

DP bitmask + profile

Jonathan Irvin Gunawan
Google Asia Pacific, Pte. Ltd.

E

EE

EEEE

JONATHAN
IRVINGANTENGBANGET

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? $X \& (1 \ll Y)$

2. ngecek ada berapa bit yang nyala?

__builtin_popcount(X)

3. nyalain bit ke-y? $X \mid (1 \ll Y)$

4. sisain bit ke-y doang? $X \& (1 \ll Y)$

5. toggle bit ke-y? $X \wedge (1 \ll Y)$

6. matiin bit ke-y? $X \& (-1 - (1 \ll Y))$

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 : $O(N)$

$$O(N! * N)$$

$$16! * 16 = 125536739328000$$



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 : $O(2^N * N^2)$ \leftarrow 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

[<](#)[< PREV](#)[RANDOM](#)[NEXT >](#)[>](#)

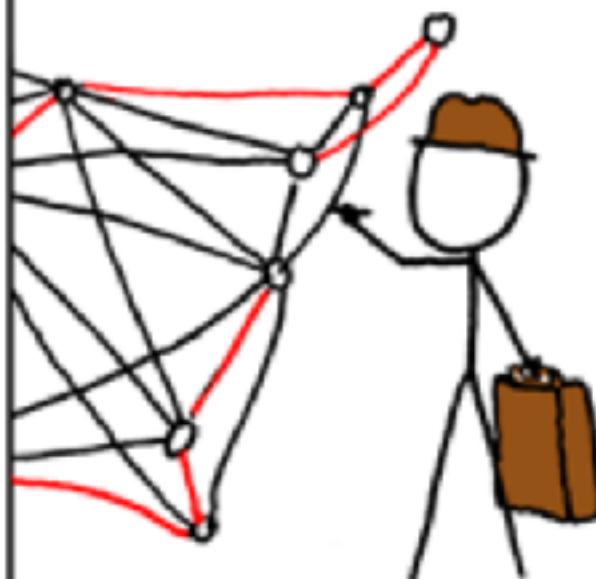
BRUTE-FORCE
SOLUTION:

$$O(n!)$$



DYNAMIC
PROGRAMMING
ALGORITHMS:

$$O(n^2 2^n)$$



SELLING ON EBAY:

$$O(1)$$

STILL WORKING
ON YOUR ROUTE?

SHUT THE
HELL UP.

[<](#)[< PREV](#)[RANDOM](#)[NEXT >](#)[>](#)

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

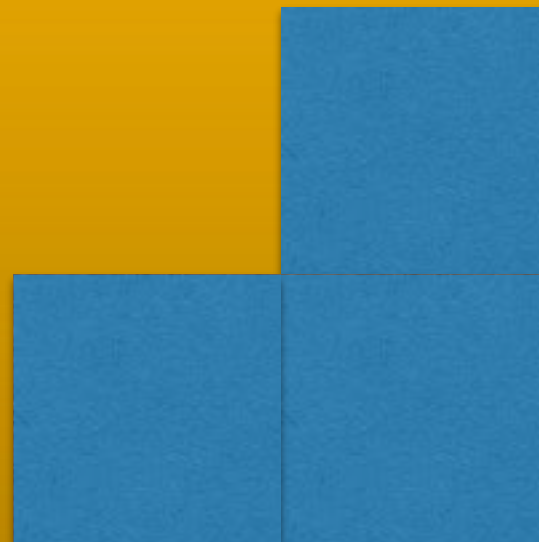
dp(now, visited)

```
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

sekarang yang state dpnya
ga ngecakup state seluruh
problemnya

dikasih grid $R \times C$. 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 :

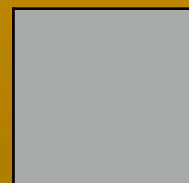
1. sekarang dimana
2. sel kiri diambil/kagak
3. sel atas diambil/
kagak

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 ambil/gak ambilnya kita
dah gak perlu tau

		ambil	ambil	
ambil		NOW		

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

			ambil	
ambil			NOW	

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

			ambil	
ambil		ambil	NOW	

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 K_i dan beberapa
temen.

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

4 orang

$$K(1) = 0$$

$$K(2) = 0$$

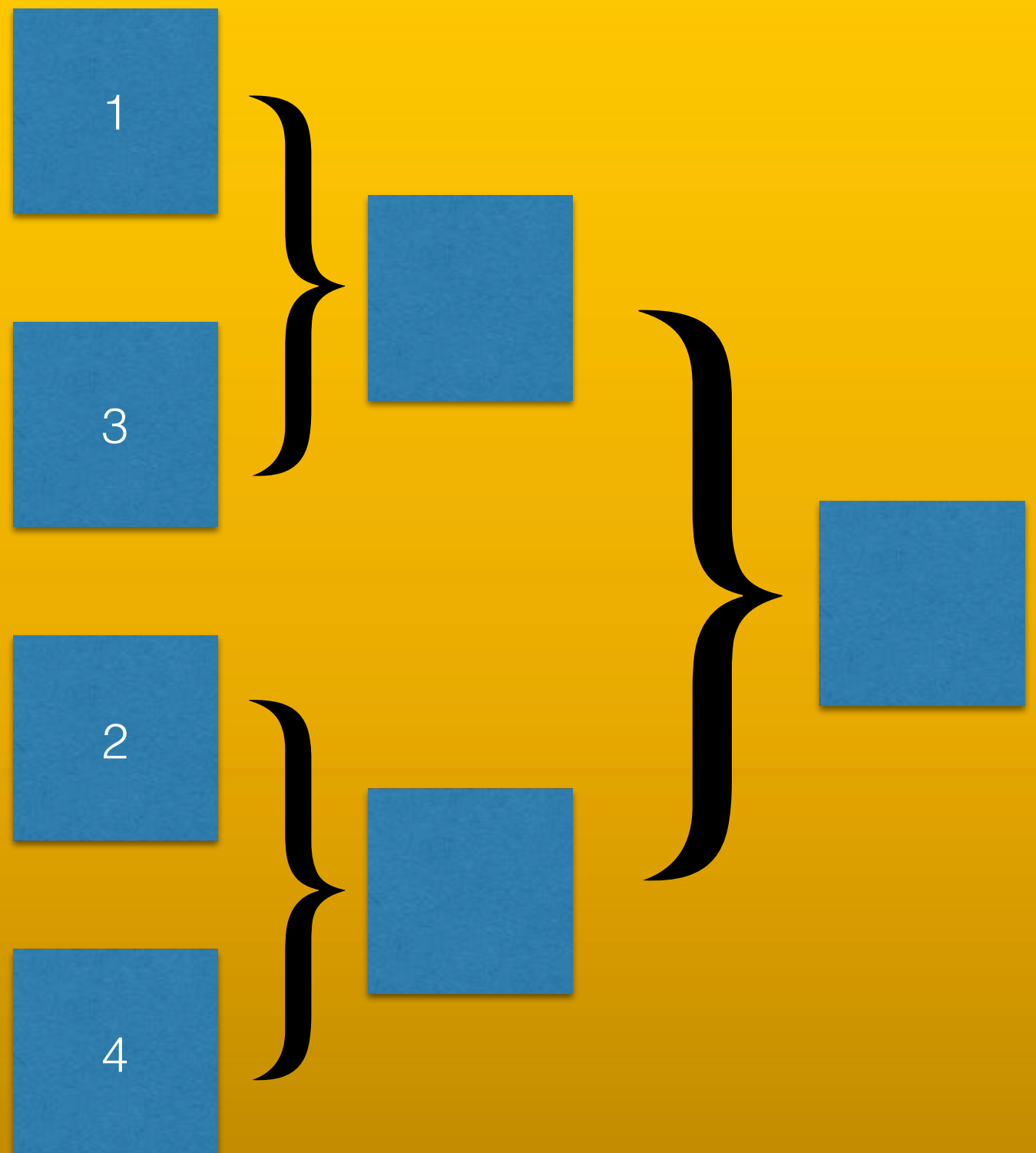
$$K(3) = 1$$

$$K(4) = 0$$

1-2 temen

2-4 temen

3-4 temen



4 orang

$$K(1) = 2$$

$$K(2) = 0$$

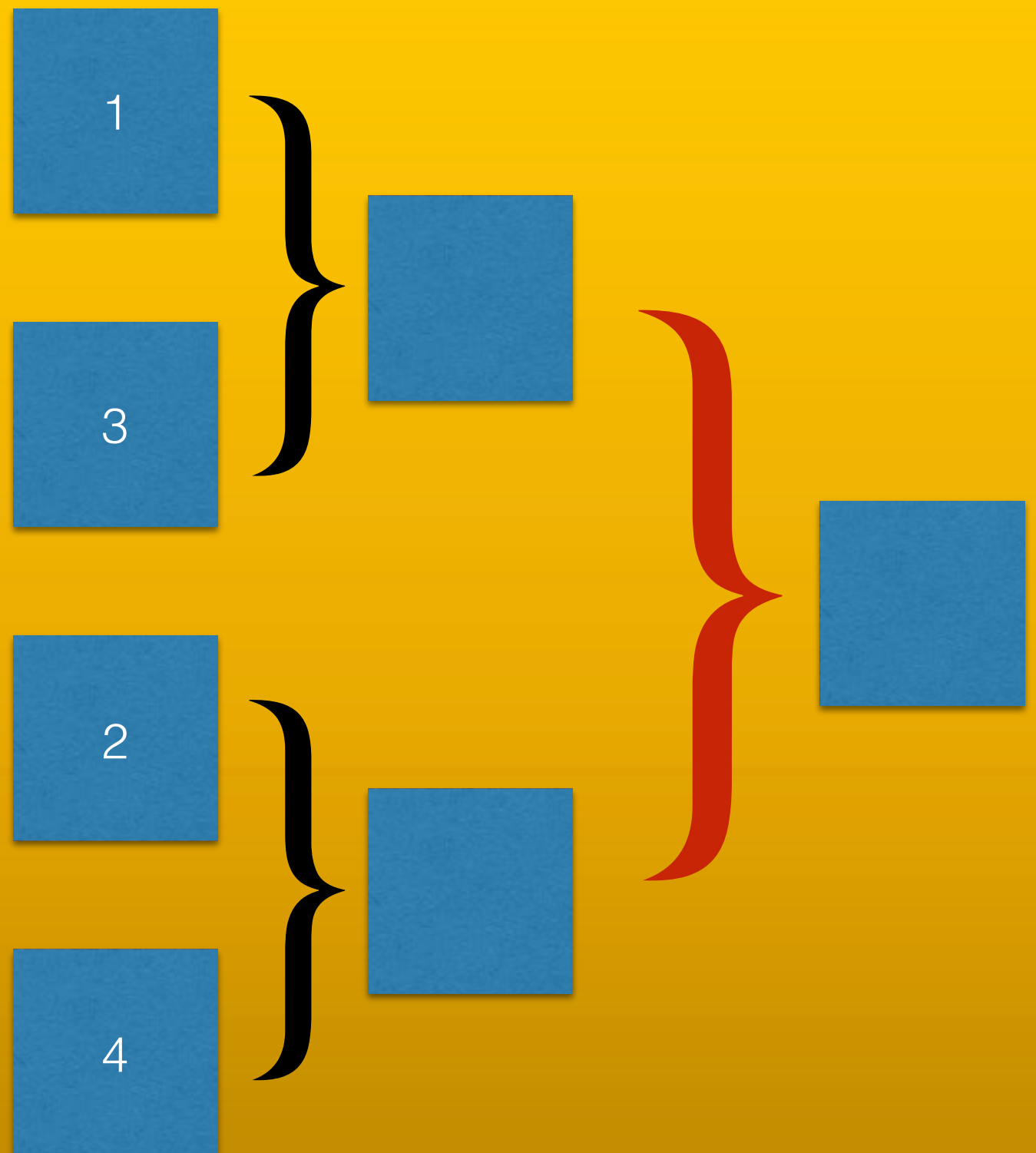
$$K(3) = 1$$

$$K(4) = 0$$

1-2 temen

2-4 temen

3-4 temen



4 orang

$$K(1) = 0$$

$$K(2) = 0$$

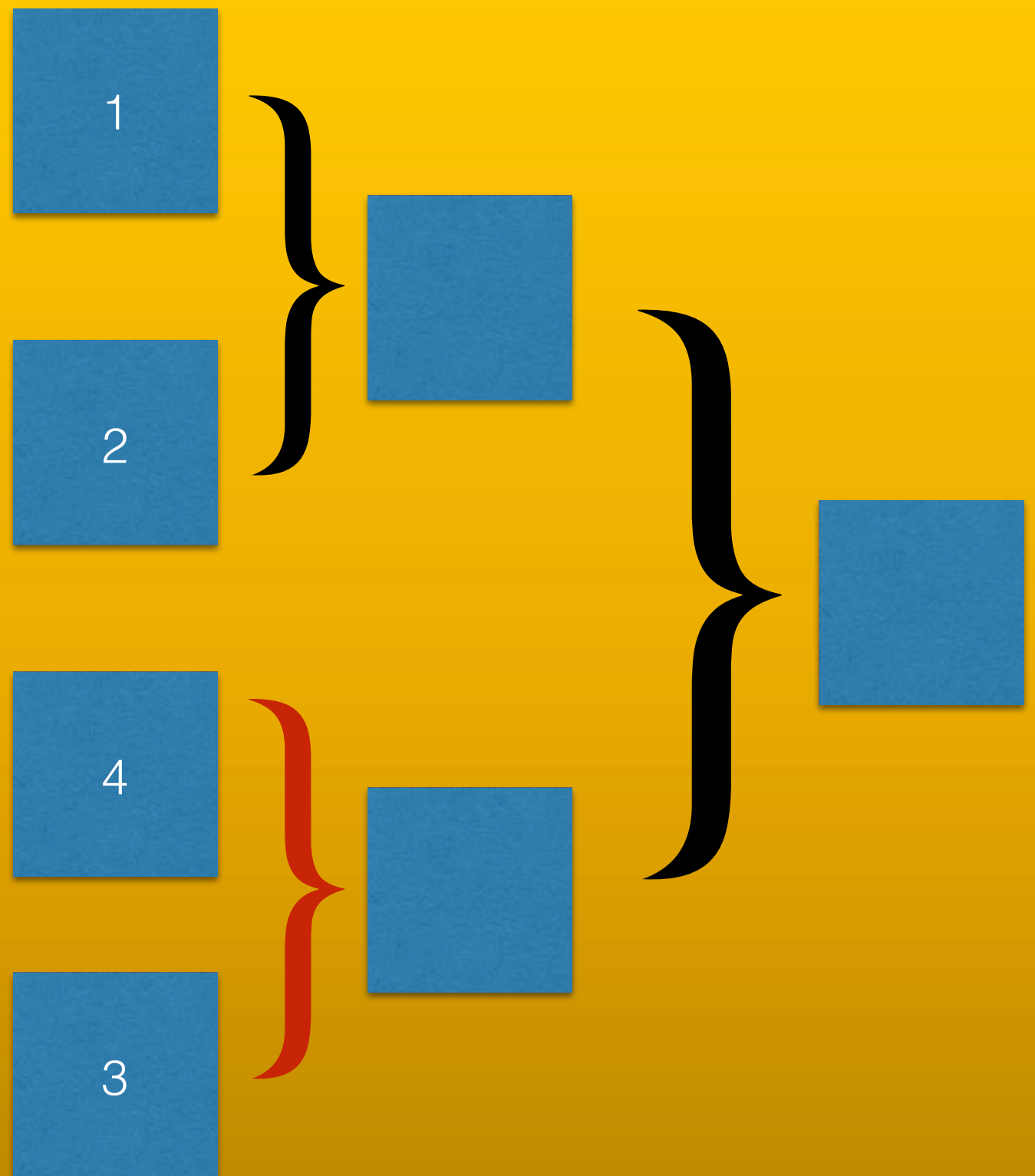
$$K(3) = 1$$

$$K(4) = 0$$

1-2 temen

2-4 temen

3-4 temen



NO SOLUTION

4 orang

$$K(1) = 0$$

$$K(2) = 0$$

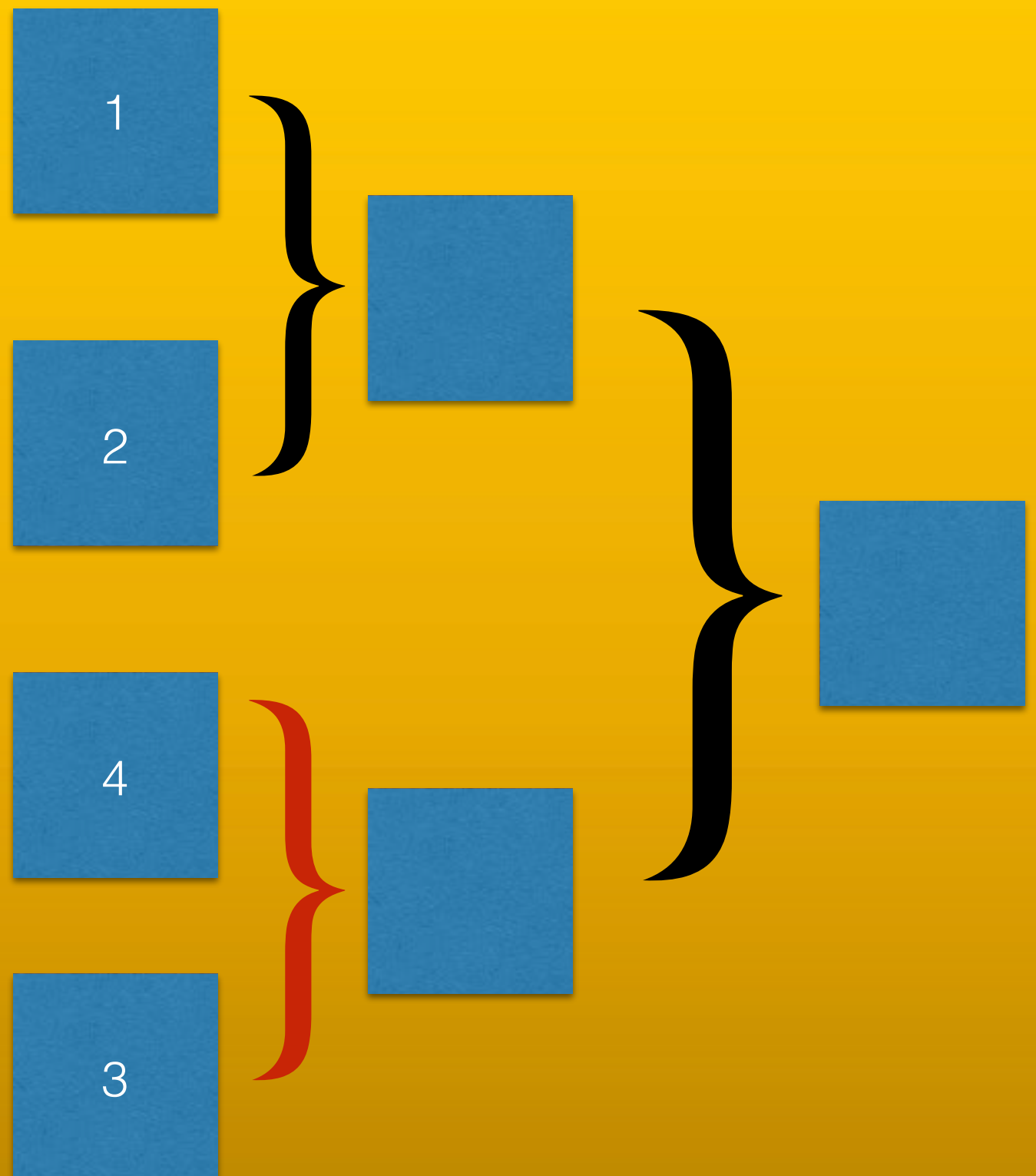
$$K(3) = 0$$

$$K(4) = 2$$

1-4 temen

2-4 temen

3-4 temen



■ ■ ■ ■ ■ ■

state :

orang2 mana aja yang mau dibikin tournament

bisa deduce bisa bikin berapa round. kalo ada
orang-i dan orang-j yang temenan dan

$\text{round} \leq K_i$,

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 $2^{\{\text{banyaknya orang didalam}$
 $\text{state}\}}$

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

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

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

$$(1+2)^n$$

$$(3)^n$$

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

kalo 4^N cuma bisa sampe 13

SPOJ

TWENTYQ

dikasih daftar N item.
tiap item diidentifikasi
dengan M bit

$$1 \leq N \leq 128$$

$$1 \leq M \leq 11$$

nah, sekarang gw pilih satu item diantara N item itu, tapi gw ga kasih tau yang mana

nah, lu boleh nanya K kali
dalam bentuk
bit ke- i dari item ini 0 ato 1

abis tiap kali lu nanya, gw jawab.
jawaban gw boleh lu pake buat mikirin
pertanyaan selanjut2nya

berapa minimum K ?

note : ini bukan interaktif, lu harus
assume worst case buat semua item
yang gw pikirin di awal

contoh nih, misal

$N = 4, M = 11$

00111001100

01001101011

01010000011

01100110001

123456789AB \leftarrow bukan item, buat
index doang

lu bisa cuma nanya 2 kali
nanya bit ke-4

kalo jawabannya 0 : nanya bit ke-7

kalo jawabannya 1 : nanya bit ke-9

■ ■ ■ ■ ■ ■

state :

untuk tiap bit, either kita tau kalo itu 0, tau
kalo itu 1, ato gatau

base case :

kalo dari info yang kita tau,
cuman ada 1 item yang memenuhi :
return 0

gak ada item yang memenuhi : return INF

rekurens :

coba2in mau nanya bit beberapa

dp(mask)

```
if (possible(mask) == 1) return 0;
if (possible(mask) == 0) return INF;
int &ret = dp[mask];
if (ret >= 0) return ret;
ret = INF;
for (int i = 0; i < M; ++i) {
    if (getBit(mask, i) == ?) {
        int mask0 = ganti bit ke-i dari mask jadi 0
        int mask1 = ganti bit ke-i dari mask jadi 1
        int temp = max(dp(mask0), dp(mask1)) + 1;
        ret = min(ret, temp);
    }
}
return ret;
```


state : 3^M

transisi : M

precompute : $3^M * N$

$$3^{11} * 11 = 1.948.617$$

$$3^{11} * 128 = 22.674.816$$

ICPC Jakarta 2014

Maze Mayhem

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

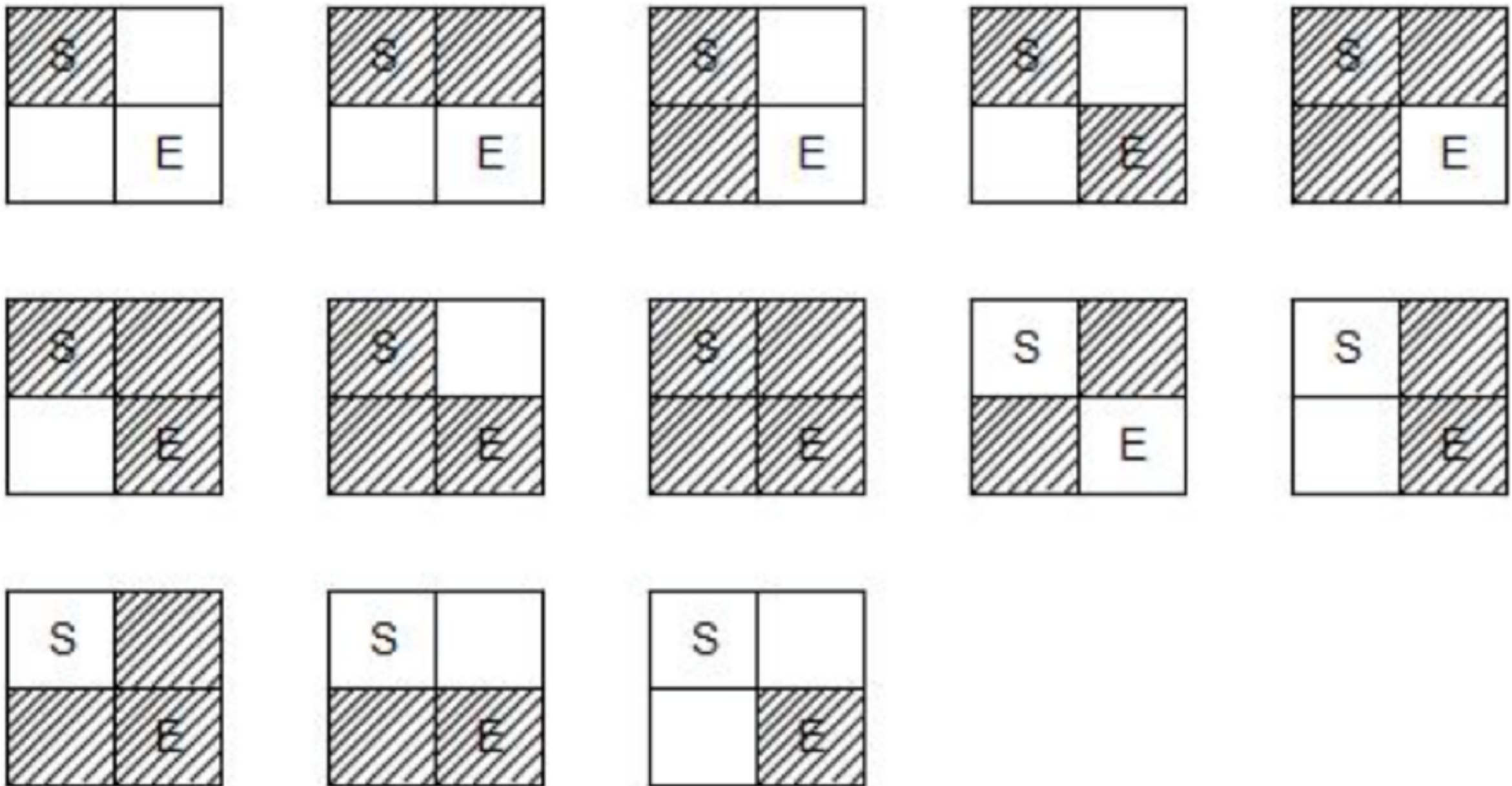
jalannya cuma bisa kanan ato bawah

$$1 \leq R, C \leq 8, R \times C > 1$$

$$0 \leq K \leq R \times C$$

contoh

$$R = 2, C = 2, K = 4$$



■ ■ ■ ■ ■ ■

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

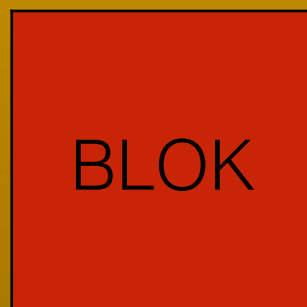
base case :

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

rekurens :

cobain mau/enggga 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		



means unvisitable

state =
 now = (3,3)
 blocked = 11010
 cellblocked = 4

		BLOK	BLOK	
BLOK		NOW		

state =

now = (3,3)

blocked = 11010

cellblocked = 4

			BLOK	
BLOK			NOW	

state =

now = (3,4)

blocked = 10100

cellblocked = 4

			BLOK	
BLOK		BLOK	NOW	

state =

now = (3,4)

blocked = 10101

cellblocked = 5

		BLOK	BLOK	
BLOK	BLOK	NOW		

state =
now = (3,3)
blocked = 11011
cellblocked = 4

		BLOK	BLOK	
BLOK	BLOK	NOW		

state =

now = (3,3)

blocked = 11011

cellblocked = 4

			BLOK	
BLOK	BLOK	BLOK	NOW	

state =

now = (3,4)

blocked = 10111

cellblocked = 4

			BLOK	
BLOK	BLOK	BLOK	NOW	

state =

now = (3,4)

blocked = 10111

cellblocked = 5

terakhir ya, ini
ultimate problem

TOC Maret 2018

Partisi Tabel Perkalian

Diberikan $A[1..N]$ dan $B[1..N]$. Semuanya dibagi 70 sisanya sama

Let $C[i][j] = A[i] * B[j]$. Partisi C semimum mungkin sedemikian sehingga setiap partisi:

1. Either satu sel doang
2. Kumpulan sel segitiga sama kaki siku2 menghadap ke kanan bawah (titik sudut di kiri atas) dengan jumlah seluruh bilangan di dalamnya merupakan bilangan prima.

$$1 \leq N \leq 18$$

$$1 \leq A[i], B[i] \leq 1.818$$

	281	421	491
71	19951	29891	34861
211	59291	88831	103601
1051	295331	442471	516041

■ ■ ■ ■ ■ ■

EOF

Q&A?