $:= 0$;
  if $(|X| = 1)$ **then**
    ToSeek := *unknown*;
  if $(|X| > 1)$ **then**
    binary search X; /* *One weighing could be saved if* $|X| \leqslant 3$ *or if two haves of* X *has been compared in the previous phase* */
    Found := *Found* + 1;
    $X := 2^{\lfloor \log |X| \rfloor}$ *coins from* X *containing the unique coin*;
  repeat
    $Y := \min\{|U|, |X|\}$ *coins from* U;
    compare X and Y;
    if Y *is pure*, **then**
      $X := X \cup Y$, $U := U - Y$;
    else
      binary search Y;
      Found := *Found* + 1;
      if *found* $= 1$ **then**
        ToSeek := *the type of the coin the binary search looks for*;
      if Y *is unique*, **then**
        $X := X \cup Y$, $U := U - Y$;
      else
        Let A_1 *and* A_2 *be the two halves involved in the last equal weighing and* A_1 *is identified in the binary search.*
        if *Found* $= 2$ **then**
          $X := A_2$, $U := U - A_1$;
        else
          binary search A_2;
          Found := *Found* + 1;
          $U := U - (A_1 \cup A_2)$;
          if $Y - (A_1 \cup A_2) \neq \emptyset$ *and there is only one equal weighing in the path,* **then** *compare* $Y - (A_1 \cup A_2)$ *against a unique set of same size*;
          if $Y - (A_1 \cup A_2)$ *is empty or pure*, **then**
            $X := X \cup Y$, $U := U - (Y - (A_1 \cup A_2))$;
          else
            if $Y - (A_1 \cup A_2)$ *is unique*, **then**
              $X := Y - (A_1 \cup A_2)$, $U := U - (Y - (A_1 \cup A_2))$;
            else
              $X := \{c\}$ *where* c *is any coin in* U, $U := U - \{c\}$;
  until *found* $= 2$ *or* $U = \emptyset$;
  /* *Now* X *either contains a single coin or is a unique set or contains exactly two coins of the ToSeek type.* */
  if X *contains two coins of the ToSeek type*, **then**
    repeat
      $Y := \min\{|U|, |X|\}$ *coins from* U;
      compare X and Y;
      if Y *contains at least two coins of the ToSeek type*, **then**
        $X := \{c\}$ *where* c *is any coin in* U, $U := U - \{c\}$;
      else
        split Y *into two halves and compare them*;
        if Y *is unique*, **then**
          $X := Y$, $U := U - Y$;
        else
          $X := X \cup Y$, $U := U - Y$;
    until X *is either unique or contains a single coin or* $U = \emptyset$;
until $U = \emptyset$;

2. 내가 말한 과제의 특성을 잘 파악해서 우리만의 알고리즘을 생각해볼 것

기존에 생각한 알고리즘도 더 개선할 수 있을지 모른다.

특히 현재까지 나온 정상 동전의 수와 불량 동전의 수를 비교해서

더 큰 값에 따라 전략을 다르게하면 효과적일 것 같다.

그리고 재호가 Coin 객체 상당히 잘 구현해놔서 이거 써서 하면 되겠다. 고수...

3. 이렇게 나온 코드들을 재호가 한 것처럼 서로 비교한다.

1%		
merge	15 29 15 25 16 30 34 22 14 22	평균 22.2, 최댓값 34, 최솟값 14
step	100 100 99 96 98 100 100 100 100 100	평균 99.3, 최댓값 100, 최솟값 96
25%		
merge	68 69 65 58 65 67 73 72 68 65	평균 67.0, 최댓값 73, 최솟값 58
step	85 81 76 81 84 76 83 82 78 82	평균 80.8, 최댓값 85, 최솟값 76
50%		
merge	70 75 74 70 65 71 69 70 71 69	평균 70.4, 최댓값 75, 최솟값 65
step	65 64 66 69 63 69 63 68 68 68	평균 66.3, 최댓값 69, 최솟값 63