Combinatorial Game Theory

Jonathan Irvin Gunawan
National University of Singapore

prerequisite

bisa DP

bisa bitwise operation

yang gampang dulu deh

assume state (dan transisinya)

gamenya small enough

contoh : ada N batu di papan, tiap orang bisa ambil {1,4} batu.

gak bisa ambil = kalah

$1 \leq N \leq 10^6$

solusi : DP

```
bool result(State s)
{
   if (s has been computed before)
      return result[s]
   if (s == WIN) return WIN;
   if (s == LOSE) return LOSE;
   for each action a
      s' = state because action a
      if (result(s') == LOSE) return WIN;
   return LOSE;
}
```

gimana kalo $1 \leq N \leq 10^9$?

dikasih N angka. in each turn, tiap pemain boleh pop_front ato pop_back.

score tiap player = total semua angka yang dia pop

$1 \leq N \leq 10^3$

untuk tiap decision, lu assume worst case dari move musuh lu

```
int score(int a,int b)
{
  if (a > b) return 0;
  int &ret = dp[a][b];
  if (ret >= 0) return ret;
  ret = 0;
  //choose X[a] element
  ret = max(ret,X[a] + min(score(a+1,b-1),
                           score(a+2,b));
  //choose X[b] element
  ret = max(ret,X[b] + min(score(a+1,b-1),
                           score(a,b-2));
  return ret;
}
```

soal errr……

Rockethon 2015 A

player A punya n1 batu, player B punya n2 batu. in each turn :
player A boleh remove [1,k1] batu di pilenya sendiri,
player B boleh remove [1,k2] batu di pilenya sendiri
yang gak bisa jalan = kalah
A mulai duluan, siapa menang?

$1 \leq n1,n2 \leq 10^3$

nah sekarang baru soal

sebenernya

dikasih N tumpukan batu,
tumpukan ke-i ada Ai batu.
tiap turn boleh ambil berapapun
batu dari **tumpukan yang sama**
gak bisa ambil = kalah

```
3 4 5                Bob   takes 2 from A
1 4 5                Alice takes 3 from C
1 4 2                Bob   takes 1 from B
1 3 2                Alice takes 1 from B
1 2 2                Bob   takes entire A heap, leaving two 2s.
0 2 2                Alice takes 1 from B
0 1 2                Bob   takes 1 from C leaving two 1s. (In mi
0 1 1                Alice takes 1 from B
0 0 1                Bob   takes entire C heap and wins.
```

banyak state = eksponensial

gak bisa pake solusi sebelumnya

define nim-sum =

xor values of all piles

contoh : piles = {2, 5, 1, 7, 3}

nim-sum =

$2 \oplus 5 \oplus 1 \oplus 7 \oplus 3 = 2$

kalo nim-sum > 0,
it’s a winning state,
otherwise losing state

```
int win(vector<int> state)
{
   int res = 0;
   for (int x : state) res ^= x;
   return res > 0;
}
```

udah gitu doang

ide proof :
dari sebuah state yang nim-sum
= 0, move apapun pasti
statenya jadi nim-sum ≠ 0

ide proof :
other way around, dari sebuah state yang nim-sum ≠ 0, ada move yang bikin nim-sum = 0

ide proof :

since losing state nim-sum = 0,

yang bisa maintain nim-sum = 0

buat musuhnya => menang

dari sebuah state yang nim-sum = 0, move apapun pasti statenya jadi nim-sum ≠ 0

$A1 \oplus A2 \oplus \ldots \oplus An = 0$

suppose ambil $y \geq 0$ batu dari Ax.

assume resulting nim-sum = 0

$A1 \oplus A2 \oplus \ldots \oplus (Ax - y) \oplus \ldots \oplus An = 0$

$A1 \oplus A2 \oplus \ldots \oplus (Ax - y) \oplus \ldots \oplus An = A1 \oplus A2 \oplus \ldots \oplus An$

$(Ax - y) = (Ax)$

$y = 0$

contradiction

dari sebuah state yang nim-sum ≠ 0, ada
move yang bikin nim-sum = 0

yang ini agak ribet. suppose current nim-sum = s
suppose d = most significant bit
dimana bit ke-d of s = 1
ambil k apapun yang memenuhi
bit ke-d of Ak = 1
(pasti ada such k yang memenuhi)

dari sebuah state yang nim-sum ≠ 0, ada
move yang bikin nim-sum = 0

kita ganti Ak jadi Ak ⊕ s
move ini legal, karena Ak ⊕ s < Ak

new nim-sum =
A1 ⊕ A2 ⊕ … ⊕ (Ak ⊕ s) ⊕ … ⊕ An
A1 ⊕ A2 ⊕ … ⊕ An ⊕ s
s ⊕ s
0

contoh

```
18:  1  0  0  1  0
 6:  0  0  1  1  0
 3:  0  0  0  1  1
20:  1  0  1  0  0
 9:  0  1  0  0  1
---------------- ⊕
 s:  0  1  0  1  0
```

```
18:  1  0  0  1  0
 6:  0  0  1  1  0
 3:  0  0  0  1  1
20:  1  0  1  0  0
 3:  0  0  0  1  1
---------------- ⊕
 t:  0  0  0  0  0
```

example taken from suhendry.net

aplikasi2

instead of harus ngambil batu,
lu boleh nambah batu.
to ensure game bakal finish,
cuma boleh nambah K batu
total

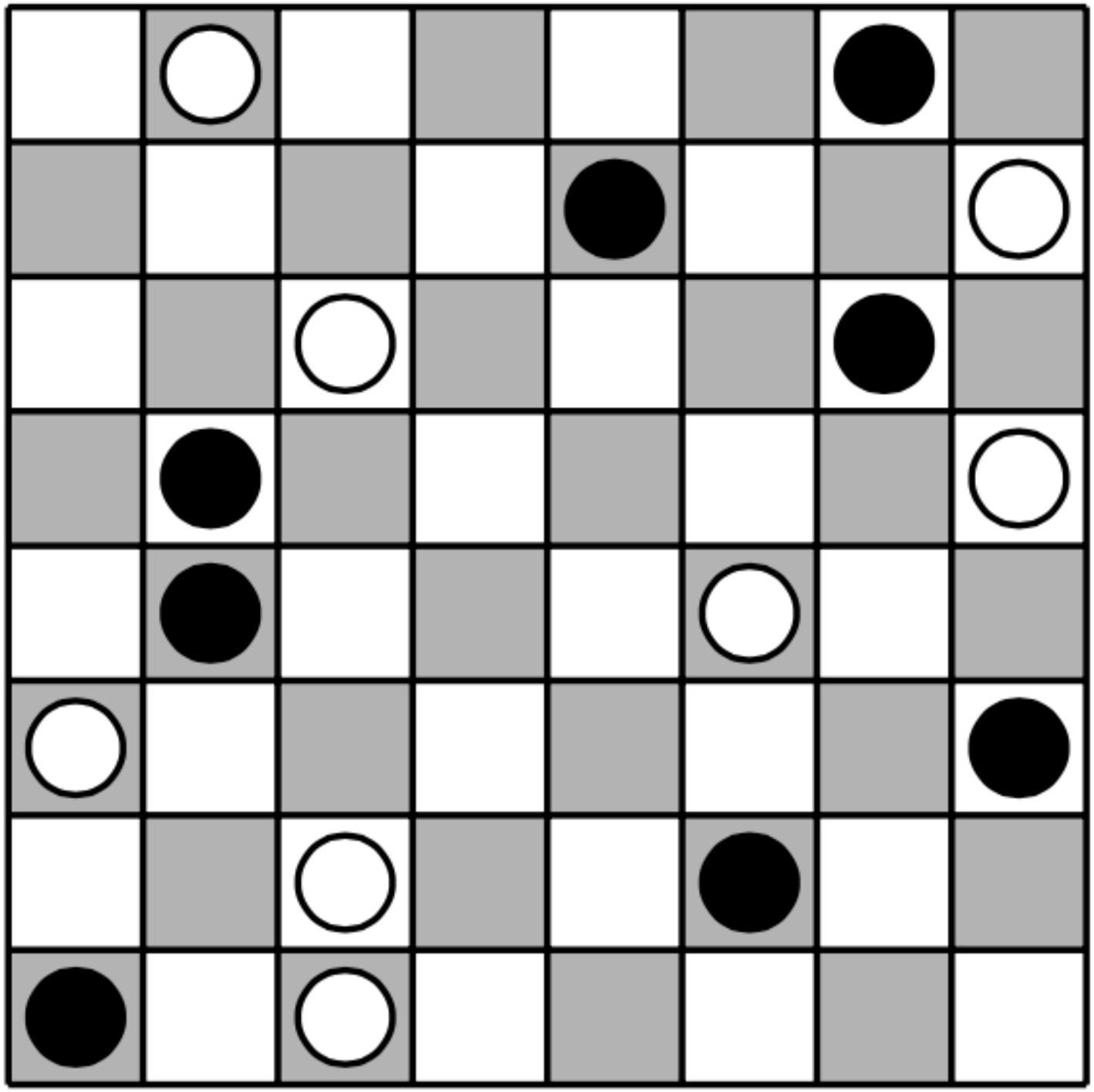
solusi : lakuin nim biasa. kalo player pake move nambah batu, turn berikutnya tinggal “undo” movenya

basically proof yang sebelumnya masih holds walaupun bisa nambah

aplikasi lagi

dikasih checkerboard. di tiap row ada exactly satu bidak hitam dan satu bidak putih

in each turn : tiap player boleh gerakin **piecenya sendiri** ke row yang sama, tapi gak boleh ngelewatin piece musuhnya



this is not an impartial game.
but we can still solve it with nim
though

aplikasi lagi

ada N kotak dinomerin [1,N] dari kiri ke kanan. tiap kotak initally either kosong ato ada koin

satu move : pindahin koin manapun ke kotak dikirinya manapun, tapi gak boleh “melewati” koin

contoh : ada koin di posisi {5,7,8,12}

satu move :

koin yang di 5 pindah ke either

{1,2,3,4}

koin yang di 12 pindah ke either

{9,10,11,12}

kita anggap gap tiap koin = banyaknya batu pada sebuah tumpukan

koin = {5,7,8,12}

nim-instance = {4,1,0,3}

ets gak bisa, tiap kita pindah koin, berarti ada satu tumpukan berkurang batunya, satu tumpukan nambah batunya dong?

kita ambil selang seling, ambil paling kanan, buang kedua terkanan, ambil ketiga terkanan, buang keempat terkanan, ……

koin = {5,7,8,12}

nim-instance = {4,1,0,3}

kita ganti jadi

nim-instance = {1,3}

tiap move jadi either nambah batu atau remove batu pada pile yang sama kan?

Grundy number

a.k.a. nimber

we changed the problem a bit

suppose untuk tiap turn, kalian cuma boleh ambil either {1,2,4,5} batu di **tumpukan yang sama**

disini kita punya N tumpukan independen :

allowed move dari tiap tumpukan tidak bergantung tumpukan lain

kita hitung nimber dari tiap tumpukan

nimber(s) = bilangan asli terkecil yang tidak terdapat pada {nimber(s’), s’ semua state yang bisa diraih dari s dengan satu turn}

kalo s = no move = losing state =>
gak ada s’ yang memenuhi =>
nimber(s) = 0

contoh:

nimber(0) = 0

nimber(1) : next state : {0} = 1

nimber(2) : next state : {0,1} = 2

nimber(3) : next state : {1,2} = 0

nimber(4) : next state : {0,2,3} = 1

nimber(5) : next state : {0,1,3,4} = 2

kita hitung nimber tiap tumpukan,
terus kita bisa pandang gamenya
sebagai nim biasa, dimana
Ai = nimber(Xi)

proof

pake papan tulis aja vin

soal lagi deh

di chessboard, dikasih N queen di papan. tiap move, player boleh ambil queen manapun dan gerakin queennya

tapi queennya “cacat”, cuman bisa gerak ke either {kiri, bawah, atau kiribawah}
queen boleh saling tumpuk

soal lagi deh

soal

dikasih N tumpukan batu. tumpukan ke-i ada Xi batu.

pada setiap turn, lu boleh ambil Y batu dari Xi, **dimana Y adalah faktor dari Xi** kurang dari Xi

```
int nimber(int x)
{
  vector<int> adjNimber;
  for (i faktor dari x)
    adjNimber.insert(nimber(x-i))
  sort adjNimber, remove duplicate
  for (i := 0 to adjNimber.size()-1)
    if (i != adjNimber[i]) return i
  return adjNimber.size()
}
```

masalah :

$1 \leq Ai \leq 1\ 000\ 000\ 000$

cari cara cepet buat nyari nimbernya

in this case : nimber(x) = (x & -x)

proof : papan tulis aja deh

EOF

Q&A?