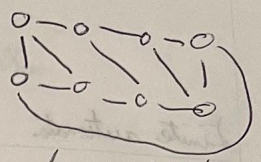
n=8

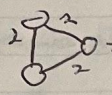


Regular graph

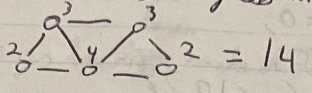
k-regular if every node has degree k

T: for each even number $n > 2$, there exists a 3 regular graph with n nodes

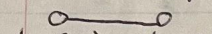
Sum of node degrees - all node degrees in G is even



= 6



= 14



deg(u) +1 deg(v) +1

sum = deg(u) + deg(v) + ...

for each edge $e = (v_i, v_j)$ it makes 1 contribution to both $\deg(v_i)$ & $\deg(v_j)$.

k edges, 2k sum

T: no 3-regular graph with odd # of nodes

proof by contradiction: if G is 3 regular graph w $2n+1$ nodes, n is integer at least 0

sum of degrees is $\text{Sum} = 3+3\ldots = 3(2n+1) = 6n+3 = 2(3n+1)+1$

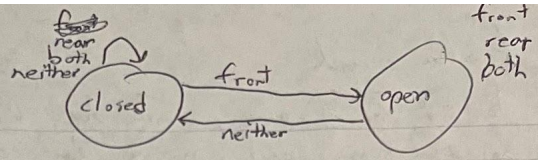
contradiction of previous theorem

induction: prove $P(k)$ is true for natural #s

basis: $P(1)$ is true

induction step: if ~~P(i+1)~~ $P(i)$ is true so is $P(i+1)$

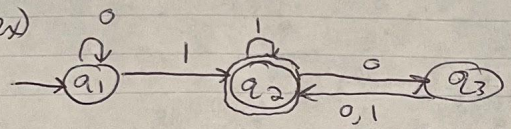
Finite automata




```
int main(){                    //receive memory
    int a[10] = {3,1,2,3,4,9,5,10,92,13,41};
    int sum=0;
    int i=0;
    while (i<10){
        sum= sum+ a[i];
        i=i+1;
    }
    cout<<sum;
    return 0;
}
```

//finite states
//State transition
+memory

ex)

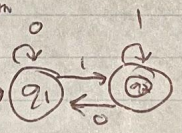


input: 0 1 1 0 1
state: $q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{1} q_2$

if computation ends in double circle, computation is accepted

input: 1 0
state: $q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3$
reject

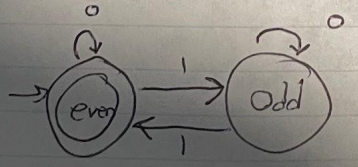
a finite automata has $(Q, \Sigma, \delta, q_0, F)$
Q is ~~a~~ finite set called states
Σ alphabet
$\delta : Q \times E \to Q$ is transition function



Language accepted by automata $L(M) = \{ w \mid$ words in machine $\}$

λ: empty sequence $\quad \phi$: empty set $\quad A$ is set of strings $L(M) = A$

input: 1 0 0 1 1 0 1



ε is empty, consumes nothing

$q_1 \xrightarrow{\varepsilon} q_2$

NFA - nondeter. DFA - deter.

at least 1 branch accepts

becomes deterministic if we go by ~~trigger~~ layer, make it transition

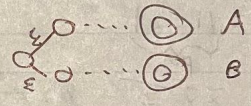
$\underset{\frac{1}{2}}{a_1} \cdots \underset{\frac{1}{2}}{a_n} = 2^n$ number of subsets

$A \circ B = \{000, 010, 100, 110\}$ $A = \{0,1\}$ $B = \{00, 10\}$

$A^* = \{\lambda, \underset{k=0}{0}, \underset{=1}{1}, \underset{=2}{00, 01, 10, 11}, \ldots\}$

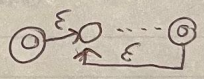
$C^* = \{\lambda, 0, 00, \ldots\}$ if $C = \{0\}$

$A \cup B$: N accepts if one of N_1 or N_2 accepts



$A \circ B$: N accepts if A accepts first part & B accepts second

A^* N accepts if



~~$\Sigma = \{0,1\}$~~

$L = \{1^{2k} 0^{2j+1} \mid k \geq 0, j \geq 0\}$

acc $1111000 \in L$

rej $11110000 \notin L$

Inductive Definition

$\Sigma = \{0,1\}$

- 0 regular expression
- 1 ——
- ε ——
- $\emptyset$ ——

$0 \cup 1$

$0 \circ 1 \to 01$

0^* $\{\lambda, 0, 00, \ldots\}$

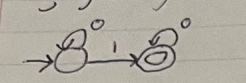
$(0 \cup 1)^*$ $\{\lambda, 0, 1, 00, 01, \ldots\}$

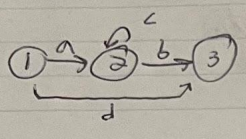
0^*1^*

$0^*, 1, 0^*1, 0^*10^*$ regular expressions

$00001001 \in 0^*1^*$

$0000111001 \notin 0^*1^*$





$ac^*b \cup d$

$(0 \cup 100)^*$

$s = xyz$ $\quad 000\ \frac{11}{y}\ 00$ $\quad xy^0z = 000\ 00$

1^{p^2}

$L = \{1^{n^2} \mid n \geq 0\}$

$n = 2: 1^{n^2} = 1^4 = 1111 \in L$

$1^5 = 11111 \notin L$

$1^{p^2} = \underbrace{11\ldots1}_{p^2}$

$x\ y\ z$

$|xy| \leq p$

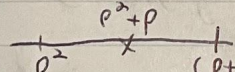
$|xyyz| = |x| + |y| + |y| + |z|$

$(|x| + |y| + |z|) + |y|$

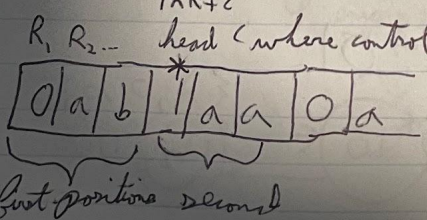
$= p^2 + |y|$

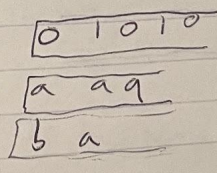
$1^{(p+1)^2} = 1^{p^2+2p+1} \leq 1^{p^2+p}$

$p^2 = |xyz| \leq |xyyz| = p^2 + |y| \leq p^2 + p$

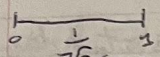


nothing in between 1^{p^2} & $1^{(p+1)^2}$

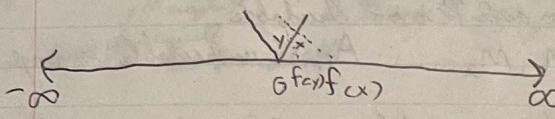




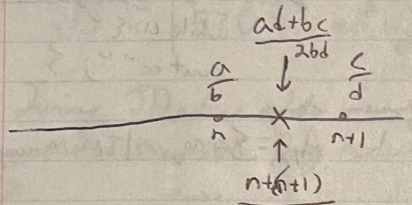
real
$0<x<1$
$(0,1)$ $(0,3)$ $(0,300)$

$\mathbb{R}$ 

$f(x)=3x$ correspondence
$g(x)=300x$



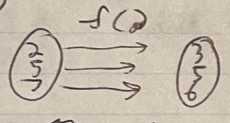
one one ✓
onto ✓
∴ same size



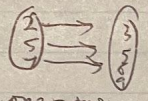
rational $= \frac{a}{b} \neq 0$

$$\frac{\frac{a}{b}+\frac{c}{d}}{2} = \frac{\frac{ad+bc}{bd}}{2} = \frac{ad+bc}{2bd}$$

$\{2,5,7\}$ $\{3,5,6\}$



one-one
onto



one-one
NOT onto

$x = 0.\circ\circ\ldots$
$\neq\neq$
$x \neq f(1) = 0.\textcircled{1}\circ\ldots$
$x \neq f(2) 0.\circ\textcircled{1}\ldots$

$x \in (0,1)$ contradiction
you can find x not in table

Decidability

$\underbrace{1011}_{w}\#\underbrace{1011}_{w} \in L$ $1011\#1100 \notin L$

accept $\langle T, 1011\#1011\rangle \in A_{TM}$

$\langle M, 11001\#10111\rangle \notin ATM$

$\langle M, 1001\#10011\rangle \in ATM$

$\langle M, w\rangle \in A_{TM}$

Turing Decidable

R has to be in the column of M_n, but it have the opposite output diagonally so $M_1(1)$ = accept but we reject, $M_2(2)$ = accept but we reject, so R is different from all M_n, contradiction and R is not decidable.
R is supposed to be one of $M_1, M_2 \ldots$ $\therefore A_{TM}$ is undecidable language

reduction

Logical connection between A_{TM} & other uncomputable problems (programs) ($HALT_{TM}$)

```
while (1) {
cout << " "; }
```

M accepts w after finite steps
M rejects w after finite steps
$M(w)$ = M runs forever

$A_{TM} = \{\langle M, w\rangle \mid TM\ M \text{ accepts } w\}$

R tells third case by running finite steps

If $HALT_{TM}$ can decide then that means A_{TM} is decidable, contradiction

$H(\langle M, w\rangle)$
↑ compiler ↑ input

Computable functions $\Sigma = \{0, 1\}$
$\Sigma^* = \{\lambda, 0, 1, 00, 01 \ldots\}$

$f(n): N \to N$
$f(n) = n+1$

$f(w): \{0,1\}^* \to \{0,1\}^*$
$f(11001) = 00110$

$A = \{w\#w \mid w \in \{0,1\}^*\}$ $1001\#1001 \in A$

$A_{TM} \leq_m HALT_{TM}$
$\langle M, w\rangle$ $\langle M, w\rangle$

$\langle M, w\rangle \in A_{TM}$ accepts w
$\langle M, w\rangle \notin A_{TM}$ rejects w
$\langle M, w\rangle \notin A_{TM}$ runs forever

$\langle M, w\rangle \in HALT_{TM}$ ↗ while (1)
$\langle M, w\rangle \in HALT_{TM}$
$\langle M, w\rangle \notin HALT_{TM}$

$g(\langle M, w\rangle) = \langle M', w\rangle$
#
M

Σ^* set of all strings finite

recognizable does
decidable does not have case of running forever

$x \notin A \Rightarrow x \in \overline{A}$

A

M accepts x
M rejects
M runs forever
$\}\ x \notin A$

$\overline{A}$ turing recognizable

M' accepts x

$\Rightarrow$ $\{0^k 1^k \mid k \geq 0\}$ is not regular, cannot be determined w/ FSA

Turing Machine adds memory with tape, DFA + ▭

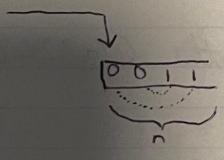
prove undecidable by reduction

$\Rightarrow$ Count number of steps $f(n)$ n = input size — Complexity
size of input

Sort 5, 3, 2 output 2, 3, 5
$n=3$

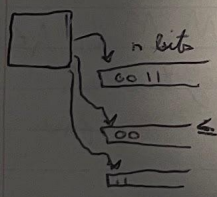
Computability
Complexity

$\frac{n^2}{n^3} \to 0$ $\quad n^2 = O(n^3)$



$A = \{0^k 1^k \mid k \geq 0\}$ in $O(n^2)$ steps

$\frac{n}{2} \times \frac{n}{4} << \underbrace{n + (n-2) + (n-4) \ldots +}_{\frac{n}{4}} \mid \leq n \times n = n^2$



two tapes recognize in $O(n)$ steps
$n + \frac{n}{2} = O(n)$

Simulating one step of 3-tape TM takes $O(n)$ steps, n is # of cells used

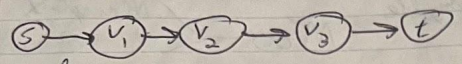
bounded time by polynomial

$P = \bigcup_k TIME(n^k)$ P represents programs solved in polynomial time

steps with TM

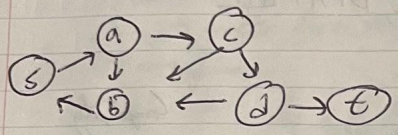
$2^n >> n^{1000}$

# of steps in computation
input size n with variable size



phases

0	1	2	3	4
s	s, v_1	s, v_1, v_2	s, v_1, v_2, v_3	s, v_1, v_2, v_3, t
s	s, a	s, a, b, c	s, a, b, c, d	s, a, b, c, d, t



path length $\leq$ # nodes which $= n$

number edges $\leq n \times n = n^2$

between phase i and phase i+1, check edges n

$n \cdot n^2 = n^3$ polynomial

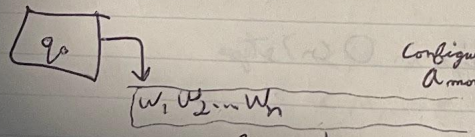
However finding a HAMPATH takes 2^n

SAT $\leq_p$ Clique if $(x_1 \vee x_2) \wedge \bar{x}_1 \wedge \bar{x}_2 \in SAT$ then $(G, 3) \notin$ Clique

2^n to find SAT boolean formula

SAT is NP complete

Proof (configuration)



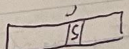
Configuration
a moment of M

length $\leq n^k$
steps

each step has its own configuration

NTM M for $A \in NP$

running time n^k fixed

step i, column j: i 

for every $A \in NP$, $A \leq_p SAT \leq_p 3SAT$

$g(), y \in A \Leftrightarrow g(y) \in SAT \Leftrightarrow f(g(y)) \in 3SAT$

$SAT \leq_p 3SAT$ $\quad 3SAT \subseteq SAT$ $\quad$ at most 3 literals per clause/set

$w: (x_1 \vee x_2 \vee x_3 \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2})$ $\quad w \in SAT$

conjunction is true, first clause is true, new variable to decrease size

$(a_1 \vee a_2 \vee a_3 \vee a_4)$ (T) ⇑ (T)

$(a_1 \vee a_2 \vee z) \wedge (\bar{z} \vee a_3 \vee a_4)$ (T, T, T)

then

$(a_1 \vee a_2 \vee a_3 \vee a_4)$

$V \to z$ $\quad \bar{z} \vee$

$(a_1 \vee a_2)$ (T) ⇑ $\quad (\uparrow \; a_3 \vee a_4)$ (T)

$w \in SAT \Leftrightarrow f(w) \in 3SAT$ $\quad$ what established this relationship?

$f(w): (x_1 \vee x_2 \vee z) \wedge (\bar{z} \vee x_3 \vee x_4) \wedge (\bar{x_1} \vee \bar{x_2})$ $\quad$ true by inserting new literals

instance for 3SAT

SAT $\quad w: (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3 \vee \bar{x_4})$

3SAT $\quad f(w): (x_1 \vee x_2 \vee z) \wedge \underbrace{(\bar{z} \vee x_3 \vee x_4 \vee x_5)}_{(\bar{z} \vee x_3 \vee h) \wedge (\bar{h} \vee x_4 \vee x_5)} \wedge (\bar{x_1} \vee \bar{x_2} \vee y) \wedge (\bar{y} \vee x_3 \vee \bar{x_4})$

3SAT becomes NP complete

$|g(y)| \leq |x|^d$

$|g(y)|^c \leq (|x|^d)^c = |y|^{dc}$

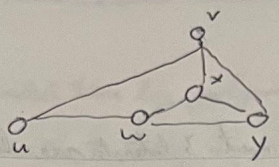
assume an algorithm $R()$ runs in polynomial n^c is time to solve SAT

for any $A \in NP$ let y be instance of A, $y \in A$

compute $g(y)$ in poly. time

$|y|^d$ (fixed), decide $R(g(y))$

$SAT \in P$? $\quad 3SAT \in P$? $\quad P = NP$?

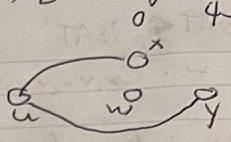


$\langle G, k \rangle \in$ Clique $\iff \langle G', k' \rangle \in$ IS set

$\langle V, E' \rangle$

$(a,b) \in E \rightarrow (a,b) \notin E'$

4-15



G' is the complementary ~~set~~ graph of G

$=$

$(V, \bar{E})$

4 clique, independent set of same size

$n = |V|$ $\quad \frac{n(n-1)}{2} \leq n^2$

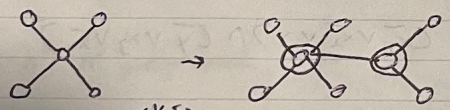
$|E| \leq \binom{n}{2}$

$|E'| \leq \binom{n}{2}$

$\langle G, k \rangle \rightarrow \langle G', k \rangle$ easy, poly. time

$A \leq_p SAT \leq_p$ Clique $\leq_p$ IS $\quad$ IS is NP-complete

Reduction from $IS \leq_p VC$ (vertex cover)



every edge connects 2 nodes

$\langle V, E \rangle$

$=$

convert $\langle G, k \rangle \in IS$ to $\langle G, k' \rangle \in VC$

$k' = n - k$

$n = |V|$ number of nodes in V

$\langle G, k \rangle \rightsquigarrow \langle G', k' \rangle$

assume $S \subseteq V$ is a IS for G

then $V - S$ is a VC for G

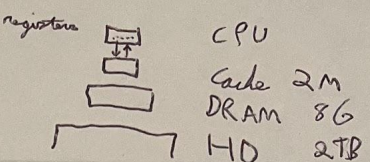
for each edge, $(u, v) \in E$

it is impossible $u \in S$ and $v \in S$, if both are in S then there is an edge

one of u or v is in $V - S$

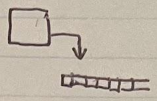
contradiction

So, $V - S$ is a vertex ~~cover~~ cover



Complexity
1. time: # of steps
2. space

April 26, 2023



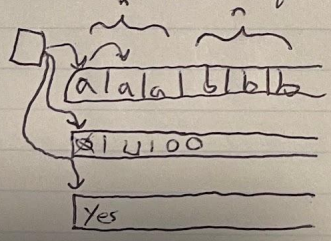
to measure space, count number of cells for computation

```
int main()
    int a[10] = {7,2,5,9,17,4,3,21,15,41};

int sum = 0; #additional variable, 4 bytes
for (int i=0; i<10; i++)  #two integers used for array
   sum = sum + a[i];
cout << sum;
return 0;
```

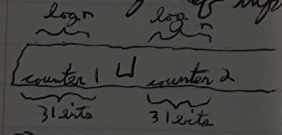
#cells we computed space

$\{a^n b^n \mid n>0\}$ space TM requires?



counter will change, count represented in binary
↓ means different counter is starting

total length of input: $n+n = 2n$



$O(\log n)$ running space

n bits, binary counter

$c_1, c_2 \ldots c_m$

0 0 ... 0 0

0 0 ... 0 1 +1

0 0 ... 1 0 +1

...

1 1 ... 1 1

$2^m \geq n \Rightarrow m \geq \log_2 n$

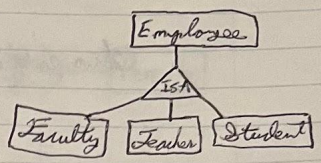
$2 \times 2 \ldots \times 2 = 2^m$ (m)

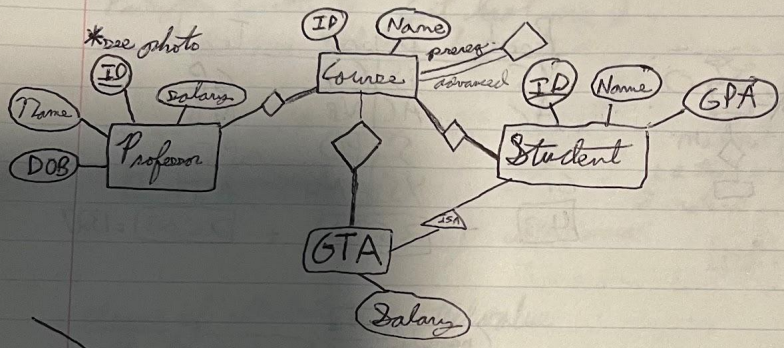
$c_1 \ldots c_n$

0 1 0 1

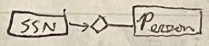
ISA Hierarchies A entity is considered a B entity also
points to parent entity
add descriptive attributes to subclass, identify entities that participate in relationships

person belongs only in one row - overlap
employees have to be hourly or contract - covering





HW1: 5/5 - Given a SSN, it only associates w/ one person. Person has any # of SSN.



assume any table does not have repeating entry

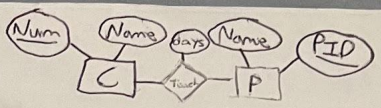
We have multiple unique identifiers - SSN ⇒ candidate key
- (SSN, name) ⇒ super key
subset of attribute is also a key

keys used to design DB → primary key, which is underlined

relation consists of relational schema - list of attribute/column names w/ data type
instance - tuples (rows, records)

schema specifies relation name, name of field and its domain, primary key underlined
- columns, attr,

relations are unordered, tuple is unique, demanded by relation definition



Teach (Num: number, PID: number) => Teach (cnum: number, PID: number, days: number)

number of tuples = cardinality (not row of attribute names)

degree = number of columns

→ bold arrow means one and only one

UniProf (salary, id, days, Name, PID) underline id or P.id

```
CREATE TABLE UniProf(
  salary REAL,
  id     char(10),
  days   INTEGER,
  Name   CHAR(10),
  PID    CHAR(10) UNIQUE,
  PRIMARY  (id)
);
```

* 1) write AS
2) translate to SQL
3) check candidate key
4) participation

→ not bold own(id, PID, days2) underline id or PID

```
CREATE TABLE UniAcct(
    salary REAL,
    id     CHAR(10),
  PRIMARY  KEY (id));
CREATE TABLE
Prof (
  Name   CHAR(10),
  PID    CHAR(10),
  PRIMARY  KEY(PID));   # entity set
```

```
*1) CREATE TABLE own (
2)      id  CHAR(10),
3)     PID  CHAR(10) UNIQUE,
4)    days  INTEGER,
      PRIMARY KEY(id),
5)    FOREIGN KEY(id)
        REFERENCE  UniAcct,
      FOREIGN KEY(PID)
        REFERENCE  Professor);
```

} Attr

reference UniAcct() table

5) participation

one bold arrow

CourseTeach(Name, CRN, PID) Professor(Name, PID)

```
CREATE TABLE CourseTeach(
Name CHAR(10),
CRN  CHAR(10),
PID  CHAR(10),  NOT NULL      # every course needs a prof.
PRIMARY KEY(CRN)
FOREIGN KEY(PID)
  REFERENCE Professor         # candidate key needs to be unique
```

example 4)

CourseTeach(Number, Name, PID)

```
CREATE TABLE CourseTeach (
      Number  CHAR(10),
      Name    CHAR(10),
      PID     CHAR(10), NOT NULL
      PRIMARY KEY(Number, PID)
      FOREIGN KEY(PID)
      REFERENCE Professor
      ON DELETE
        CASCADE );
```

example 5)

Manager(SSN, name, lot) Dept(did, dname, budget) Manages(SSN, Did, since)

```
CREATE TABLE Dept(
did  CHAR(10),
dname CHAR(10),
budget INTEGER
PRIMARY KEY(did))

CREATE TABLE Manager(
   ssn  CHAR(10),
   name  CHAR(10),
   lot   INTEGER,
   PRIMARY KEY(SSN))

CREATE TABLE Manages(
ssn     CHAR(10)
Did     CHAR(10)
since   DATE,
PRIMARY KEY(SSN, Did)
FOREIGN KEY(SSN)
  REFERENCE Manager
FOREIGN KEY(Did)
  REFERENCE Dept );
```

example 6) similar to ex5 but Manager (no more SSN PRIMARY KEY (DID))

NP × State Np.state = State.state

name	Np.state	State.state	capital
YS	WY	WY	Ch
~~YS~~	~~WY~~	~~TX~~	~~Au~~
~~BB~~	~~TX~~	~~WY~~	~~Ch~~
BB	TX	TX	Au

⋈ is easier

S.id of sailors who reserve both red & green boat?

```
(SELECT S.id
 FROM Reserve R, Boat B
 WHERE R.b.id = B.b.id
   AND B.color = 'red' )
INTERSECT
(SELECT Sid FROM Reserve R2, Boat B2 WHERE R2.bid = B2.bid
  AND B2.color = 'green')
```

Table A: sid of sailors who reserve red boat

B: sid of sailors ... green

two different modes

or green, all that changes is INTERSECT to UNION

red boat but never reserve green boat

change INTERSECT to EXCEPT

another solution

```
SELECT Rs.id Rname
FROM Reserve R, Boat B, Sailor S
     Reserve R2, Boat B2, Sailor S2
WHERE
   R.sid = R2.sid AND R.b.id = B.b.id AND R2.bid = B2.bid
   AND B.color = 'red' AND B2.color = 'green' AND
   B.b.id <> R2.b.id) AND S.s.id = R.sid AND S2.s.id = R2.sid
```