DP bitmask + profile

Jonathan Irvin Gunawan
National University of Singapore

prerequisite

tau DP

tau bitmask / bitwise operation

tau graph

ada beberapa soal tentang graph di slide ini dan gw bakal re-define apa itu path, vertex, dll.

bisa ngitung

quick review on bitmask

gimana caranya :

1. ngecek bit ke-y? **X & (1 << Y)**

2.ngecek ada berapa bit yang nyala?

__builtin_popcount(X)

3. nyalain bit ke-y? **X | (1 << Y)**

4. sisain bit ke-y doang? **X & (1 << Y)**

5. toggle bit ke-y? **X ^ (1 << Y)**

6. matiin bit ke-y? **X & (-1 - (1 << Y))**

extra :

7. cari y manapun s.t. bit ke-y nyala **X & (-X)**

basically, hari ini kita akan belajar DP
yang statenya pake bitmask

langsung contoh soal aja deh ya

soal klasik di DP bitmask

TSP / Hamiltonian Path

TSP = Travelling Salesman Problem

TSP / Hamiltonian Path

dikasih weighted graph, lu mau bikin sebuah path s.t. pathnya mengunjungi semua node tepat sekali dan total weightnya minimum

$1 \leq N \leq 16$

TSP / Hamiltonian Path

From Wikipedia, the free encyclopedia

The **travelling salesman problem** (**TSP**) asks the following question: *Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?* It is an NP-hard problem

TSP / Hamiltonian Path

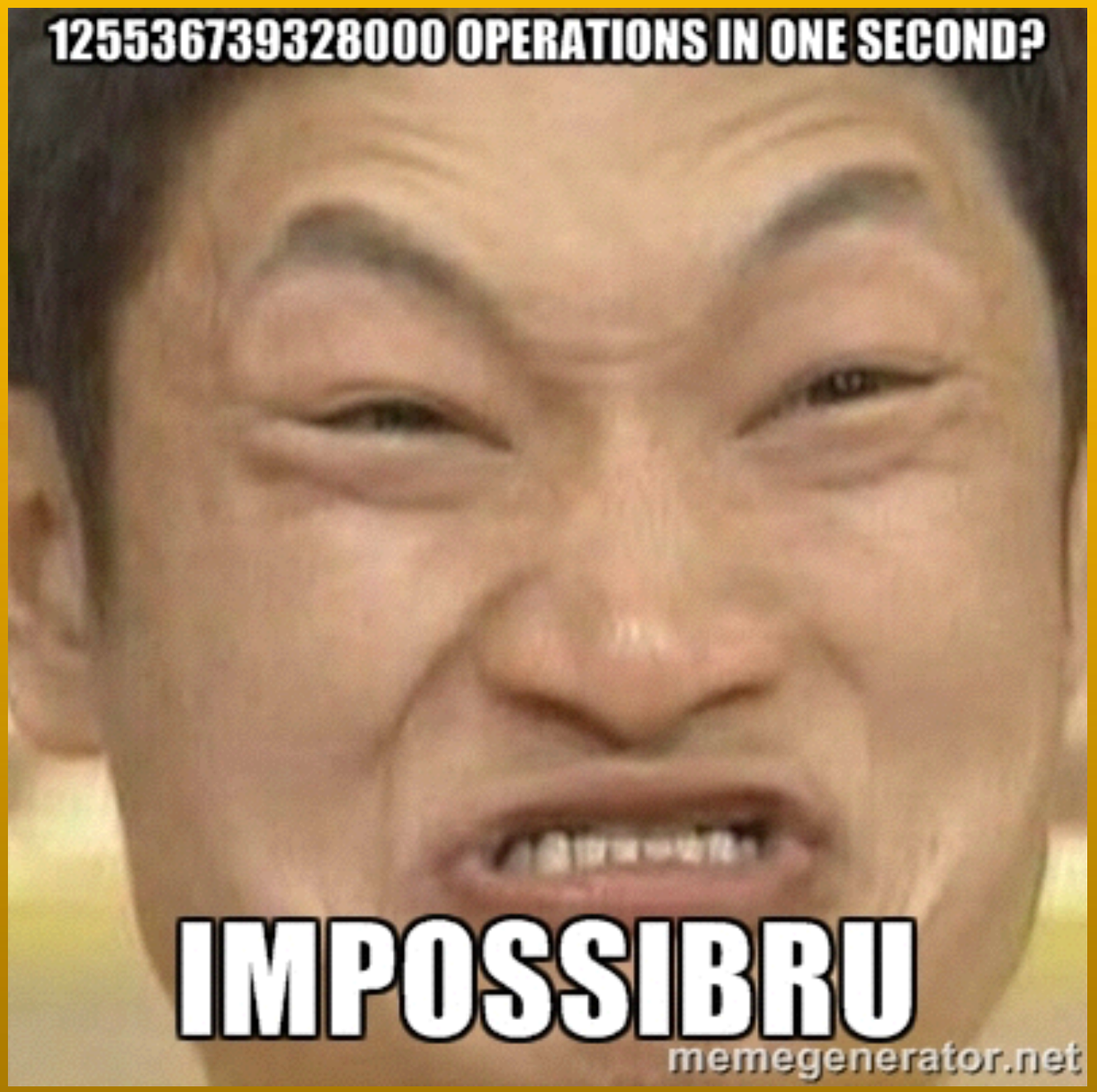
kalo bruteforce bener2

banyaknya kemungkinan : O(N!)

tiap kemungkinan harus ngecek valid ato gak, dan hitung total cost nya berapa : O(N)

O(N! * N)

16! * 16 = 12553673932800O



kita pake DP buat solve soal ini

state :

1. sekarang lagi di node mana
2. node yang udah di visit mana aja

notice this will overlap

1 -> 2 -> 3 -> 4 : (4, {1,2,3,4})

1 -> 3 -> 2 -> 4 : (4, {1,2,3,4})

dp(now, visited)

base case?

kalo visited berisi N elemen (in other words, semua N nodes udah visited)
return 0;

dp(now, visited)

rekurens?

untuk semua v neighbour dari now yang gak di visited, cari minimum dari dp(v, visited + v) + w(now,v)

dp(now, visited)

```
if (__builtin_popcount(visited)==N) {
  return 0;
}
int &ret = dp[now][visited]; // teknik dp kayak gini tau kan ya
if (ret >= 0) return ret;
ret = INFINITY;
for (pair<int,int> v : adj[now]) {
  if (((visited & (1 << v.first))) == 0) {
    ret = min(ret,
              dp(v.first, visited | (1 << v.first) + v.second);
  }
}
return ret;
```

dp(now, visited)

kompleksitas :

state : O(2^N * N)

transisi : O(N)

total : O(2^N * N^2) <— palindrom

dp(now, visited)

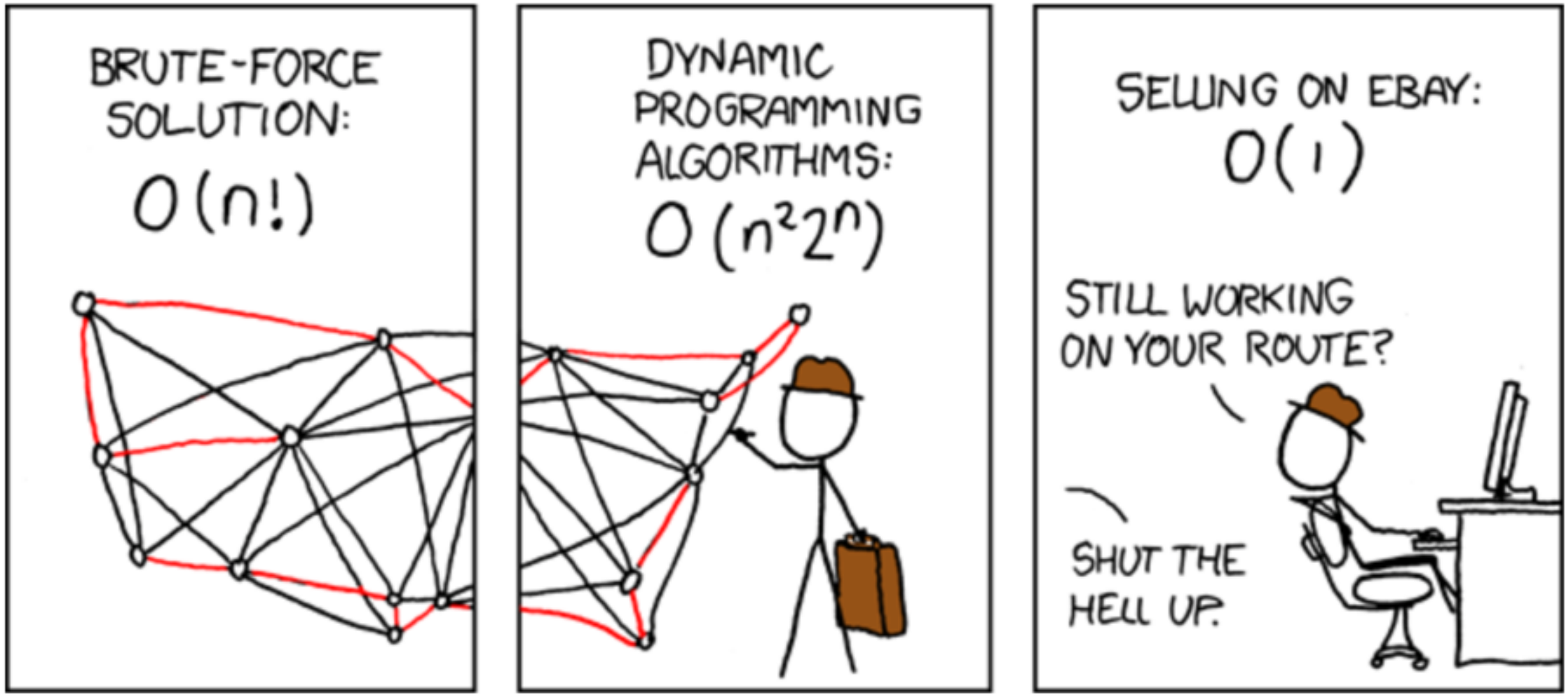
```
if (__builtin_popcount(visited)==N) {
  return 0;
}
int &ret = dp[now][visited];
if (ret >= 0) return ret;
ret = INFINITY;
for (pair<int,int> v : adj[now]) {
  if (((visited & (1 << v.first))) == 0) {
    ret = min(ret,
              dp(v.first, visited | (1 << v.first) + v.second);
  }
}
return ret;
```

bisa dioptimisasi dikit
cek bit2 yang mati aja

optimisasi?

1. pake teknik x & (-x)
2. precompute list of on/off bits untuk semua bilangan

Travelling Salesman Problem



ntar bisa aja soalnya bisa statenya ga 2^N

tapi 3^N

statenya bisa visited, unvisited, plus unknown ato apa gitu

dp(now, visited)

```
if (__builtin_popcount(visited)==N) {
  return 0;
}
int &ret = dp[now][visited];
if (ret >= 0) return ret;
ret = INFINITY;
for (pair<int,int> v : adj[now]) {
  if (((visited & (1 << v.first))) == 0) {
    ret = min(ret,
              dp(v.first, visited | (1 << v.first) + v.second);
  }
}
return ret;
```

jadi ga bisa pake bit operation doang

dp(now, visited)

```
if (__builtin_popcount(visited)==N) {
  return 0;
}
int &ret = dp[now][visited];
if (ret >= 0) return ret;
ret = INFINITY;
for (pair<int,int> v : adj[now]) {
  if (((visited & (1 << v.first))) == 0) {
    ret = min(ret,
              dp(v.first, visited | (1 << v.first) + v.second);
  }
}
return ret;
```

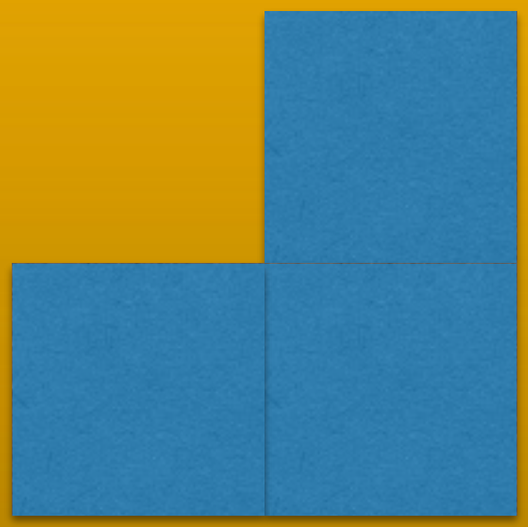
harus bikin sendiri buat 3 pangkat

DP profile

nih ya sekarang

langsung contoh soal deh

dikasih grid R*C. tiap sel ada angkanya. ambil beberapa sel s.t. total angkanya maksimum dan sel2 yang lu ambil gak ada yang berbentuk



contoh

5	6	1	1
7	7	1	1

jawabannya 22, ambil yang ijo.
kalo lu ambil yang biru manapun sekarang, bakal violate constraint problemnya

state :
1. sekarang dimana
2. sel kiri diambil/kagak
3. sel atas diambil/kagak

cukup kan buat determine lu boleh ambil sel sekarang ato kagak

cuman O(RC*4)
polynomial yah. hore

gimana transisinya?

x = sel atas

y = sel kiri

kalo abis dari (r,c,x,y)

transisi ke (r,c+1,x’,y’)

y’ = decision dari sel (r,c)

tapi cara taunya x’ gimana?

yah berarti statenya salah dong

ya iyalah kalo statenya emang gitu, soal ini gak bakal gw bahas.

kan hari ini materinya dp bitmask something yang bau2nya eksponensial

state salah :

1. sekarang dimana

2. sel kiri diambil/kagak

3. sel atas diambil/ kagak

state benar :

1. sekarang mau nyoba2in sel yang mana
2. sel di kiri diambil ato kagak
3. sel dikirinya lagi diambil ato kagak
4. sel dikirinya lagi diambil ato kagak

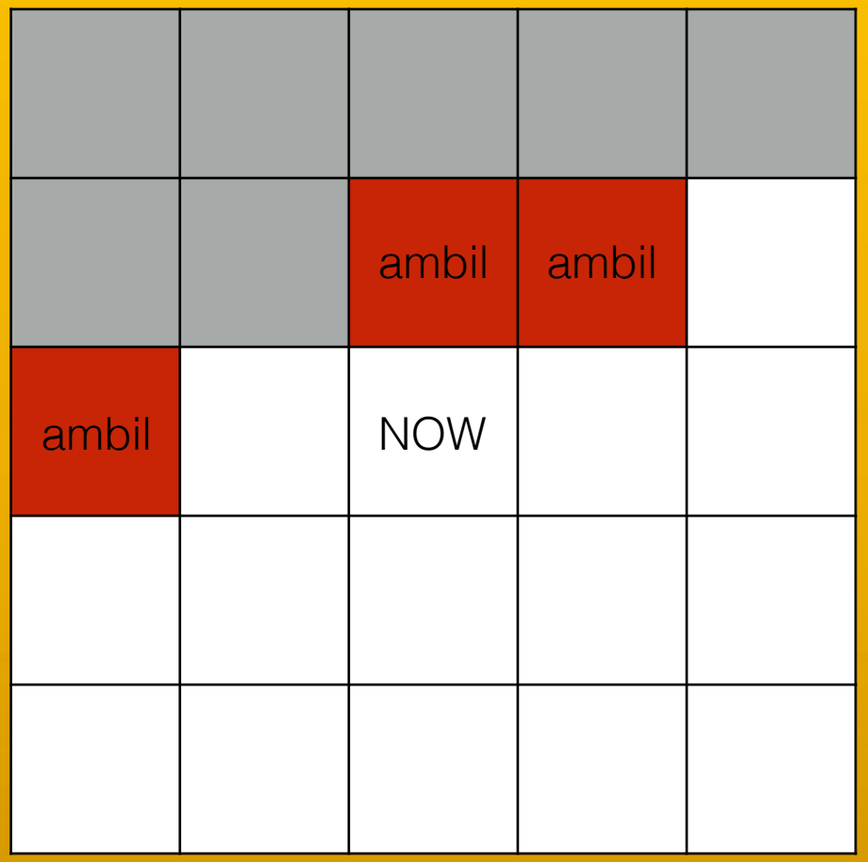
.....

5. sel diatasnya diambil ato kagak

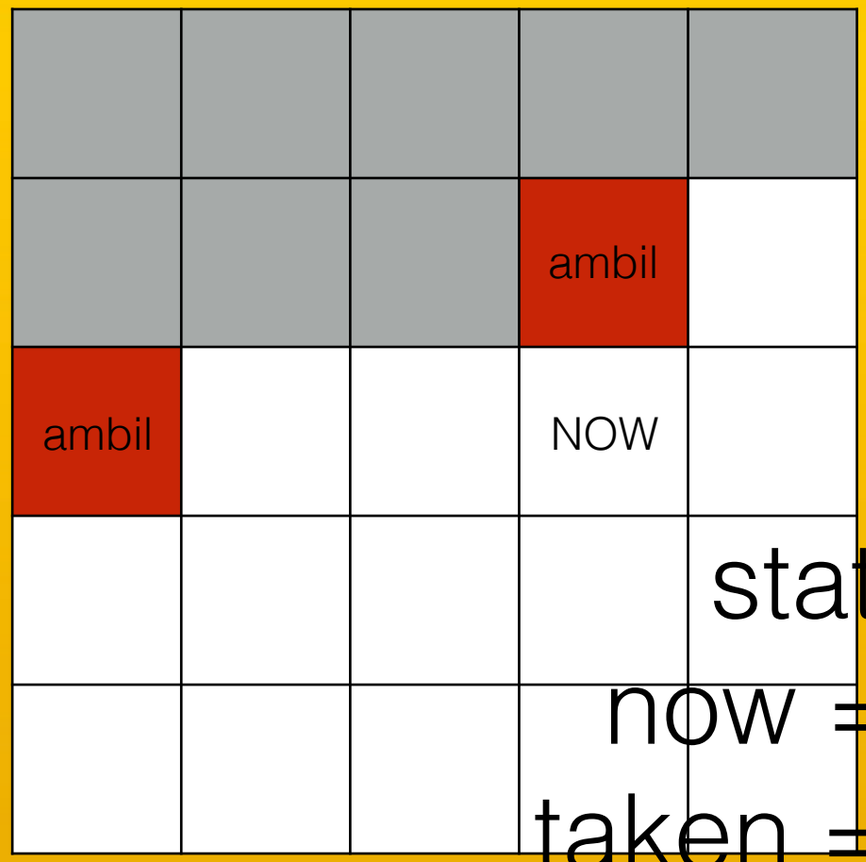
		ambil	ambil	gak ambil
ambil	gak ambil	NOW		

state =
now = (3,3)
taken = 11010

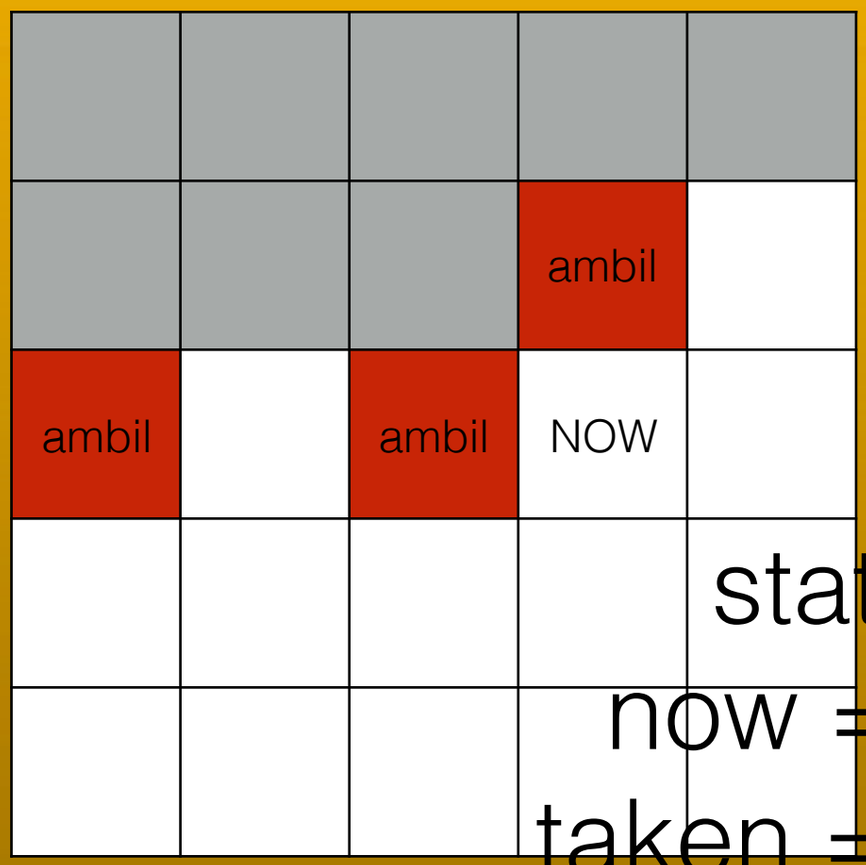
sel yang valuenya kita dah gak perlu tau



state =
now = (3,3)
taken = 11010



state =
now = (3,4)
taken = 10100



state =
now = (3,4)
taken = 10101

dp(now, mask)

```
if (now.r == R + 1) return 0;
int &ret = dp[now][mask];
if (ret >= 0) return ret;
int next_mask = (mask << 1) % (1 << C);
ret = dp(next(now), next_mask); //not taking the current cell
if (!(mask & 1) || !(mask & (1 << (C-1)))) {
  // either left cell or top cell is not taken
   ret = max(ret, dp(next(now), next_mask + 1) + isi[now]);
}
return ret;
```

coba ya latihan soal

maximum independent set

dikasih sebuah graph. tiap vertex ada weightnya. pilih beberapa vertex sedemikian sehingga total weightnya maksimum dan tidak ada vertex yang adjacent

state :
node mana aja yang
udah pasti dipilih

dp(used)

```
int &ret = dp[used];
if (ret >= 0) return ret;
for (int i = 0; i < N; ++i) {
 for (int j = 0; j < N; ++j) {
  if (adj[i][j] && (used & (1 << i)) && (used & (1 << j))) {
   return ret = 0;
  }
 }
}
ret = __builtin_popcount(used);
for (int i = 0; i < N; ++i) {
 if (used & (1 << i)) continue;
 ret = max(ret, dp(used ^ (1 << i)));
 // ret = max(ret, dp(used | (1 << i)));
 // ret = max(ret, dp(used + (1 << i)));
}
return ret;
```

maximum weighted bipartite matching

statenya :
1. sekarang kita lagi mau cobain node kiri yang mana
2. node kanan mana aja yang udah di match

dp(now, used)

```
if (now == N) return 0;
int &ret = dp[now][used];
if (ret >= 0) return ret;
ret = 0;
for (int i = 0; i < N; ++i) {
 if (used & (1 << i)) continue;
 ret = max(ret, dp(now + 1, used | (1 << i)) + w[now][i]);
}
return ret;
```

GCJ APAC 2015 C

gGames

dikasih N ($1 \leq N \leq 16$) orang, kita mau bikin knock-out turnamen.
tiap orang punya Ki dan beberapa temen.

kita mau bikin turnamennya sedemikian sehingga apapun yang terjadi gak ada orang i yang ketemu temennya sebelum Ki round

4 orang

K(1) = 0

K(2) = 0

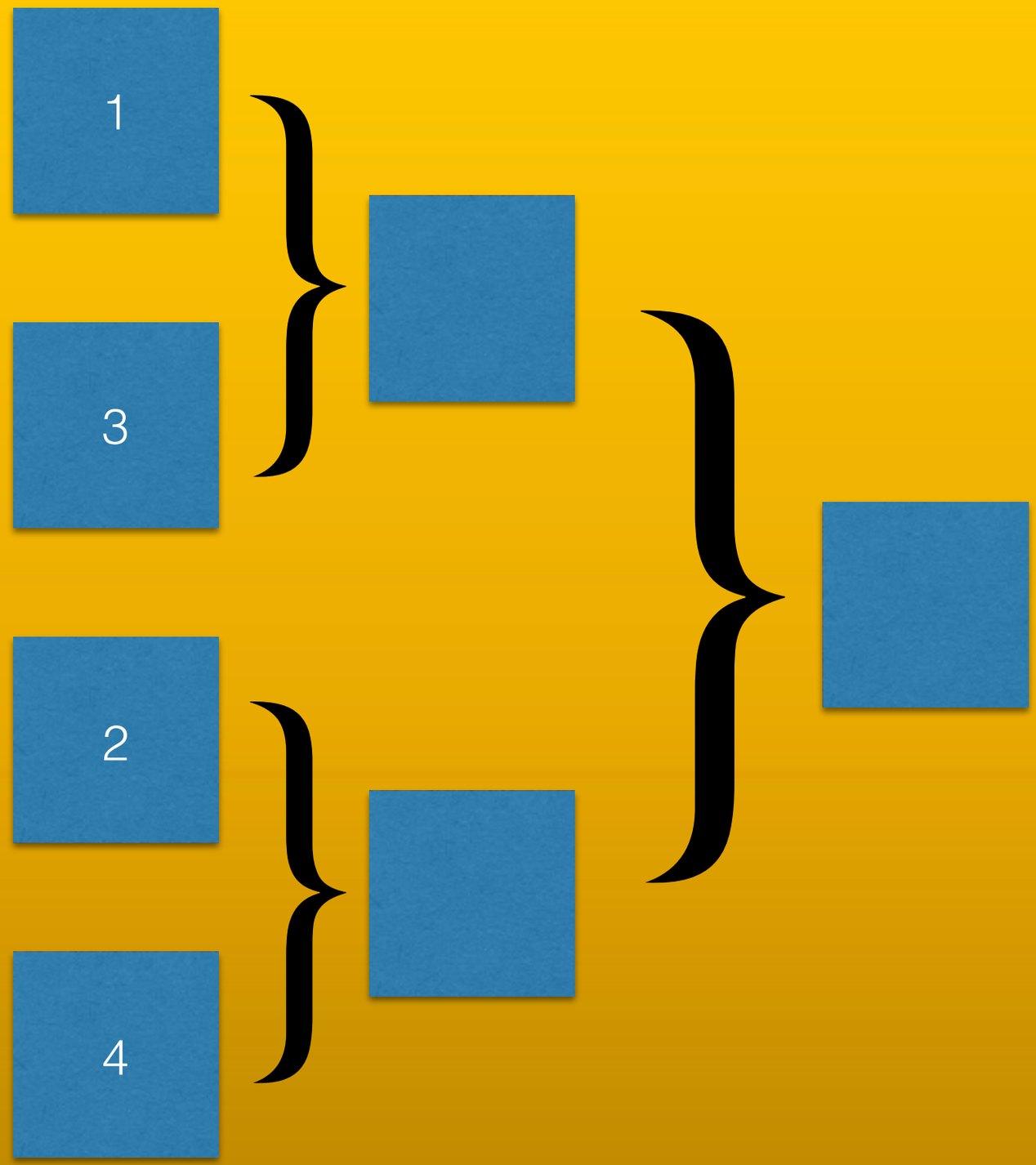
K(3) = 1

K(4) = 0

1-2 temen

2-4 temen

3-4 temen



4 orang

K(1) = 2

K(0) = 0

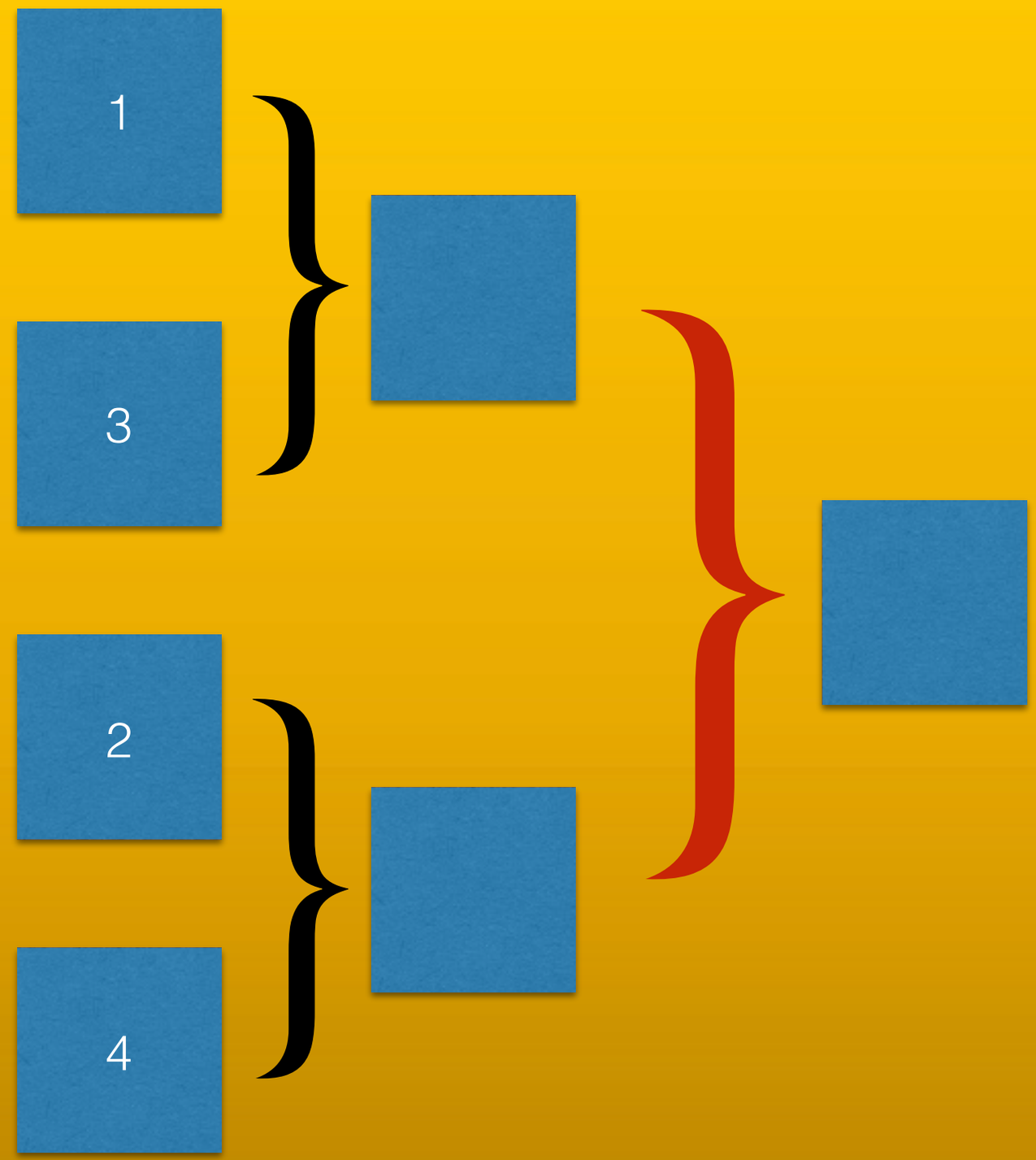
K(3) = 1

K(4) = 0

1-2 temen

2-4 temen

3-4 temen



4 orang

K(1) = 0

K(2) = 0

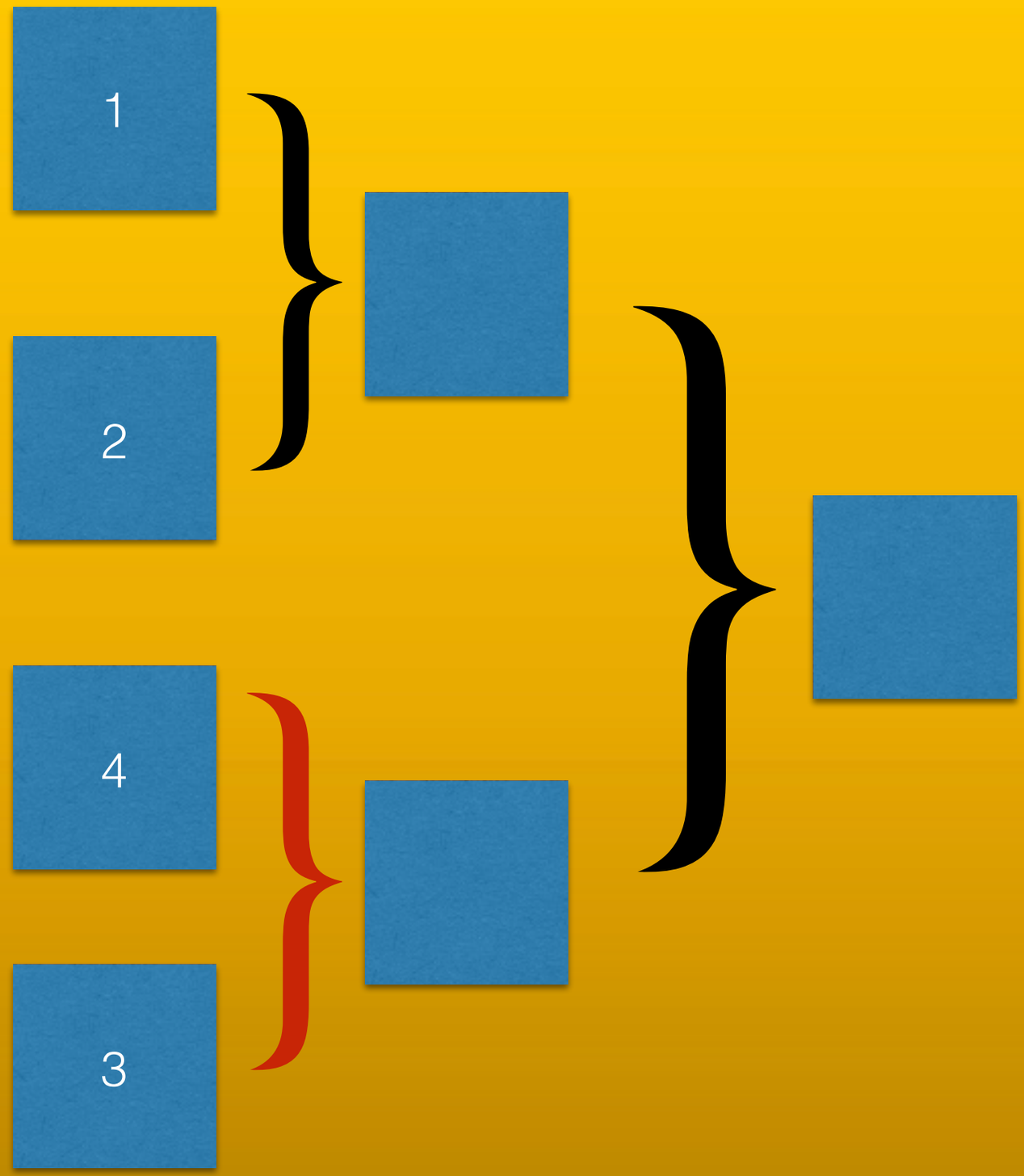
K(3) = 1

K(4) = 0

1-2 temen

2-4 temen

3-4 temen



NO SOLUTION

4 orang

K(1) = 0

K(2) = 0

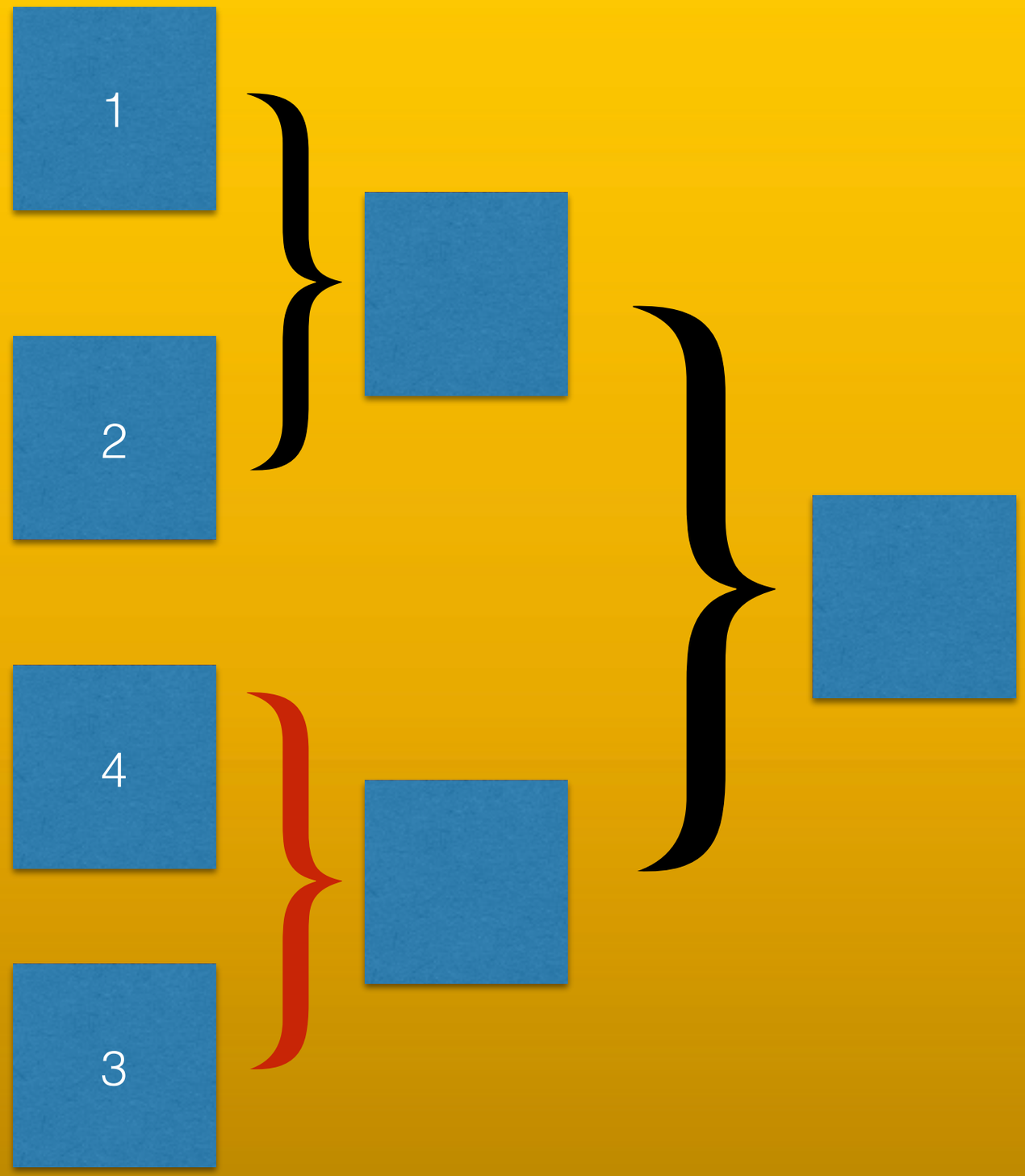
K(3) = 0

K(4) = 2

1-4 temen

2-4 temen

3-4 temen




10:00

state :
orang2 mana aja yang mau dibikin turnament

bisa deduce lagi round keberapa. kalo ada orang-i dan orang-j yang temenan dan round≤Ki,
return 0

base case :
kalo cuma satu orang, return 1

otherwise, coba semua kemungkinan split orang2 itu ke dua group.

state : O(2^N)

transisi dalam state : O(2^N)

total : O(4^N)?

nononono, tiap state kan ngga 2^N juga

tapi 2^{banyaknya orang didalam state}

$$\sum_{i=0}^{2^n-1} 2^{\text{--builtin\_popcount(i)}}$$

$$\sum_{i=0}^{n} \binom{n}{i} 2^i$$

$$\sum_{i=0}^{n} \binom{n}{i} 1^{n-i} 2^i$$

$$(1+2)^n$$

$$(3)^n$$

jadi O(3^N)
bisa sampe 17
3^17 = 129.140.163

kalo 4^N cuma bisa sampe 13

DP profile

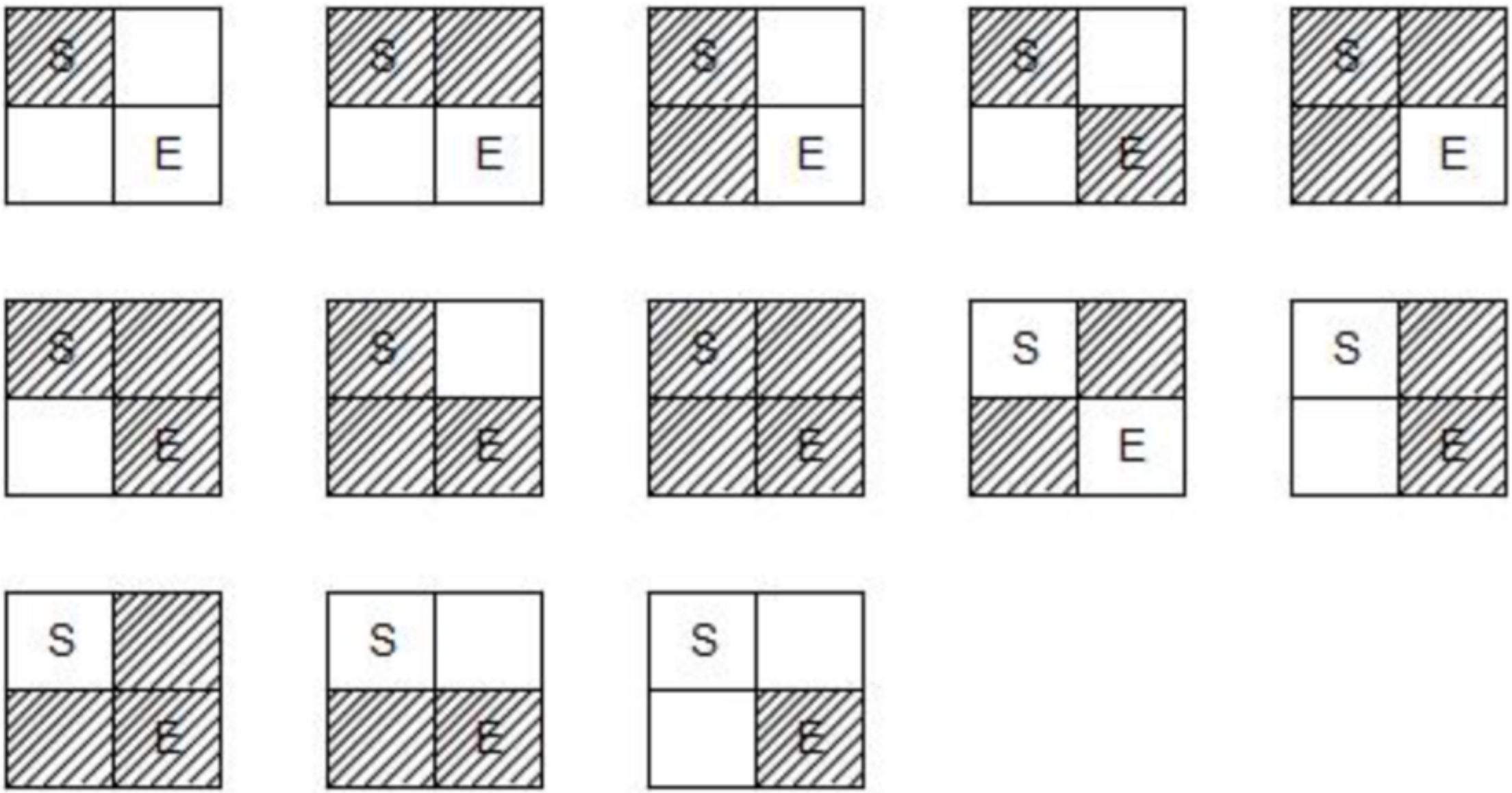
ICPC Jakarta 2014

dikasih grid R*C, tentuin ada berapa cara untuk blok at most K sel sedemikian sehingga lu ga bisa jalan dari top left ke bottom right

jalannya cuma bisa kanan ato bawah

contoh

R = 2, C = 2, K = 4




6:00

state :

1. sekarang mau nyoba2 sel yang mana
2. sel kiri bisa divisit ato engga
3. sel kirinya lagi bisa divisit ato engga

...

4. sel atas bisa divisit ato engga
5. udah blok berapa sel

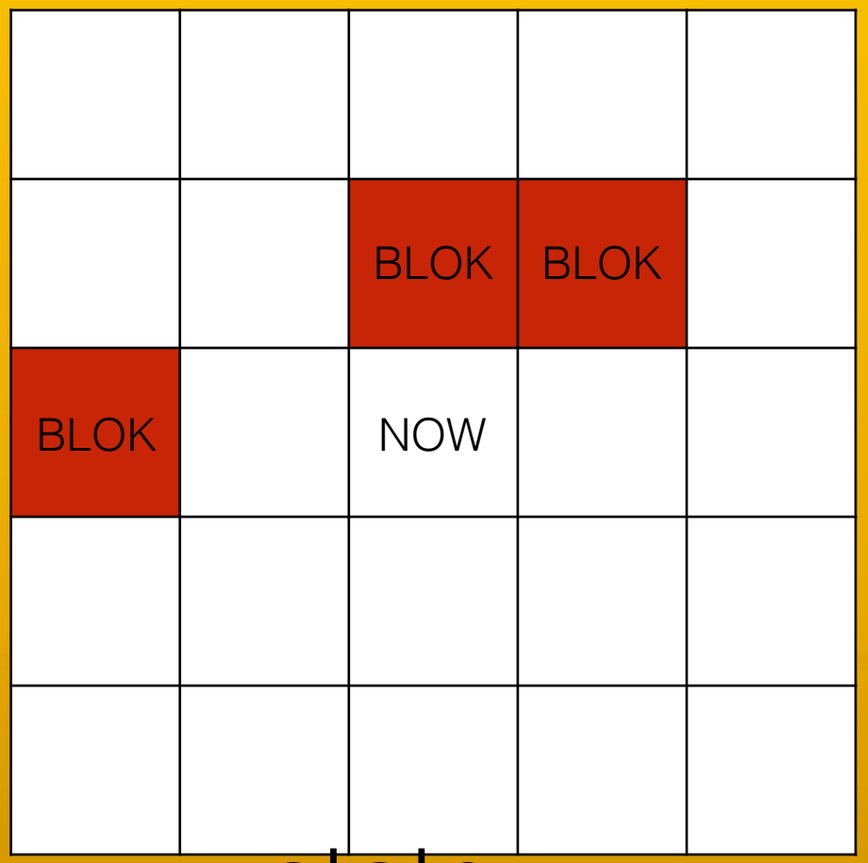
base case :
kalo udah nyampe sel terakhir, return 1 kalo gak visitable dan gak blok lebih dari K, return 0 kalo visitable

rekurens :
cobain mau/engga mau blok sel sekarang. update mask accordingly. kalo sel kiri dan sel atas gak visitable, apapun decisionnya, current sel bakal jadi unvisitable

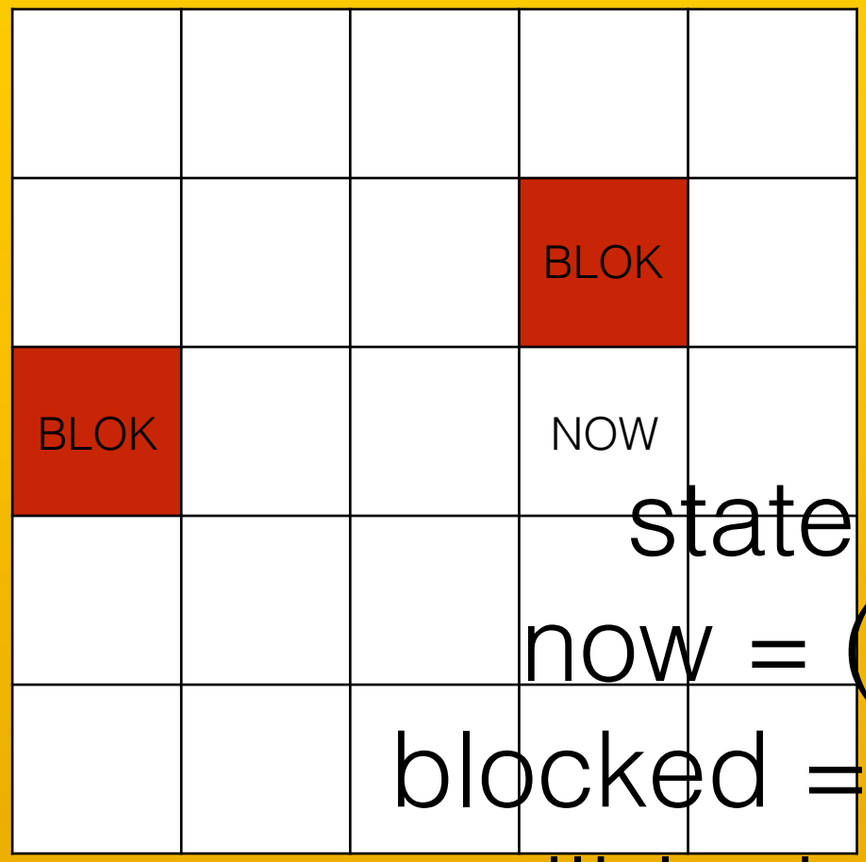
		BLOK	BLOK	
BLOK		NOW		

state =
now = (3,3)
blocked = 11010
cellblocked = 2

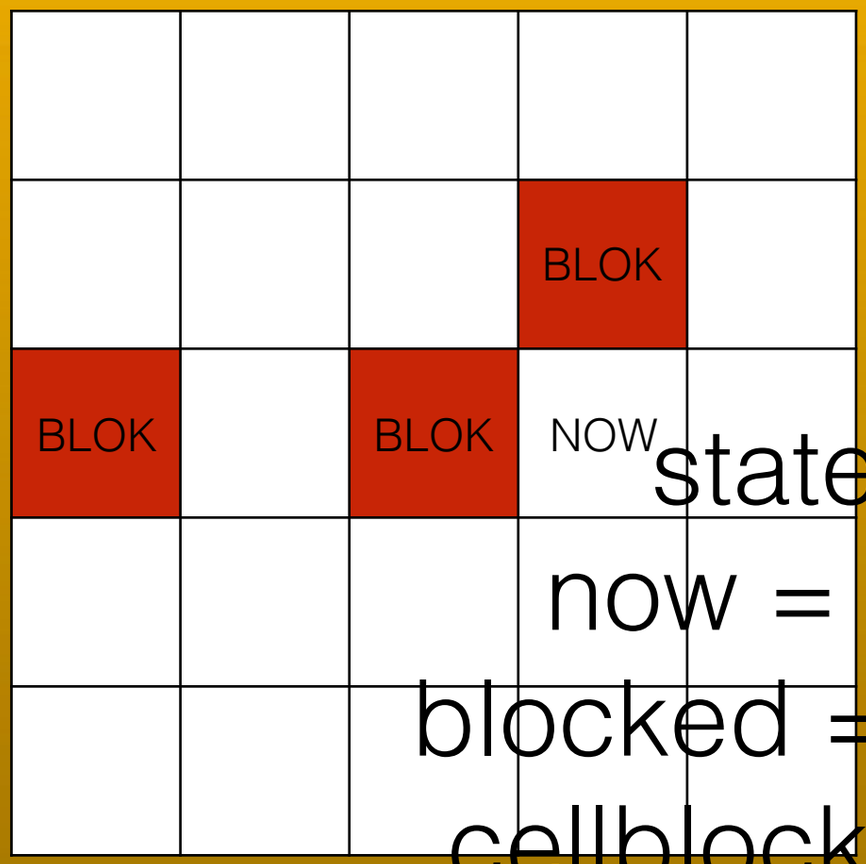
BLOK means unvisitable



state =
now = (3,3)
blocked = 11010
cellblocked = 2



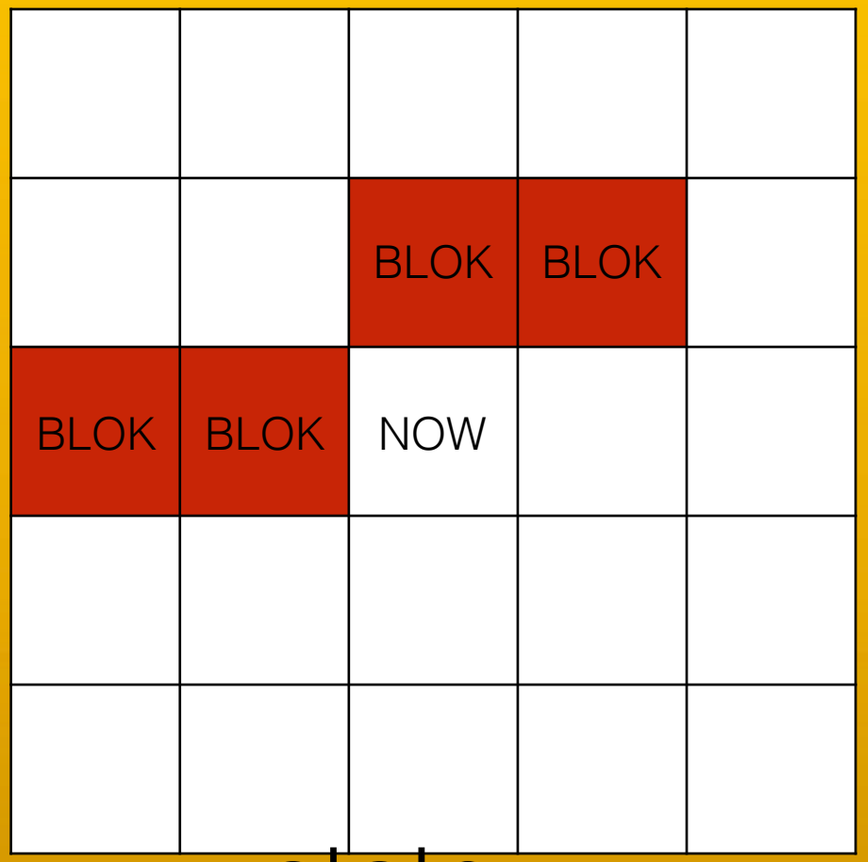
state =
now = (3,4)
blocked = 10100
cellblocked = 2



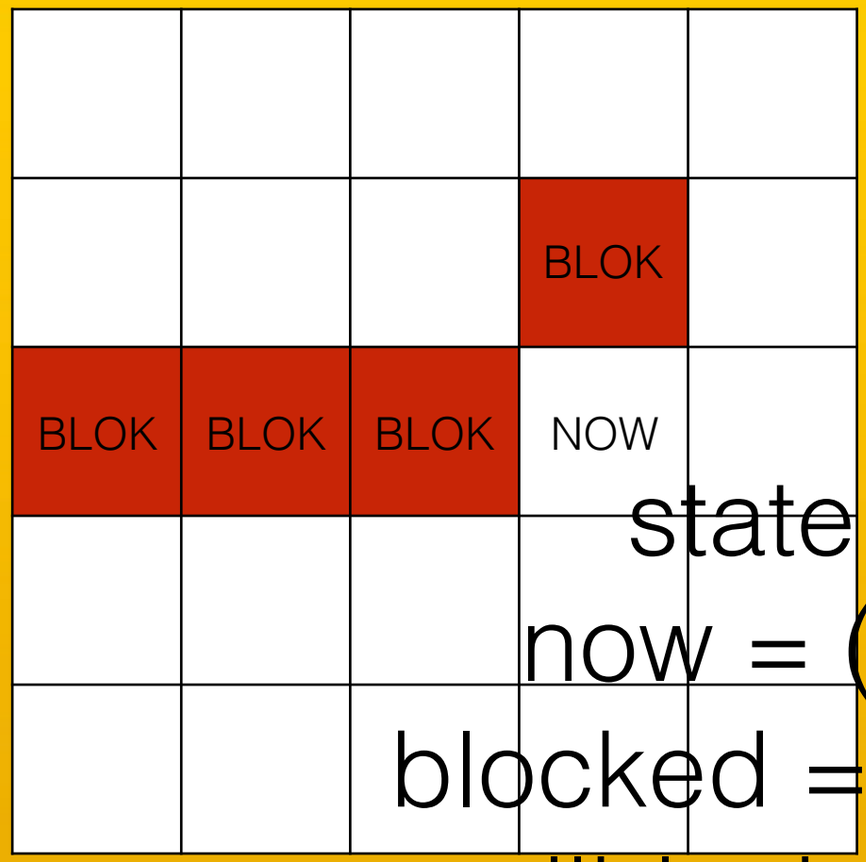
state =
now = (3,4)
blocked = 10101
cellblocked = 3

		BLOK	BLOK	
BLOK	BLOK	NOW		

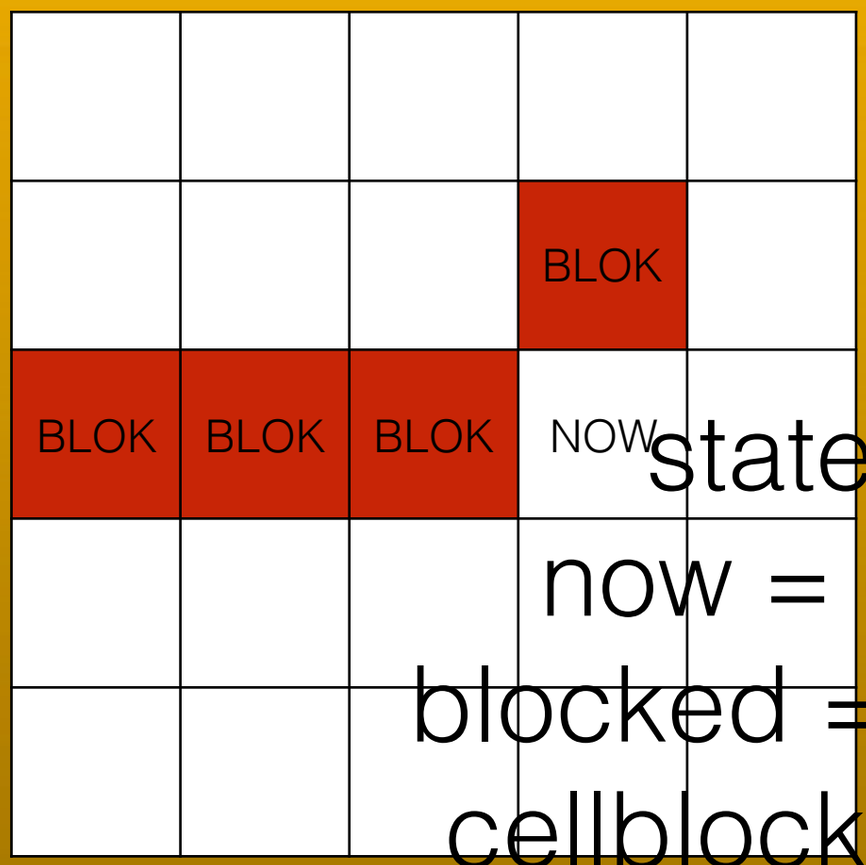
state =
now = (3,3)
blocked = 11011
cellblocked = 2



state =
now = (3,3)
blocked = 11011
cellblocked = 2



state =
now = (3,4)
blocked = 10111
cellblocked = 2



state =
now = (3,4)
blocked = 10111
cellblocked = 3

EOF

Q&A?