NOI 2009—Solutions to Contest Tasks

Martin Henz

March 14, 2009

The Scientific Committee of NOI 2009

- Ben Leong
- Chang Ee-Chien
- Colin Tan
- Frank Stephan
- Martin Henz
- Sung Wing Kin

1. XMAS
2. INVEST
3. LAZYCAT
4. CIPHER

1. XMAS
   - Problem
   - Example
   - Background
   - Algorithm
   - Program in Pascal
2. INVEST
3. LAZYCAT
4. CIPHER

Problem

At a “Secret Santa” Christmas party, each guest brings a gift and places it under the Christmas tree. Each guest gets exactly one gift from under the tree.

Problem

At a "Secret Santa" Christmas party, each guest brings a gift and places it under the Christmas tree. Each guest gets exactly one gift from under the tree.

You are given a list that says whose gift each person receives, and need to compile a list of guests that pick each person's gift.

Example

- Alice brings an **apple**, Bob brings a **balloon**, and Carol brings a **cake**.
- Given: Alice gets the **balloon**, Bob gets the **cake**, and Carol gets the **apple**.
- Required list: Apple goes to **Carol**, balloon goes to **Alice**, cake goes to **Bob**.

Background

Picking a gift

corresponds to a *permutation* of the list of guests.

Background

Picking a gift

corresponds to a *permutation* of the list of guests.

Task

compute the *inverse* of the permutation.

Algorithm

Idea

For each gift, remember its receiver. Then output the receivers in a loop.

Program in Pascal

```
program xmas;
var
   Receiver:array[1..100000] of integer;
   guests,gifts,guest,gift:integer;
begin
   readln(guests); gifts := guests;
   for guest := 1 to guests do
   begin
     readln(gift);
     Receiver[gift] := guest;
   end;
   for gift := 1 to gifts do
      writeln(Receiver[gift]);
end.
```

Complexity

- There is a fixed amount of work done in each loop body.

Complexity

- There is a fixed amount of work done in each loop body.
- Each loop runs through all guests (gifts).

Complexity

- There is a fixed amount of work done in each loop body.
- Each loop runs through all guests (gifts).
- Overall: the runtime grows proportional to the number of guests.

1. XMAS
2. INVEST
   - Problem
   - Example
   - Algorithm
3. LAZYCAT
4. CIPHER

Problem

- Investment options: oil, shares, steel, silver and gold.
- Investor has to commit to an investment plan over m months.
- Investor has to commit to a sum of $2520 which is invested each month.
- Each month, the investor has to buy one product for $2520.
- Investor has to buy the same product in the first 15 months of the plan (or in all months if it is shorter than 15 months).
- Two consecutive changes of the product bought must be separated by at least 14 months, in which the product remains unchanged.

Example

For a 36 months plan it is possible to buy gold the first 18 months, then silver the next 15 months and then gold again the remaining three months.

Example Input

```
6
7 2 2 6 6
3 1 7 3 3
8 4 3 2 7
5 5 7 6 6
6 4 7 2 7
1 6 6 2 2
3 4 7 7 8
```

First six rows contain buy price in given month; last row contains sell price after last month.

Desired Output

Best returns: The amount of money that can be earned from selling all investments at the end of the period, after investing each month in the best possible way.

A Naive Approach

- Consider all possible combinations of investments. Example: `G-G-O-O-O-S`
- For each combination, check that it meets the rules.
- If it meets the rules, compute the returns.
- Output the highest return.

Problem

Complexity!

We have to go through 5^{months} combinations!

First Idea

Observation

We are allowed to change the investment product only after at least 15 months

First Idea

Observation

We are allowed to change the investment product only after at least 15 months

Reduce combinations

Only consider months in which a change of product happens.

First Idea

Observation

We are allowed to change the investment product only after at least 15 months

Reduce combinations

Only consider months in which a change of product happens.

Complexity

reduces to $5^{\text{months}/15}$ combinations!

Second Idea

Memoization

Once we have computed the best return when starting with a certain product on a given month, we don't need to compute it any more; we simply remember the computed value in a table.

Second Idea

Memoization

Once we have computed the best return when starting with a certain product on a given month, we don't need to compute it any more; we simply remember the computed value in a table.

Complexity

The overall runtime reduces to

$$\mathrm{numberOfProducts} \cdot \mathrm{numberOfMonths}$$

1. XMAS
2. INVEST
3. LAZYCAT
   - Problem
   - Algorithm
4. CIPHER

Problem

The map of a house is given by a grid like this one:

B		F	
X	X	F	
F	X	X	F
S			

A "lazy" cat at a "S"tarting point wants to pick up all "F"ood items and then go to "B"ed.

Rules

B		F	
X	X	F	
F	X	X	F
S			

- Only step left, right, up or down
- Do not step on walls marked with "X"

Traveling Salesman Problem (TSP)

The problem

Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once.

Traveling Salesman Problem (TSP)

The problem

Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once.

Example

Traveling Salesman Problem (TSP)

The problem

Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once.

Example

- Optimal tour visiting Germany's 15 largest cities

Traveling Salesman Problem (TSP)

The problem

Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once.

Example

- Optimal tour visiting Germany's 15 largest cities
- Shortest among 43,589,145,600 possible tours

Traveling Salesman Problem (TSP)

The problem

Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once.

Example

- Optimal tour visiting Germany's 15 largest cities
- Shortest among 43,589,145,600 possible tours
- There is no known algorithm that can solve this problem efficiently

Traveling Salesman Problem (TSP)

The problem

Given a list of cities and their pairwise distances, the task is to find a shortest possible tour that visits each city exactly once.

Example

- Optimal tour visiting Germany's 15 largest cities
- Shortest among 43,589,145,600 possible tours
- There is no known algorithm that can solve this problem efficiently
- Runtime of best algorithm grows exponentially with number of cities

TSP With A Twist

B		F	
X	X	F	
F	X	X	F
S			

TSP With A Twist

B		*F*	
X	*X*	*F*	
F	*X*	*X*	*F*
S			

What is the distance between "F" ood items?

We need to compute the *shortest path* between two food items, considering the walls marked with "X"

Overall Strategy

- Compute the distances between each pair of food items and between “S” and “B” and the food items using a “shortest path” algorithm

Overall Strategy

- Compute the distances between each pair of food items and between “S” and “B” and the food items using a “shortest path” algorithm
- Try all possible combinations starting with “S”, going through all “F”, and ending in “S” and compute the travel distance

Overall Strategy

- Compute the distances between each pair of food items and between "S" and "B" and the food items using a "shortest path" algorithm
- Try all possible combinations starting with "S", going through all "F", and ending in "S" and compute the travel distance
- Remember the shortest distance obtained so far, and at the end, output that number

Background: Cryptography

Problem

Alice sends a message m to Bob. The message may be intercepted by Eve along the way.

Background: Cryptography

Problem

Alice sends a message m to Bob. The message may be intercepted by Eve along the way.

Solution

Alice encrypts m using a key k that is only known to Bob.

Background: Cryptography

Problem

Alice sends a message m to Bob. The message may be intercepted by Eve along the way.

Solution

Alice encrypts m using a key k that is only known to Bob.

Symmetric key cipher

$$\begin{aligned} c &= E(m, k) \\ m &= D(c, k) \end{aligned}$$

Engima—Cryptography in WW-I and WW-II

German encryption machine

The most common encryption technique in WW-II employed by the German military was based on an encryption machine called "Enigma".

Breaking Enigma

Weaknesses

Enigma had cryptographic weaknesses, but ultimately the following factors led to the breaking of codes:

- procedural flaws,
- operator mistakes,
- occasional captured machines and key tables

Breaking Enigma

Weaknesses

Enigma had cryptographic weaknesses, but ultimately the following factors led to the breaking of codes:

- procedural flaws,
- operator mistakes,
- occasional captured machines and key tables

Guessing messages

"Keine besonderen Vorkommnisse"
(in English: "No particular events")

Alan Turing

Turing's role in breaking German ciphers

Alan Turing worked at Bletchley Park, Britain's codebreaking centre.

Alan Turing

Turing's role in breaking German ciphers

Alan Turing worked at Bletchley Park, Britain's codebreaking centre.

"Bombe"

Turing devised a number of techniques for breaking German ciphers, including the method of the "bombe", an electromechanical machine that could find settings for the Enigma machine.

Alan Turing

Turing's role in breaking German ciphers

Alan Turing worked at Bletchley Park, Britain's codebreaking centre.

"Bombe"

Turing devised a number of techniques for breaking German ciphers, including the method of the "bombe", an electromechanical machine that could find settings for the Enigma machine.

Father of computer science

Turing provided an influential formalisation of the concept of the algorithm and computation with the *Turing machine*.

Breaking a Symmetric Key Cipher

Symmetric key cipher

$$\begin{aligned} c &= E(m, k) \\ m &= D(c, k) \end{aligned}$$

Breaking a Symmetric Key Cipher

Symmetric key cipher

$$
\begin{aligned}
c &= E(m, k) \\
m &= D(c, k)
\end{aligned}
$$

Attacker stituation

Eve gets hold of c, m, E and D, and wants to find k.

Breaking a Symmetric Key Cipher

Symmetric key cipher

$$\begin{aligned} c &= E(m,k) \\ m &= D(c,k) \end{aligned}$$

Attacker stituation

Eve gets hold of c, m, E and D, and wants to find k.

Strategy

Try all possible keys k_i and test:
$c = E(m, k_i)$?

Double Encoding

Idea

$$\begin{aligned} c &= E(E(m, k_1), k_2) \\ m &= D(D(c, k_2), k_1) \end{aligned}$$

Double Encoding

Idea

$$\begin{aligned} c &= E(E(m, k_1), k_2) \\ m &= D(D(c, k_2), k_1) \end{aligned}$$

Attacker stituation

Eve gets hold of c, m, E and D, and wants to find k_1 and k_2.

Double Encoding

Idea

$$
\begin{aligned}
c &= E(E(m, k_1), k_2) \\
m &= D(D(c, k_2), k_1)
\end{aligned}
$$

Attacker stituation

Eve gets hold of c, m, E and D, and wants to find k_1 and k_2.

Naive strategy

Try all possible keys k_1^i and k_2^j, an compute:
$c = E(E(m, k_1^i), k_2^j)$

Better Approach

Idea

Work forward and backward, and check if results match

Better Approach

Idea

Work forward and backward, and check if results match

Implementation

Try all possible keys k_1^i and compute:

$x = E(m, k_1^i)$

and

$x' = D(c, k_2^i)$

Better Approach

Idea

Work forward and backward, and check if results match

Implementation

Try all possible keys k_1^i and compute:
$x = E(m, k_1^i)$
and
$x' = D(c, k_2^i)$

Hash table

Use a hash table to remember all x values, and for each x' value, look in the hash table for a match.