HKOI 2014/15 Junior Q2 - Royal Bodyguard

Problem Author: Sampson Lee

Solution Writer: Alex Tung

The Problem

- There is a function that assigns 0 (FALSE) or 1 (TRUE) to all length-N binary strings (denote such string by `S[1..N]`)
- It is a 1-decision list that looks like:

```
if (S[p[1]] == d[1])
    return a[1];
else if (S[p[2]] == d[2])
    return a[2];
...
else if (S[p[N]] == d[N])
    return 1;
else
    return 0;
```

The Problem

- Your task is to find one set of values of p[], d[], and a[].
- p[1..N] is a permutation of {1, ..., N}
- d[i] is 0 or 1
- a[i] is 0 or 1

Sample I/O

`S[1..3]`	**Value (f(S))**
000	1
001	1
010	1
011	0
100	1
101	1
110	0
111	0

Output:

```
2 0 1
1 1 0
3 0
-----------------------
if (S[2] == 0)
    return 1;
else if (S[1] == 1)
    return 0;
else if(S[3] == 0)
    return 1;
else
    return 0;
```

Statistics

- Attempts: 30
- Mean: 7.566
- Max: 100 (percywtc)
- Standard deviation: 21.029

Considered a VERY HARD problem for junior...

Main obstacle is implementation

Algorithm 1: solving for p[j] = j

- Works for subtask 1 (30 points)

```
for i from 1 to (N - 1)
    set p[i] := i
    if f(S) is the same among all uncrossed S with S[i] = 0
        set d[i] := 0
        set a[i] := that common value
        cross out all S with S[i] = 0
    if f(S) is the same among all uncrossed S with S[i] = 1
        set d[i] := 1
        set a[i] := that common value
        cross out all S with S[i] = 1
set p[N] := N; set d[N] according to the two uncrossed strings
```

Example

S[1..3]	Value
000	1
001	1
010	1
011	1
100	1
101	0
110	0
111	0

```
i = 1

set p[1] := 1
all uncrossed S with S[1] = 0 has value 1
=>
set d[1] := 0
set a[1] := 1
cross all strings with S[1] = 0
```

Example

S[1..3]	Value
~~000~~	~~1~~
~~001~~	~~1~~
~~010~~	~~1~~
~~011~~	~~1~~
100	1
101	0
110	0
111	0

```
i = 2

set p[2] := 2
all uncrossed S with S[2] = 1 has value 0
=>
set d[2] := 1
set a[2] := 0
cross all strings with S[2] = 1
```

Example

`S[1..3]`	**Value**
~~000~~	~~1~~
~~001~~	~~1~~
~~010~~	~~1~~
~~011~~	~~1~~
10**0**	1
10**1**	0
~~110~~	~~0~~
~~111~~	~~0~~

```
i = 3

set p[3] := 3
set d[3] := 0
```

Algorithm 1: time complexity

- Ranging from $O(2^N)$ to $O(2^N N^2)$, depending on implementation
- Depends on:
  - How you maintain and iterate through the uncrossed strings
  - How you represent the strings (string? number?) and retrieve S[i]

Algorithm 2: based on algorithm 1

- Try all permutations p[1..N] of {1, 2, ..., N}
- Once the permutation is fixed, apply algorithm 1
- C++: `next_permutation()` can help

- Time complexity: $O(N!\ 2^N)$ to $O(N!\ 2^N N^2)$
- WAY too slow to get 100 points...

Algorithm 3: full solution

- Maintain a list of uncrossed strings
- For each i from 1 to (N - 1)
  - Find p[i] and d[i] s.t.
    - Function value is the same among all uncrossed strings S with S[p[i]] = d[i]
    - p[i] has not been chosen before (!)
  - Choose p[i], d[i], a[i]
  - Cross all strings with S[p[i]] = d[i]
- Set p[N] to be the remaining index
- Choose d[N] by looking at the two uncrossed strings

Example (Sample I/O)

`S[1..3]`	Value
000	1
001	1
010	1
011	0
100	1
101	1
110	0
111	0

```
i = 1

all uncrossed S with S[2] = 0 has value 1
=>
set p[1] := 2
set d[1] := 0
set a[1] := 1
cross all strings with S[2] = 0
```

Example (Sample I/O)

S[1..3]	Value
~~000~~	~~1~~
~~001~~	~~1~~
010	1
011	0
~~100~~	~~1~~
~~101~~	~~1~~
110	0
111	0

```
i = 2

all uncrossed S with S[1] = 1 has value 0
=>
set p[2] := 1
set d[2] := 1
set a[2] := 0
cross all strings with S[1] = 1
```

Example (Sample I/O)

`S[1..3]`	Value
~~000~~	~~1~~
~~001~~	~~1~~
01**0**	1
01**1**	0
~~100~~	~~1~~
~~101~~	~~1~~
11**0**	0
11**1**	0

```
Alternatively:

all uncrossed S with S[3] = 1 has value 0
=>
set p[2] := 3
set d[2] := 1
set a[2] := 0
cross all strings with S[3] = 1
```

Example (Sample I/O)

`S[1..3]`	Value
~~000~~	~~1~~
~~001~~	~~1~~
010	1
~~011~~	~~0~~
~~100~~	~~1~~
~~101~~	~~1~~
110	0
~~111~~	~~0~~

```
i = 3

set p[3] := 1
set d[3] := 0
```

The `Impossible` cases

Scenario 1: at some stage you cannot find `?`'s so that

`all uncrossed S with S[?] = ? has value ?`

Scenario 2: `i = N` but the two remaining strings have the same value

Algorithm 3: time complexity

- Ranging from $O(2^N N)$ to $O(2^N N^3)$, depending on implementation
- Extra factor of N is from finding p[i]
- Depends on:
  - How you maintain and iterate through the uncrossed strings
  - How you represent the strings (string? number?) and retrieve S[i]

Implementation Tips

- Read the strings **0-based**
- Convert the strings `str[0..N-1]` to numbers X in the range $[0, 2^N)$
- Note that the place value of the i-th position of `str` is 2^i

e.g. `str` = “10010”, corresponding X = $01001_2 = 9$

red 1 has place value 2^0

blue 1 has place value 2^3

- To check if the i-th position of `str` is 1, use

(C++): `(X & (1 << i)) > 0`

- `&` is bitwise AND, << is left-shift

Think about...

- Why does algorithm 3 work?
  - Will it ever return a wrong output?
  - Will it ever miss a valid output?