Probabilistic Algorithm

Algorithm : Design & Analysis

[22]

In the Last Class…

- Polynomial Reduction
- Conquer the Complexity by Approximation
- Approximation Algorithm for Bin Packing
- Evaluate an Approximation Algorithm
- Online algorithm

Probabilistic Algorithm

- Adding Randomness into Algorithm
- Primality Problem
- Monte Carlo Algorithm for Primality Testing
- *p*-Correctness and Bias
- Decreasing Error Probability by Repeating

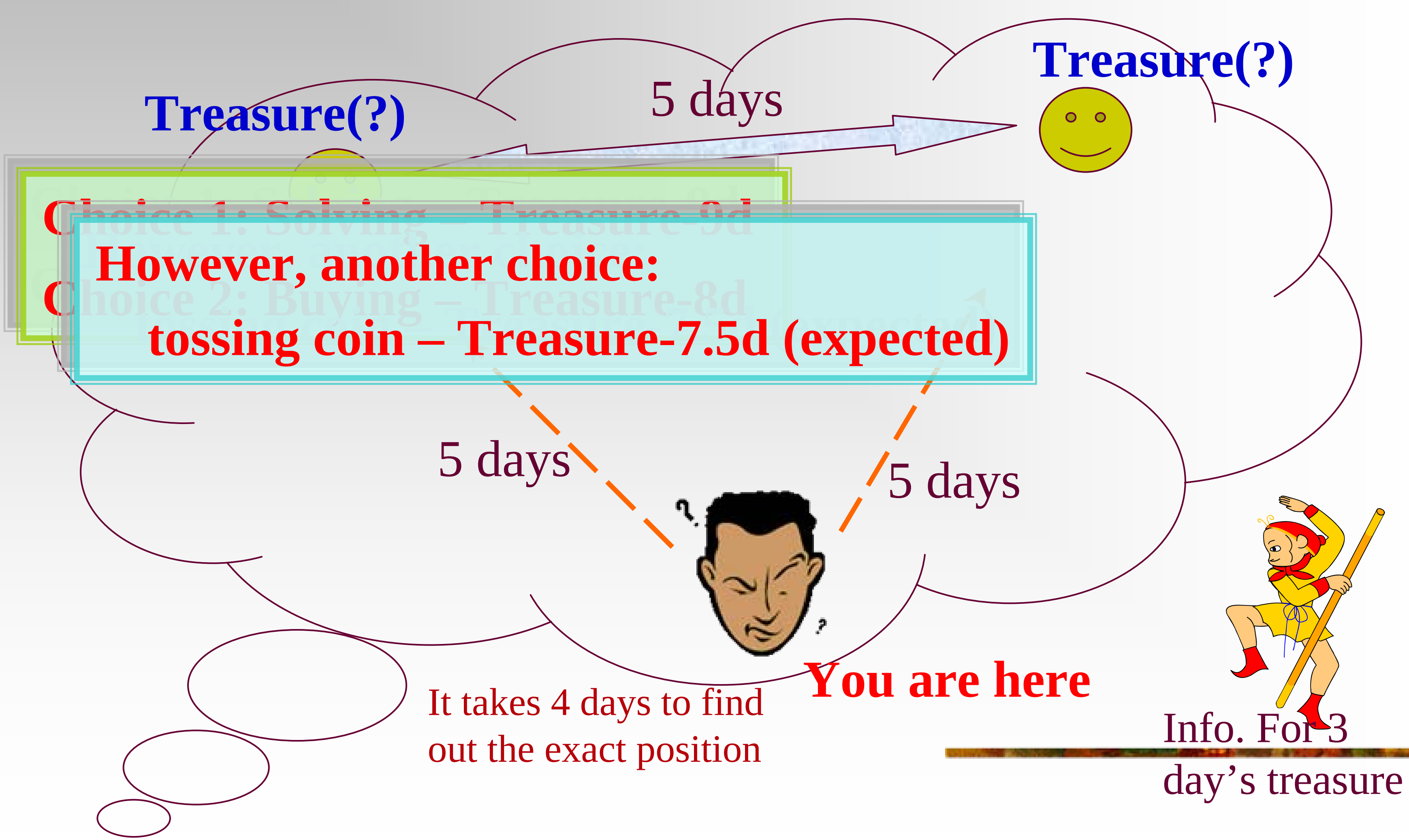
Randomization as a Choice

Choice 1: Solving – Treasure-9d

Choice 2: Buying – Treasure-8d

However, another choice:

tossing coin – Treasure-7.5d (expected)



A More Serious Application

Computer A

Computer B

We want to verify the consistency between the two databases of size n (n=10^{16}, say) located on A and B.

For a deterministic answer, we have to transfer a message of at least n bits, and (!) without an error on the way. It doesn't look a pleasant task.

Randomness Can Help

Let the database in A is $x=x_1x_2\ldots x_n$, and that in B is $y=y_1y_2,\ldots,y_{n=}$
Number (x) is the value of x as a binary number.

Randomized Protocol for Equality (A,B)

1. A chooses uniformly a prime p from $[2, n^2]$ at random.
2. A computes s=number (x) mod p, and sends s, p to B (note: s and p can each be represented in $\lceil \log_2 n^2 \rceil$ bits.)
3. B reads s and p, and computes q=number (y) mod p
4. B sends message: “yes” if $q=s$; and “no” otherwise.

No Guarantee, but Pretty Good

- The protocol may err: x=01111(15), y=10110(22), if p=7, then s=1, and q=1. So, 7 is a “bad” prime in $[2, n^2]$.
- In fact, if $x=y$, the protocol always gives the correct answer, and if $x \neq y$, wrong result is give if a “bad” prime selected.
- So, the probability that the protocol errs is the ratio of “bad” prime in the set of all prime in $[2, n^2]$.
- Fortunately, the probability is low: $\ln n^2/n$, that is, no greater than 0.36892×10^{-14} for $n=10^{16}$
- Another good news: the length of the message transferred is no longer than $4\lceil\log_2 n^2\rceil$ (that is, 256 for $n=10^{16}$)

Two Categories of Randomness

- Introducing randomness to the process of solving a problem, but always give a correct result.
  - Quicksort using randomly selected pivot.
- Introducing randomness to the results. (which year was the PRC founded? as an example)
  - 1949, 1949, 1948, 1005, 1949, 654B.C, 1949, 1949, 2005, 1949
  - 1949, sorry, 1949, 1949, sorry, 1949, sorry, 1949, 1949, 1949
  - Between 1900 to 2000; between 1945 to 1950; between 1503 to 1845; between…

(the problem is: what should I believe?

Primality Testing

- Problem: Is a given (odd) positive integer a prime?
- The algorithm you are familiar with:

Boolean TestPrimality(n)
  int i;
  for (i=2; $i \leq \lfloor \sqrt{n} \rfloor$; i++)
    if (n **mod** i = 0)
      return false;
  return true

The Algorithm is Exponential

- For number-theoretic algorithms, we usually take the input size to be the number of digits in n.
- Using decimal system, the number of digits in n is $m = \lceil \log_{10}(n+1) \rceil$
- The function *TestPrimality* executes the **for** loop $\lfloor \sqrt{n} \rfloor$ times in the worst case, which is about $10^{m/2}$.
- If n has more than 40 digits, it would take millions of years to say “yes” .

Randomness Introduced Simply

```
Boolean TestPrimalityRandom(n)
    int j;
    Random({2,...,⌊√n⌋},j)
    if (n mod j = 0)
        return false;
    else
        return true;
```

However, the algorithm is *unacceptable* in practice.

Even though it is really "not" when returned value is "false", it has no any positive "*fixed confidence level*" if the returned value is "true".

Monte Carlo Algorithms

- For a (fixed) real number p, $0<p<1$, a p-correct Monte Carlo algorithm is a probabilistic algorithm that returns the correct answer with probability not less than p no matter what input is considered.
- Note: the algorithm *TestPrimalityRandom* is 100% correct if it returns "false", (on the other hand, if $n=p_1p_2$ with primes $p_1<p_2$, the algorithm has only $1/p_1$ probability to give a correct "false"), but is correct with ANY small probability if it returns "true".
- ***Bad news***: there is usually no efficient method available to test whether an answer returned by a Monte Carlo algorithm for a given input is correct.

Help from as Early as 1640

- Fermat's little theorem:
  - If n is a prime, then for all positive $a<n$,

$$a^{n-1} \equiv 1 \pmod{n}$$

(for example: $1^6=0\times 7+1$; $2^6=9\times 7+1$; …, $5^6=2232\times 7+1$; $6^6=6665\times 7+1$)

- [Contraposition of Fermat's little theorem]
  - If n and a are integers such that $1\le a\le n$, and the modular equation above does not hold for them, then n is NOT a prime.

(a **deterministic** result for negating an natural number being prime)

What about Positive Result

- The following statement **DOESN'T** hold:
  - If $a^{n-1} \equiv 1 \pmod{n}$ for some natural number n and $a(a<n)$, then n is prime (**false!**).
  - If fact, there **do** exist **composite** numbers n, for which $a^{n-1} \equiv 1 \pmod{n}$ for most $a<n$. (even for all $a<n$)
- However
  - Good news: **most** composite numbers n have **many** integers a in $\{2,\ldots,n-1\}$ such that $a^{n-1} \neq 1 \pmod{n}$
- It seems that a test for primality for THOSE composite numbers that has a high probability of being correct is to test whether $a^{n-1} \neq 1 \pmod{n}$ for a random choice of a.

Monte Carlo Algorithm for Primality

```
Boolean FermatTestPrimality(n)
    int a;
    Random ( {2,3,…, n-2}, a);
    if expomod (a, n-1, n) =1
        return true;
    return false;
```

```
int expomod (a,n,z) // computes a^n mod z
    int i, r, x;
    i=n; r=1; x=a mod z;
    while (i>0)
        if i is odd
            r=rx mod z;
        x=x^2 mod z;
        i=i div 2;
    return r;
```

The loop is executed about log n times

Note the facts:

xy **mod** z =

$(x$ **mod** $z)$ $(y$ **mod** $z)$ **mod** z;

$(x$ **mod** $z)^y$ **mod** z = x^y **mod** z;

False Witness of Primality

- When FermatTestPrimality returns “false”, *n* is NOT a prime with certainty, but what about a returned value of “true”?
- Examples of fails: $(n-1)^{n-1} \bmod n = 1$ for all odd $n \geq 3$; and a nontrivial example: $4^{14} \bmod 15 = 1$.(4 is called a *false witness of primality*)

What's the Risk

- Good news – false witness are rather few: for all odd composite less than 1000, more than half have only 2 false witness, and less than 16% have more than 15. The average error probability on odd composite smaller than 1000 is less than 3.3%.
- Bad news – some odd composites admit a significant proportion of false witness: the worst case among odd composites smaller than 1000 is 561 with 318 false witness, and 651693055693681 is a composite but return "true" with probability of 99.9965%.
- In fact, FermatTestPrimality is NOT *p*-correct for any *p*>0.

A Special Set for Testing

- Let n be an odd integer greater than 4, and $n-1=2^s t$, where t is odd. Define set B(n) as follows:
  - $a \in B(n)$ if and only if $2 \le a \le n-2$, and
  - $a^t \bmod n = 1$, or
  - there is an integer i, $0 \le i < s$, such that $a^\wedge(2^i t) \bmod n = n-1$

Which implies a is Fermat false witness as well.

- Example: $158 \in B(289)$
  - $289-1 = 288 = 2^5 \times 9$, so s=5, t=9
  - $x=158^9 \bmod 289 = 131 \neq 1$, so, condition 1 does not hold; but
  - We successively square x (mod n) up to 4 times (s-1=4)

    $131^2 \bmod 289=110$; $110^2 \bmod 289=251$; $251^2 \bmod 289=288$
  - So, $158 \in B(289)$

Extension of Fermat's Little Theorem

- If n is prime, then $a \in B(n)$ for all $2 \leq a \leq n-2$.
- If $n>4$ is an odd composite number and there is some a in [2,n-2] satisfying $a \in B(n)$, a is called a strong false witness of primality for n.
- Good news:
  - Strong false witnesses are much rarer than Fermat false witness. For example, 4 is NOT strong false witness for 15.
  - For all odd composites smaller than 1000, the average probability of randomly selecting a strong false witness is less than 1%, more than 72% of them do not admit strong false witness.
  - At least one of 2,3,5,7 or 61 is NOT a strong false witness for every odd composite integer between 5 and 10^{13} .
  - Most importantly, there is a guarantee that the proportion of strong false witness is small for every odd composite.

Improved Algorithm – Miller-Rabin

```
Boolean MillRab(n)
//only for n>4 is odd
    int a;
    Random ({2,…n-2}, a);
    return Btest(a,n)
```

```
Boolean Btest(a,n)
    int i,s,t,x;
    s=0; t=n-1;
    while ((t mod 2)≠1)
        s=s+1; t=t/2;
    x=expomod(a,t,n);
    if (x= =1 or x= =n-1) return true;
    for (i=1; i<s; i++)
        x=x^2 mod n;
        if (x= =n-1) return true;
    return false;
```

It can be proved that if n is composite, then $|B(n)| \leq (n-9)/4$, which means that MillRab is **¾-correct** Monte-Carlo algorithm

Repeat for Better

- The answer “false” is guaranteed to be correct.
- If n>4 is prime, MillRab always return the correct answer, on the other hand, if n>4 is an odd composite, MillRab has probability at most ¼ of hitting a strong false witness and erroneously returning “true”.
- So, the following algorithm is $(1-4^{-k})$-correct:

```
Boolean RepeatMillRab(n,k)//only for n>4 is odd
  int i;
  for (i=1; i≤k; i++)
    if MillRab(n)=false return false;
  return true;
```

The Complexity

- How much time does it take to decide on the primality of n with an error probability bounded by ε?
  - Repeat the Miller-Rabin algorithm test k times such that $4^{-k} \leq \varepsilon$, that is, $2^{2k} \geq 1/\varepsilon$, so, $k = \lceil 0.5 \lg(1/\varepsilon) \rceil$
  - Each call of MillRab involves one modular exponentiation, with t as exponent and s-1 modular squaring, which is dominated by modular multiplication, in $O(\log n)$ times, each taking a time in $O(\log^2 n)$, resulting $O(\log^3 n)$
  - So, the complexity is $O(\log^3 n \lg(1/\varepsilon))$.
  - This is reasonable in practice for thousand-digit numbers and error probability less than 10^{-100}.

Probability Algorithm in General

- Motivation: wrong guess vs. high cost
- Characteristics: behave differently on same input
- Understanding:
  - What’s the real meaning of the sentence: “ it is a prime with the probability of 99.99999%” ?
  - What’s the meaning of the sentence: “I believe that it IS a prime.”

Home Assignment

- Do anything you think helpful for your exam.

OK! That's all for the semester.
Thank you all!
Good luck, for everything in general,
and for the exam in specific!