Algorithm Class 2.

Asymptotic Notation

△ comparing Running Times
Group running time into classes

Convention: Ignore multiplicative constant
Ignore additive constants > two assumption

(Ex) $5n+7$ vs $5n$ are same
(Ex): n vs $1000n$ are same to

{ + , * } > ignore constant

Definition: Let $f(n)$ and $g(n)$ be running time functions
We say $f(n)$ and $g(n)$ grows at the same rate
and with $f \in \Theta(g)$, $g \in \Theta(g)$
if: $\lim_{n \to \infty} \frac{f(n)}{g(n)}$ = nonzero constant

(Ex): $5n+7 \in \Theta(5n)$
$\because \lim_{n\to\infty} \frac{5n+7}{5n} = \lim_{n\to\infty} 1 + \frac{7}{5n} = 1 \neq 0$

(Ex): $5n \in \Theta(5n+7)$
$\because \lim_{n\to\infty} \frac{5n}{5n+7} = 1 \neq 0$

120

(Ex): $n \in \Theta(100n)$ because $\lim_{n\to\infty} \frac{n}{100n} = \frac{1}{100} \neq 0$

(Ex): $5n^2+6 \in \Theta(n^2)$ because $\lim_{n\to\infty} \frac{5n^2+6}{n^2} = 5 \neq 0$

△ Def → We say $f(n)$ grows faster than $g(n)$ and ~~with~~ write $f \in \omega(g)$ (little-omega)

if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$

(EX): $n^2 \in \omega(n)$

because $\lim_{n\to\infty} \frac{n^2}{n} = \lim_{n\to\infty} n = \infty$

(Ex): $n^2 \in \omega(10^9 n)$

$\lim_{n\to\infty} \frac{n^2}{10^9 n} = \frac{n}{10^9} = \infty$

△ Def We say $f(n)$ grows slower than $g(n)$ and write $f \in o(g)$ (little oh)

if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

(Ex): $n \in o(n^2)$ because $\lim_{n\to\infty} \frac{n}{n^2} = \frac{1}{n} = 0$

$= \Theta$

$> \omega$ ★

$< o$

$\geq$ / $\leq$ → why need? → not sure for results

Def → We say f grows at least as greater as g if $f \in \Theta(g)$ / $f \in \omega(g)$ we write $f \in \Omega(g)$

↓ big omega

Ex: $n^2 \in \Omega(n)$

much better statement is $n^2 \in \omega(n)$

Def → We say f grows at least as slow as g if $f \in \Theta(g)$ / $f \in o(g)$ we write $f \in O(g)$

↓ big oh

$= \Theta$

$> \omega$

$< o$

$\geq \Omega$

$\leq O$

How to find limits? $\rightarrow \lim_{n \to \infty} \frac{f(n)}{g(n)}$

Alternable definition

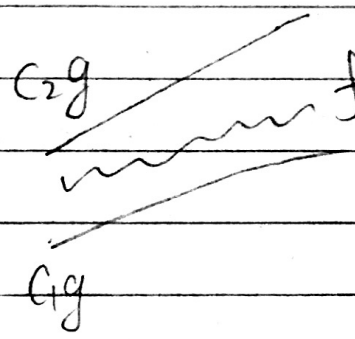
Let f, g be running time function

We say $f \in \Theta(g)$ if there exist positive constants c_1, c_2, c_3 such that:

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \text{for } n \geq c_3$$

may be big or small

Don't care the beginning just focus end



$$\frac{c_1 g(n)}{g(n)} \leq \frac{f(n)}{g(n)} \leq \frac{c_2 g(n)}{g(n)} \quad \text{for } n \geq c_3.$$

$$a \leq \frac{f(n)}{g(n)} \leq c$$

Strang Running time

Nice function families

1) polynomical functions.

$n^0, n^1, n^2, \ldots n^{\text{constant}}$

2) exponention fuactions

$2^n, 3^n, \ldots (\text{constant})^n$

3) Logarithmic function

$\log_1 n, \log_2 n, \log_{\text{const}} n \ldots \log_{\text{const}} n \ldots$

Fact: b^n and $\log_b n$ are inverse of one another

In other words: $b^{(\log_b n)} = n = \log_b(b^n)$

Properties of $\log_b n$

1) $\log_b(xy) = \log_b(x) + \log_b(y)$

2) $\log_b(x^y) = y \log_b x$

3) $\log_b n = \frac{\log_a n}{\log_a b}$ → Prove.

$\log_{10} 16 = \frac{\log_2 16}{\log_2 10}$

$\log_b n = b^{\log_a n}$

$n = b^{\log_b n}$

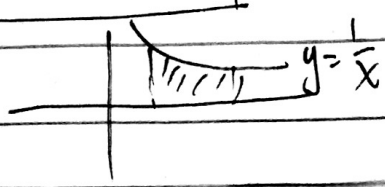
$\log_a n = \log_a (b^{\log_b n})$

$\log_a n = \log_b n \cdot \log_a b$

$\frac{\log_a n}{\log_a b} = \log_b n$

$\log_2 \rightarrow \lg$ Δ

$\log e \leftrightarrow \ln$



Theoem $n^a \in w(n^b)$ for $a > b \geq 0$.

Ex: $n^{2.1} \in w(n^2)$

Prof: $\lim_{n\to\infty} \frac{n^a}{n^b} = \lim_{n\to\infty} n^{a-b>0} = \infty$

Theoem $b^n \in w(a^n)$ for $b > a \geq 1$

Ex: $3^n \in w(2^n)$

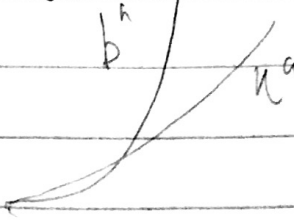
Prof: $\lim_{n\to\infty} \frac{b^n}{a^n} = \lim_{n\to\infty} \left(\frac{b}{a}\right)^n = \infty$ ($\frac{b}{a} > 1$)

Theoem $\log_b n \in \Theta(\log_a n)$ $a, b > 1$

Prof: $\lim_{n\to\infty} \frac{\log_b n}{\log_a n} = \frac{\log_a n}{\log_a b} = \frac{1}{\log_a b} \neq 0$

Theory $b^n \in w(n^a)$ for $b \geq 1$, $a \geq 0$

Ex: $(1.000000001)^n \in w(n^{1000000000})$



$(e^n)' = e^n$

Prof: $\lim_{n\to\infty} \frac{b^n}{n^a} = \frac{\lim (b^n)'}{n\to\infty (n^a)'}$

$\lim_{n\to\infty} \frac{b^n}{n^a} = \frac{((e^{\ln b})^n)' = (e^{n(\ln b)})'}{a n^{a-1}} = (e^{a \ln b} \cdot \ln b)'$

$= b^n \ln b$

$$\frac{(b^n)'}{(a^n)'} = \lim_{n\to\infty} \frac{b^n \ln b}{a n^{a-1}} = \lim_{n\to\infty} \frac{b^n (\ln b)^2}{a(a-1) n^{a-2}}$$

$$\cdots\cdots \lim_{n\to\infty} \frac{b^n (\ln b)^a}{a!} = \infty$$

Theaory

~~Theaory~~ $n^b \in \log_a(n)$ for $b>0$ $a>1$

Prof: $\lim_{n\to\infty} \frac{n^b}{\log_a b} = \lim_{n\to\infty} \frac{b n^{b-1}}{\frac{\ln n}{\ln a}} = \lim_{n\to\infty} \frac{b n^{b-1}}{\frac{1}{n \ln a}}$

$= \lim_{n\to\infty} b \ln a \, n^b \to \infty$

n^n
$n!$
3^n } worst
2^n
$\vdots$
n^3
n^2
n

$\log(n!) \in \Theta(n \log n)$

$\log_2 n$ $\log_3 n$ → best rate

Theaory $n! \in \omega(b^n)$ $b>1$

Prof: $\lim_{n\to\infty} \frac{n!}{b^n} = \frac{n \times (n-1) \times \cdots \times 2 \times 1}{b \times b \times \cdots \times b \times b} = \infty$

Theaory $n^n \in n!$

~~Ex~~ $\log(n!) \in \Theta(n \lg n)$

$$\lim_{n \to \infty} \frac{n \lg n}{n} = \infty$$

$$\lim_{n \to \infty} \frac{n \lg n}{n^2} = \frac{\lg n}{n} = 0$$

Theory $\lg(n!) \in \Theta(n \lg n)$

Prof : $n! \leq n^n \rightarrow \left(\frac{n}{2}\right)^{\frac{n}{2}} \leq n! \leq n^n$

$n! = 1 \times 2 \cdots \frac{n}{2} \cdots n$

$\left(\frac{n}{2}\right)^{\frac{n}{2}} = \frac{n}{2} \times \cdots \frac{n}{2}$

$$\frac{n}{2} \log\left(\frac{n}{2}\right) \leq \log(n!) \leq n \lg n$$

Theory **Wrong** : $\lim_{n \to \infty} \frac{f}{g} = \lim_{n \to \infty} \frac{\ln f}{\ln g}$ ✗

Prof : $f(n) = n^n$
$g(n) = n!$ $\Rightarrow f \in \omega(g)$

but : $\lg(f) \in \Theta(\lg g)$

9.29 recursive

```
fibonacci (unsigned n)
    if (n==0)
        return 1;
    return f(n-1)*n;
```

$M(n)$ = # of $f(n)$ performs *

1) code → Recurrence

2) solve recurrence

$M(0) = 0$, no more operation → base case

$M(n) = 1 + M(n-1)$, $n>0$ } recurrence for $M(n)$

$M(0) = 0$

$M(1) = 1 + M(0) = 1+0 = 1$

$M(2) = 1 + M(1) = 1+1 = 2$

$M(3) = 1 + M(2) = 1+2 = 3$

$M(n) = n$, $n>0$ ---> guess isn't enough

Verify:

1) $M(0) = 0$ ✓

2) $M(n) = 1 + M(n-1)$ $n>0$

$n = 1 + n-1$

⇓

$n = n$ ✓ $n>0$

数学归纳法

2^{n+2} 3^{n+1}

$2n^{n^2-2}$

```
f(unsigned n)
{
    if(n==0)
        return 1+1+1;
    return f(n-1)+f'(n-1);
```

A(n) # of +'s performed by f(n)

$A(0) = 2$

$A(n) = 1 + A(n-1) + A(n-1),\ n>0$

$= 1 + 2A(n-1),\ n>0$

$A(0) = 2$

$A(1) = 1 + 2A(0)$

$= 1 + 2*2 = 5$

$A(2) = 1 + 2A(1) = 1 + 2*5 = 11$

$A(3) = 1 + 2A(2) = 1 + 2*11 = 23$

$A(4) = 1 + 2A(3) = 1 + 2*23 = 47$

0 ~ 2
1 ~ 5
2 ~ 11
3 ~ 23
4 ~ 47

$3 \cdot 2^n - 1$ → verify:

$A(n) = 3 \cdot 2^n - 1,\ n \geq 0$

$A(0) = 3 \cdot 2^0 - 1 = 2$

$A(n) = 1 + 2A(n-1),\ n>0$

$3 \cdot 2^n - 1 = 1 + 2(3 \cdot 2^{n-1} - 1)$

$= 1 + 3 \cdot 2^n - 2 = 3 \cdot 2^n - 1$ ⇒ verified

3^{n-1} 2^1

0 0 1 3 5 8 ...

$\log_2 n - 1$

4 1 3 5 | 8 7 6 2

1 3 4 5 2 6 7 8

left mod mid+1 right

sorted sorted

$M(n)$ = # "$A[j] < A[i]$" performed by mergesort on arrays of

by MERGE on array of

$$\sum_{k=0}^{n-1} (1) = n-1 - 0+1 = n-1$$

$$\sum_{k=0}^{n-x} (1) = n-x - 0+1$$

$M(0) = 0$

$M(1) = 0$

$M(n) = M(\lfloor \frac{n}{2} \rfloor) + M(\lceil \frac{n}{2} \rceil) + n-1, \quad n>1$

$M(0) = 0$

$M(1) = 0$

$M(2) = 1$

$M(3) = M(1) + M(2) + 2 = 3$

$M(4) = 1 + 1 + 3 = 5$

$M(5) = M(2) + M(3) + 4 = 8$

$M(6) = M(3) + M(3) + 5 = 11$

$2^1 - n - 1 \quad 2^n - 2n$

No eviden Lric
suggest that

$M(n) \in \Theta(n \lg n)$

We want: exact formula

Simplified Recursive
When $n = 2^m$

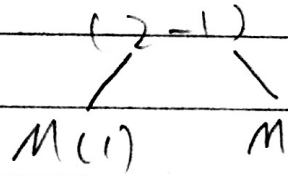
$M(1) = 0$

$M(n) = M(\frac{n}{2}) + M(\frac{n}{2}) + (n-1)$ when $n = 2, 4, 8, 16$
$= 2^m$
$= 2M(\frac{n}{2}) + (n-1)$

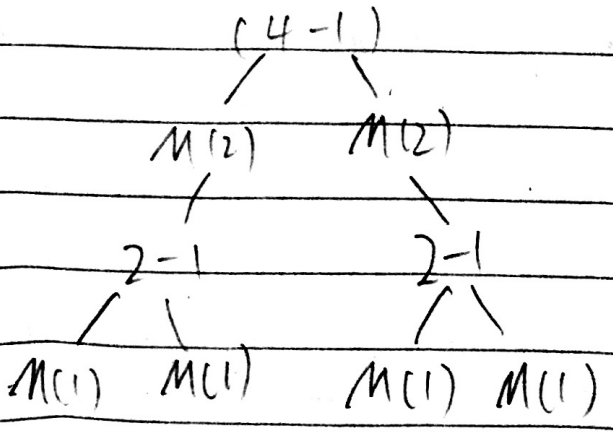
$M(1) = 0$

$M(2^m) = 2M(2^{m-1}) + (2^m - 1), m > 0$

$M(2) = M(1) + M(1) + 2 - 1 =$



$M(4) = M(2) + M(2) + 4 - 1$



→ tree method

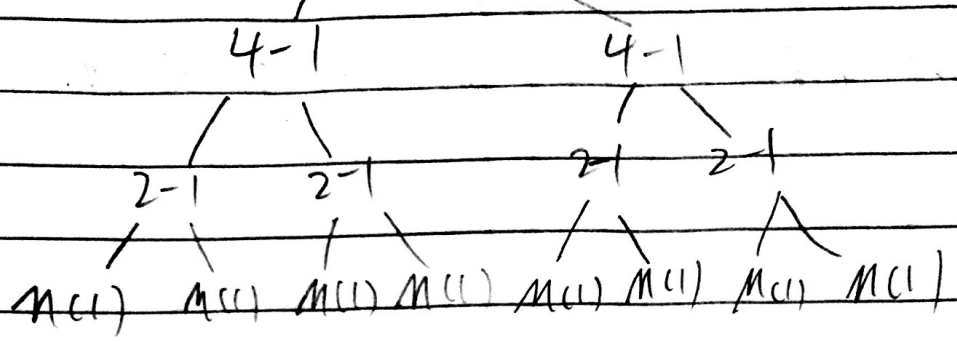
2 1
3 2 4-1

$\log_2 n$ $n \log n$

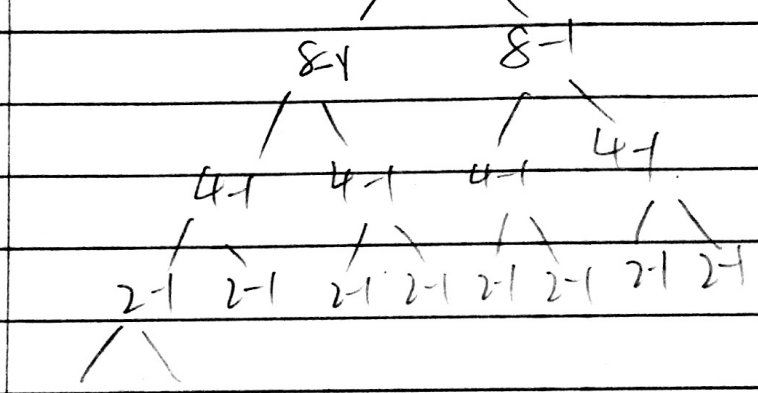
3 1×2 $(n \times n-1) - 2$ $(n-1)! + 1$

2	1	
3	3	2x2 -1
4	5	2x3 -1
5	8	3x3 -1
6	11	3x4 -1

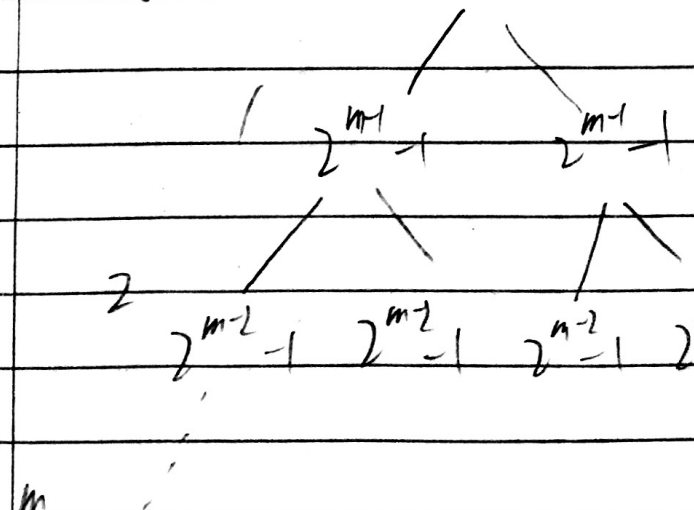
$M(8) = 8-1$



$M(16) = 16-1$



$M(2^m) = 0 \quad 2^m - 1$



m+1 levels

2^L node at level $0 \le L \le m$

m

$M(1)\ 2^{m-m}$

or

Each node at level l
has value $2^{m-l} - 1$ $0 \leq l < m$
Each node at level m has value $(M(i))$

10.1

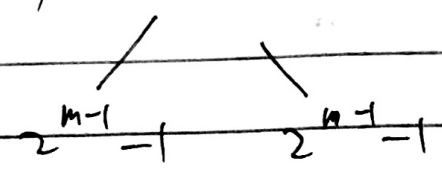
$M(1) = 0$

$M(n) = M(\lfloor \frac{n}{2} \rfloor) + M(\lceil \frac{n}{2} \rceil) + n-1, \quad n > 1$

$= 2M(\frac{n}{2}) + n - 1$ ∴ $n \lg n$

$M(1) = 0$

$M(2^m) = 2^{m-1}$



$M(1)$

1) # levels = $m+1$
(labeled $0, 1, 2, 3, \cdots m$)
2) level i has 2^i nodes
3) Each node on level (m) has node $M(1) = 0$
4) Each node on level $(m-i)$ has value $2^{m-i} - 1$

每一层上的 node values 和 = $2^i (2^{m-i} - 1)$

↑ 个数 ↑ value

$$M(2^m) = 0 + \sum_{i=0}^{m-1} (2^i)(2^{m-i} - 1)$$

$$= \sum_{i=0}^{m-1} 2^i (2^{m-i} - 1) = \sum_{i=0}^{m-1} (2^m - 2^i)$$

$$= \sum_{i=0}^{m-1} (2^m) - \sum_{i=0}^{m-1} (2^i)$$

$$= 2^m \sum_{i=0}^{m-1} (1) - \sum_{i=0}^{m-1} (2^i)$$

$$= m 2^m - (2^m - 1)$$

$$M(2^m) = (m-1)2^m + 1$$

$$\sum_{i=0}^{m-1} (2^i) = 2^0 + 2^1 + 2^2 + \cdots + 2^{m-1}$$

$$= 2^m - 1$$

△ Verify

$M(1) = M(2^0) = (0-1)2^0 + 1 = -1 + 1 = 0$ ✓

$M(2^m) = 2M(2^{m-1}) + (2^m - 1)$

$(m-1)2^m + 1 \stackrel{?}{=} 2\left(((m-1)-1)2^{m-1} + 1\right) + (2^m - 1)$

$= (m-2)2^m + 2 + 2^m - 1$

$= (m-1)2^m + 1$

$M(2^m) = (m-1)2^m + 1$

$n = 2^m$

$m = \lg n$

$M(n) = (\lg n - 1)n + 1 = n\lg n - n + 1$

$(n = 1, 2, 4, 8 \cdots\cdots)$

△ In general

$M(n) = n\lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1, \quad n \geq 1$

$M(3) = M(1) + M(2) + 2 = 3$

$M(5) = M(2) + M(3) = 8$

$M(1) = 0$

$M(3) = 3$

$M(5) = 8$

$M(7) = 7 \cdot \lg 7 - 2^{\lg 7} + 1 = 7 \cdot 4 - 2^4 + 1 = 13$

Master Theasom ☆ recursive

Let $M(1)$ = constant $= c$

$$M(n) = a\,M\left(\frac{n}{b}\right) + \theta(n^d)$$

for $n = b, b^2, b^3, \ldots$

a, b, c, d are constants

$c > 0,\ b \geq 2,\ a \geq 1,\ d \geq 0$

for example: $M(n) = 2M\left(\frac{n}{2}\right) + n-1,\ n \geq 2, 4, 8, \ldots$

$\in \theta(n)$

$$\begin{cases} c = 0 \\ a = 2 \\ b = 2 \\ d = 1 \end{cases}$$

☆ → Then Case 1

Find the $\theta(n)$ Quickly

$a < b^d$	$M(n) \in \theta(n^d)$
$a = b^d$	$M(n) \in \theta(n^d \lg n)$
$a > b^d$	$M(n) \in \theta(n^{\log_b a})$

have to use exactly base!

$M(n) = 2M\left(\frac{n}{2}\right) + \theta(n^1)$

$b = 2$, $a = 2$, $d = 1$

a vs b^d

$\Downarrow$

$a = b^d$

$\Downarrow$

$n \lg n$

example: $M(n) = 4M(\frac{n}{2}) + \theta(n^5)$

$a = 4$

$b = 2 \quad 4 < 2^5 \quad M(n) \in \theta(n^5)$

$d = 5$

binary search

```
bs (A, left, right, x)

    if (left == right)
        return comp(A[left] == x)
    mid = (left + right) / 2
    case comp (A[mid], x)
        0 : return true;
        + : retun bs (A, left, mid-1, x)
        - : return bs (A, mid+1, right, x)
```

$C(n)$ = # of comp()

$C(1) = 1$

$C(n) = 1 + C(\lfloor \frac{n}{2} \rfloor), \quad n \geq 2$

when n is pow of 2

$C(1) = 1$

$C(n) =$

† n bit #

1 1 0

1 1 1

乘法	$\Rightarrow O(n^2)$
十进制	

10.6

```
f(A[], left, right)
{
    if (left == right)
        return 1*1
    mid = (left + right) / 2.
    a1 = f(A, left, mid);
    a2 = f(A, left, mid);
    a3 = f(A, mid+1, right);
    a4 = f(A, mid+1, right);
```

① ② ③

```
n = righ - left + 1;
for (i=1; i<=√n; i++)
    prod = prod * i;
return a1*a2*a3*a4
    *prod;
```

$M(n)$ = worst-case # of *'s performed by f on array of n = right − left + 1

$M(1) = 1$

$$M(n) = \underbrace{2M\left(\lfloor \tfrac{n}{2} \rfloor\right) + 2M\left(\lceil \tfrac{n}{2} \rceil\right)}_{①} + \underbrace{\sum_{i=1}^{\lfloor \sqrt{n} \rfloor} (1)}_{②} + \underbrace{4}_{③}, n>1$$

When $n = 2^m$ the recurrence become

$M(1) = 1 \quad M(2^m) = 4 \cdot (2^{m-1}) + 2^{m/2} + 4$

In terms of n

$M(1) = 4$

$M(n) = 4M(\frac{n}{2}) + \sqrt{n} + 4$, n in a power of 2.

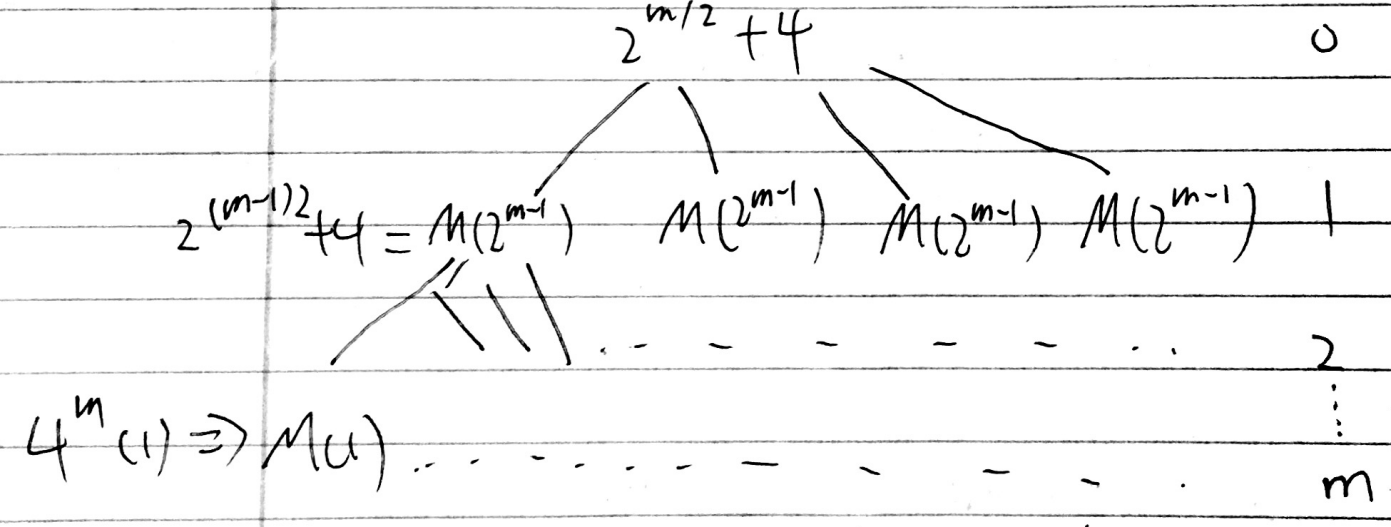
Applying MT, we have

$a = 4, b = 2, d = \frac{1}{2}$

$\Rightarrow M(n) \in \Theta(n^{(\log_2 4)}) = \Theta(n^2)$

Trace method when $n = 2^m$

$M(2^m) = 4M(2^{m-1}) + 2^{m/2} + 4$



There are $m+1$ levels in this tree

label of $0, 1, 2, \cdots m$

# of nodes at level l? 4^l

value of each node at last level (m): $M(1) = 1$

Level $< m$: $2^{(m-l)2} + 4$

$\Rightarrow$ Pf: $\sum_{l=0}^{m-1} 4^l \left(2^{(m-l)2} + 4\right)$

$$\therefore M(2^m) = 4^m c(1) + \sum_{i=0}^{m-1} 4^i \left(2^{(m-i)/2} + 4\right)$$

$$= \sum_{i=0}^{m-1} 4^i \left(2^{(m-i)/2}\right) + \sum_{i=0}^{m-1} 4^i (4)$$

Geometric Sum formula: $4\left(\frac{4^n - 1}{4-1}\right) = \frac{4}{3}(4^n - 1)$

$$\sum_{i=0}^{n} b^i = \begin{cases} \frac{b^{n+1}-1}{b-1} \\ \text{for } b \neq 1 \\ n+1 \text{ of } b=1 \end{cases}$$

$$S = b^0 + b^1 + b^2 + \cdots + b^n$$

$$b \cdot S = b^1 + b^2 + \cdots + b^{n+1}$$

$$S - bS = b^0 - b^{n+1}$$

$$S(1-b) = 1 - b^{n+1}$$

$$S = \frac{1-b^{n+1}}{1-b} = \frac{b^{n+1}-1}{b-1}$$

$$\sum_{i=0}^{n-1} 4^i \left(2^{(n-i)/2}\right) = 4^i \left(\sqrt{2}\right)^{n-i}$$

$$= 4^i \frac{\left(\sqrt{2}\right)^n}{\sqrt{2}^i} = (\sqrt{2})^n \sum_{i=0}^{n-1} \left(\frac{4}{\sqrt{2}}\right)^i$$

$$= (\sqrt{2})^n \sum_{i=0}^{n-1} \left(\frac{4}{\sqrt{2}}\right)^i$$

$$= (\sqrt{2})^n \left(\frac{\left(\frac{4}{\sqrt{2}}\right)^n - 1}{\frac{4}{\sqrt{2}} - 1}\right)$$

$n = m$

$$M(2^m) = (\sqrt{2})^m \frac{\left(\frac{4}{\sqrt{2}}\right)^m - 1}{\frac{4}{\sqrt{2}} - 1} + \frac{4}{3}(4^m - 1) + 4^m$$

$n = 2^m$

$(\sqrt{2})^m = 2^{m/2} = (n)^{\frac{1}{2}} = \sqrt{n}$

$4^m = n^2$

$$M(2^m) = M(n) = \sqrt{n} \frac{\left(\frac{n^2}{\sqrt{n}} - 1\right)}{\left(\frac{4}{\sqrt{2}} - 1\right)} + \frac{4}{3}(n^2 - 1) + n^2$$

$$= \left(\frac{4}{\sqrt{2}} - 1\right)\left(n^2 - \sqrt{n}\right) + \frac{7}{3}n^2 - \frac{4}{3}$$

$$= \Theta(n^2)$$

Problem:

☆ Write an algorithm to read an integer n and compute $\lfloor \sqrt{n} \rfloor$

$$\lfloor \sqrt{n} \rfloor^2 \le n < (\lfloor \sqrt{n} \rfloor + 1)^2$$

```
int sqrt(n)
    for (i=0; i<=n; ++i)
        if (i*i<=n && (i+1)*(i+1)>n)
            return i;
```

try the brute-force strategy { easy to try; good enough.

☆ 2) Find smallest in an array A[left+...right]

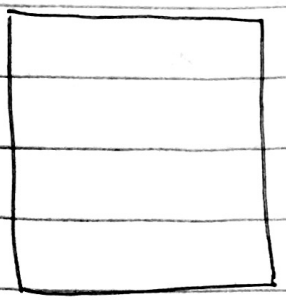
☆ 3) Closet pair (P[1....n]) ⟶ ans = ∞

```
for (i=1; i<n; ++i)
    for (j=i+1; j<=n; ++j)
        if (ans > dist(P[i], P[j]))
            ans = dist(P[i], P[j])
return ans;
```

running time

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (1) = \Theta(n^2)$$

Web Page



n letters in the "text"
m letters in the "pattern"

n-m+1

```
text_search (text[1...n], pat[1...m])
{ for (s=1; s≤n-m+1; ++s)
      for (p=1; p≤m, ++p)
          if pat[p] != text[s+p-1]
              break;
      if (p>m) // no match
          return true;
}                                   Θ(n·m)
  return false
}
```

☆ order A[left .... right]
sort (A[left+...right]
{
for each rearrangement π of A[]
for (i = left, i<right; ++i)
if A[π[i]] > A[π[i+1]]
break;
if (i >= right)
output π;

☆ Partition

$c_1, c_2, c_3 \cdots c_n$

$A = \{c_1, c_5, c_7, c_8\}$ Brute-Force

$B = \{c_2, c_3, c_4, c_6\}$

$\underbrace{\_ \_ \_ \_ \_ \_ \_ \_}_{2\ 2\ 2\ 2\ 2\ 2\ 2\ 2} \quad 2^8$

☆ Traveling salesman

10.8

Reduce problem
into subproblem(s) of smaller

Greedy Method

Ex: sorthing
3 1 2

Remove the larger
put it at the right most position

3 1 2
→ [2 1] (3) pick the larger

Sort remaining list

```
SELECT [Aleft .. right]
{
  if (left < right)
  {
    swap( A[right], A[largest]);
    SELECT [Aleft ..... right-1]; }
  }
```

swap(A[right], A[largest]) : (n-1)
SELECT [Aleft ..... right-1]; : S(n-1)

$S(1) = 0$

$S(n) = S(n-1) + (n-1), n \geq 2$

$S(1) = 0$

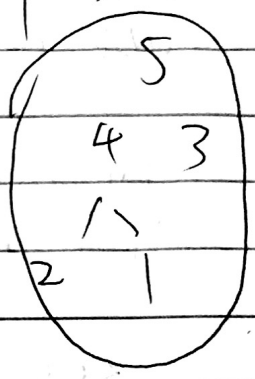
$S(2) = S(1) + 1 = 0 + 1 = 1$

$S(3) = S(2) + 2 = 1 + 2 = 3$

$S(4) = S(3) + 3 = 3 + 3 = 6$

$S(5) = S(4) + 4 = 6 + 4 = 10$

heap sort



Guess:

$S(n) = \frac{n(n-1)}{2}, n \geq 1$

$S(1) = \frac{1(1-1)}{2} = 0$

$S(n) = S(n-1) + (n-1)$

$\frac{n(n-1)}{2} = \frac{(n-1)(n-2)}{2} + n-1$

```
min ( A[left .... right])
{
  if (left == right)
      return A[right];
      temp = min (A[left ... right-1]);
      return ( temp < A[right] ? temp : A[right]);
}
```

$M(1) = 0$

$M(n) = M(n-1) + 1, n \geq 2$

$M(1) = 0$

$M(2) = 1 + M(1) = 1 + 0 = 1$

$M(3) = 1 + M(2) = 1 + 1 = 2$

$M(n) = (n-1)$

$n \geq 1$

COIN CHANGE

1 Q 25
1 D 10
1 N 5
1 P 1
41

Input integer, $n \geq 0$

output smallest number of pennys, nickels, quarter | required to make change for n cent

300

259

41

[25] [41-25]

16 [10] [16-10]

[5] [6-5]

[1]

```
COIN_CHANGE(n)
{
    if (n == 0)
      return 0,
    return 1 + COIN_change (n - max_den (n, [25, 10, 5, 1]))
      (n/25) + (n%25/10)
}
```

hardly to prove

~~SELECT~~

Prove of COIN_CHANGE

Claim: There is an optimal solution to COIN_CHANGE(n) that contains a coin of the largest possible

Proof: by case

$0 \le n < 5$, ① 1 11, the optimal solution is n pennies
So it does contain a coin of the largest (which is 1) as claimed

$5 \le n < 10$: any optimal solution contains only pennies and nickels.
If an optimal solution does not contain a nickel (largest poss den)
then it contain pennies. But being optimal
it can't contain more than 4 pennies
So $n \le 4$, contradiction $n \ge 5$

$10 \le n < 25$: any optimal solution contains only pennies, nickle, ~~dime~~ (largest possible)
Suppose it does not contain dime, so it can have at most 4 pennies,
So $n \le 5+4 = 9$, contradiction $n \ge 10$

$n \ge 25$: 4 pennies $4+20$
2 dime + 0 nickels
1 dime + 1 nickels $n \le 4+10+5 \le 24$
0 dime + 1 nickels $4+5$

Claim: The greedy strategy (always pick the largest possible denomination doesn't work when only quarters, dimes, and pennies are available)

$n = 31$

$25 + 1 + 1 + 1 + 1 + 1 + 1$

Greedy point: 7 coins

Best: ⑤ + ⑫ + ⑭ + ⑪

quarter: 25
dime: 10
copper: 7
nickel: 5
penny: 1

greedy: $10 + 1 + 1 + 1 + 1$
best: $7 + 7$

Movies.

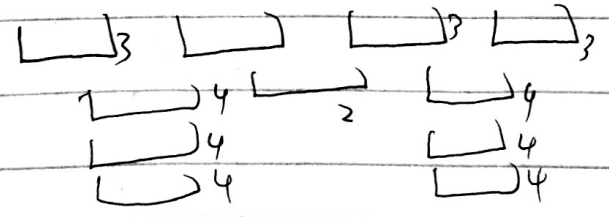
Input: show time (start, finish)
$(s_1, f_1), (s_2, f_2), (s_n, f_n)$

Output: the maximum number of movies that can be watched

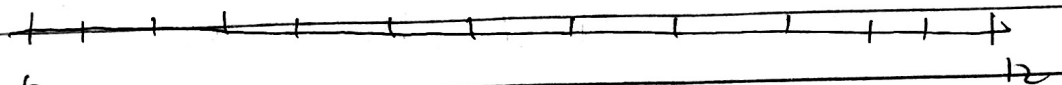
① shortest:

② earliest:

③ least overlapping:



(1, 4).
(3, 5)
(0, 6)
(5, 7)
(3, 8)
(5, 9)
(6, 10)
(8, 11)
(8, 12)
(2, 13)
(13, 14)



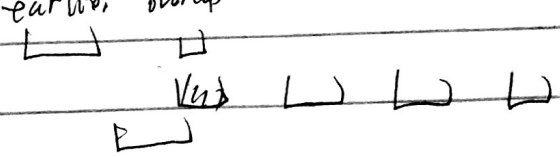
See 第2半结束

Claim There ~~dist~~ a optimal that contain the earliest finish movie.

Prof let O be an optimal solution for the movie problem

if O has the move that finish first then the claim true

If O has not has the move that finish first



Replace the first movie in O, with the movie that finish 1st to claim a new optimal solution

movie_selection (M[1...n]) // each M[i] has 2 fields, s and f

sort M[1...n] in increasing finish time

solution = M[1]

finish = M[1].f

for (i=2; i≤n; ++i)

if M[i].s >= finish

{

solution += {M[i]}

finish = M[i].f

}

$\frac{O(n \log n)}{O}$

Oct 13

MINIMUM-WEIGHT SPANNING TREE PROBLEM

A Graph $G = (V, E)$

Ex $G = (\{a, b, c, d\}, \{(a,b), (b,c), (c,d), (d,a)\})$

Convention: 1) $n = |V|$

2) $m = |E|$

$m \leq \binom{n}{2} = \frac{n(n-1)}{2}$ $\quad m \in O(n^2)$

Graph Representation

1) Adjacency matrix

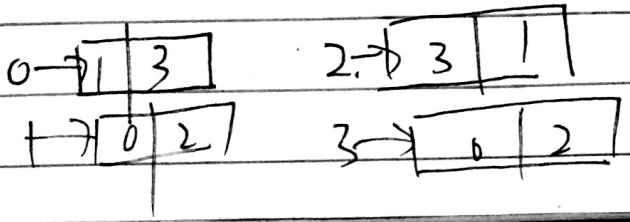
$\langle \{a, b, c, d\}, \{(a,b), (b,c), (c,d), (d,a)\} \rangle$

$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow$ $\quad (0,1) \quad (1,2) \quad (2,3) \quad (3,0)$

$0 \quad 1 \quad 2 \quad 3$ $\quad (1,0) \quad (2,1) \quad (3,2) \quad (0,3)$

	0	1	2	3
0	0	1	0	1
1	1	0	1	0
2	0	1	0	1
3	1	0	1	0

2) Adjacency Linked Lists



Space need

$2m + n$

Definition: 1) Let (a,b) be an edge.
We say a is adjacent to b
a is incident to (a,b)

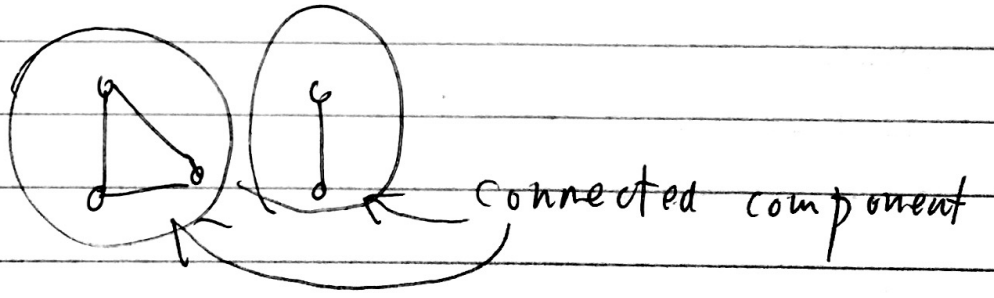
2) a path in a graph is a sequence of vertices
$v_0, v_1, v_2, \dots v_k$
such that $(v_0,v_1),(v_1,v_2)\dots(v_{k-1},v_k)$ are edge

3) a cycle is a path whose first and last vertices are the same

4) a simple path has no regressive vertices

5) a simple cycle, has no repeat vertices except the

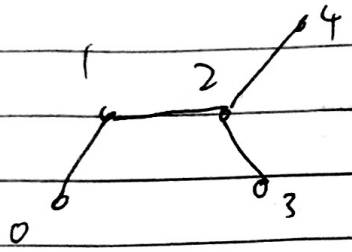
Def: A graph is connected if there is a path between any 2 vertices



Def: A graph is cyclic if it has a cycle otherwise it is acyclic.

Def: A graph that is connected acyclic is called a tree

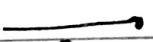
Ex:



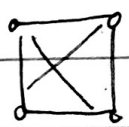
Thearm: A graph T is a tree if and only if T is connected and T has n vertices and n-1 edge.
2) T is acyclic and Th. n vertis and n-1 edges.
3) There is a unique path between any 2 vertices of T.

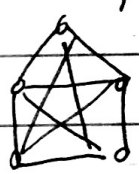
Speculate graph

1) compute . has all possible eadges

K_2 

K_3 

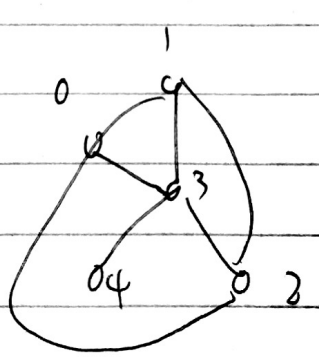
K_4 

K_5 

Def: $G = (V, E)$ on bipath
if $V = A \cup B$
$A \cap B = \phi$
such that if $(a,b) \in E$ then, $a \in A$ or $a \in B$
$b \in B$ $b \in A$

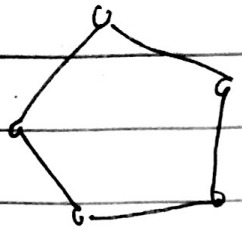
Bread-th first search
is a systematic way of visiting all the vertices in a graph

push start vertex on an empty query marked of.
While (queue is NOT empty)
remove from element of push of
unvisiable vertex on queue
4



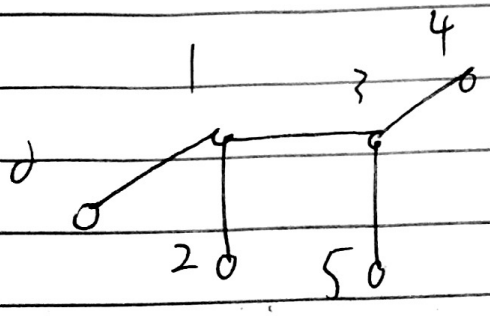
0 1 3 2

visited



	0	1	2	3	4
visited	✓	✓	✓	✓	✓

queue ~~4~~ 3 0 2 1



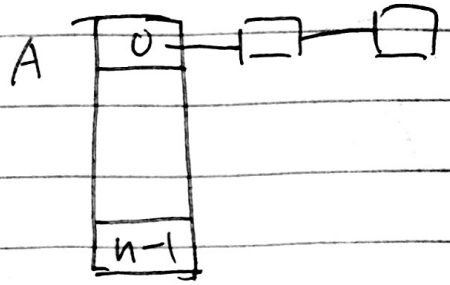
	0	1	2	3	4	5
visited	✓	✓	✓	✓	✓	✓

Queue ~~3~~ 1 ~~4~~ 5 0 2

A graph is bipartite iff it does not have an odd-cycle

10.15

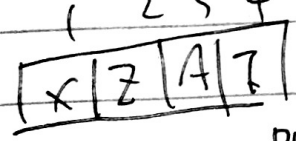
brute_MST (V, E, W) → (A[0..n-1])




```
{  ans = ∞
   E = the set of edge
   for each subset So of E
   {
      check that (V,S) is connected
      if weight (V,S) < and
                 and = weight (V,S);
   }
{  return ans;
}
```

$\Omega(2^m(n+m))$

A[0...k]



pow(2,k)

for (i=1; i<<2 k ++i) for (i=0; i < k;

s = bin(i)

i = 5

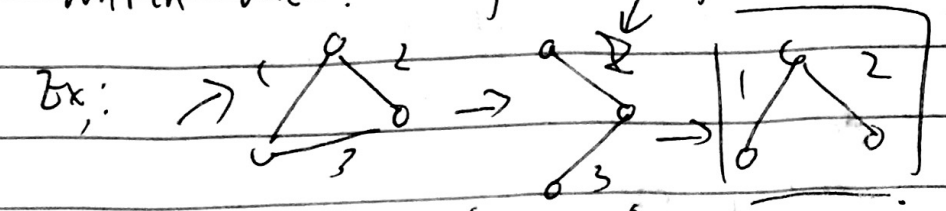
0 1 0 1

Greedy:

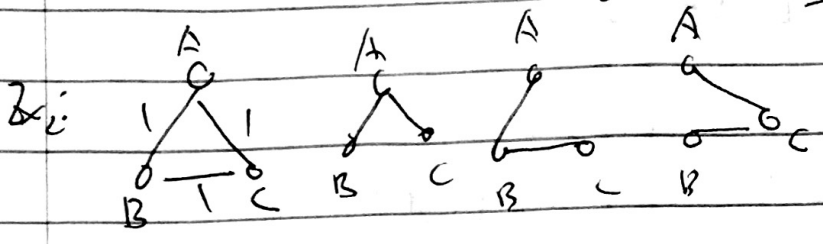
reduce the problem to one of smaller size

- remove an edge
- Which one? lightest edge

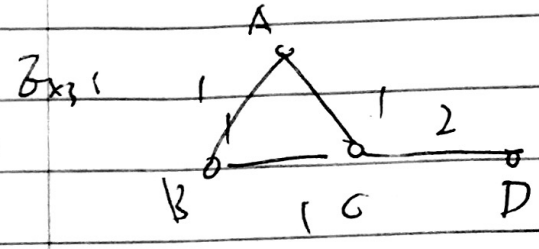
Ex:



Ex:



Ex:



not recycle.

greedy-MST(V, E, W)

{

sort E in non-decreasing weight

for each e in E

{

if subset is acyclic

S += {e}

}

if (S is connected)

return S

else return ∅

}

running time

$O(m \lg n)$

$m(\theta(n+m))$

greedy-disjoint-MST (V, E, W)

$O(m \log n)$ {

$S \leftarrow \emptyset$

Sort E in nondecreasing weight

$\Theta(n)$ (for each $v \in V$

MAKE-SET (v)

$\approx O(m+n)$ { for each $e = (a, b) \in E$

if (ROOT(a) != ROOT(b))

{

$S \leftarrow S \cup \{(a, b)\}$

JOIN (ROOT(a), ROOT(b))

}

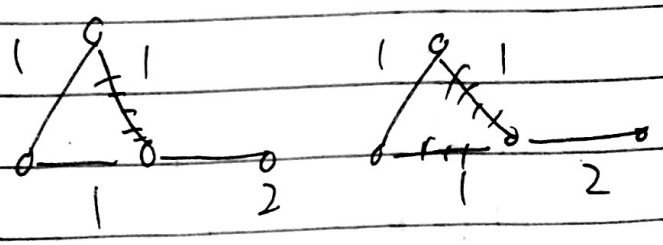
$b(m+n)$ { if (V, S) is connected

return (V, S)

else return $\emptyset$

pick lightest edge (a,b)

such that a has been visited . b has not



Prof

claim : KRUSKAL's Algorithm always produce an MST

Def. let S_i be the solution produced by kruskal's alg at the end of the iteration

$S_0 = \phi$ stop at any point could

Theom $S_i \subseteq$ some MST T_i $0 \leq i \leq m$

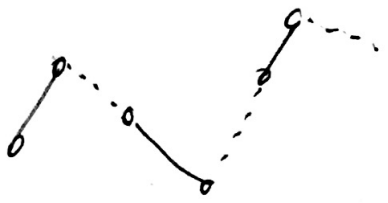
Proof : By induction on i.

$i = 0$

$S_0 = \phi \subseteq$ any MST.

Induction : Supposes $S_i \subseteq T_i$

case (a) $S_{i+1} = S_i \subseteq T_i = T_{i+1}$



Case b $S_{i+1} = S_i + \{(a,b)\}$ $(a,b) \in T_i$

$S_{i+1} \subseteq T_i = T_{i+1}$

Case i

T^{MST}

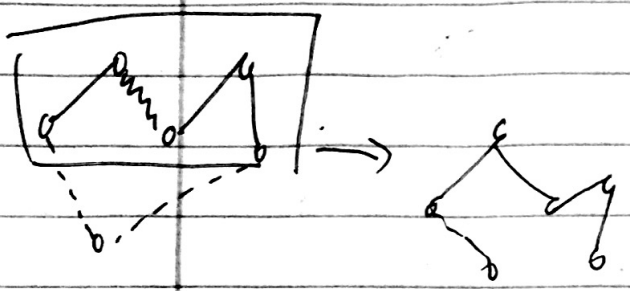
$S_{i+1} = S_i \cup \{a,b\}$ $(a,b) \notin T_i$

$T_i \cup \{(a,b)\}$ must have cycle which must contain an edge $(a',b') \notin S_i$

$w(a',b') \geq w(a,b)$

So $T_{i+1} T_i \cup \{a,b\} - \{a',b'\}$ is another spanning tree whose weight $\leq$ weight of T_i

So T_{i+1} is a MST that contains S_{i+1}

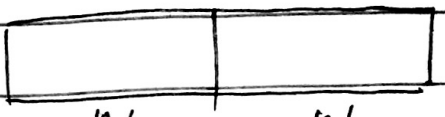


in cyclic

10.20

Brute
Greedy
Divide & Conquer

Merger Sort

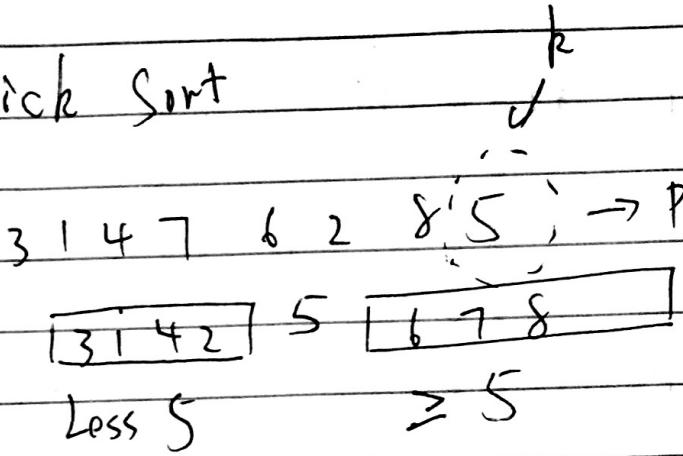


T: O(nlogn)

divide
conquer
combine

3 1 4 7 | 6 2 8 5
1 3 4 7 | 2 5 6 8

☆ Quick Sort




```
qs (A[left ... right])
{
    if (left < right
    {
        k = partition (A[left ... right]);
        // k is the new index of the pivot A[right]
        //qsA[left ... k-1]
        //qs (A[k+1 ... right];
```

notice: [k-1], k, [k+1]
k: pivot

```
partition (A[left .... right])
{
    i = left;
    for (j = left ; j < right ; ++j)
    {
        if (A[j] < A[right])
            swap(A[i++], A[j])
    }
    swap (A[i], A[right]),
    return;
}
```

$n = right - left + 1$

$$\sum_{j=left}^{right-1} (1)$$

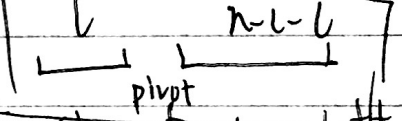
$= right - 1 - (left + 1)$

$= right - left$

$= n - 1$

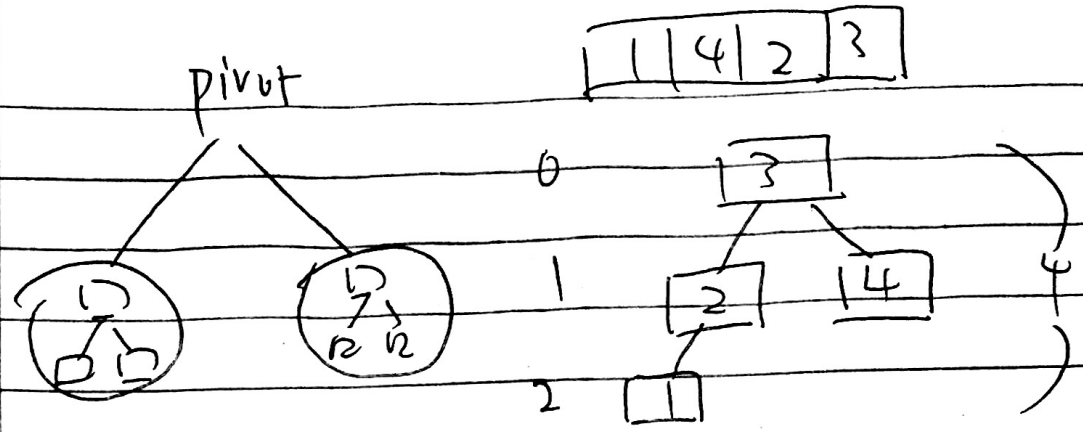
worst_case

$C(n) =$ # of $(A[j] < A[right])$ performed by QS an arrays if size n.



$C(1) = 0$ L is the size of the left partition.

$C(n) = \{n-1 + C(L) + C(n-1-L)\}_{max}, 0 \le L \le n-1$



$$\sum_{i=0}^{n-1} (i) = \frac{n(n-1)}{2} \quad \text{worst case}$$

$$\sum_{i=0}^{\lfloor \lg n \rfloor} 2^i \cdot i = n \lg n \quad \text{best}$$

Worst case: each node occupys a seperate level.

$$c(n) = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2} \in \Theta(n^2)$$

Best case: each level (except last) is

$$c(n) = \sum_{i=0}^{\lceil \lg n \rceil} 2^i \cdot i \in \Theta(n \lg n)$$

Average: $\Theta(n \lg n)$

SELECTION

INPUT select array A[left...right]

rank k, $1 \le k \le right - left + 1$

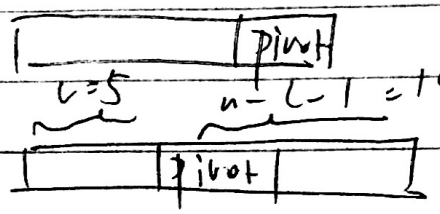
output. element with rank k

(k^{th} smallest element)

Sort A (left ... right)

return A[left ... k-1];

$\theta(n \lg n)$ mergesort.



n=20

k= 4th

Quick select (A[left ... right], k)

// assume $1 \le k \le$ right - left + 1

if (left == right) // base case

return A[left]

p = partition (A[left ... right]);

pivot_rank = p - left + 1

switch (comp (pivot_rank, k))

case 0 ; return A[p];

case 1 ; return quick_select (A[left, p-1]);

case -1 ; return quick_select (A[p+1, right, k-pivot

-N

Worst case of Quick Select (arrays already, sorted)

$$S(n) = \max_{0 \le i \le n-1} \{ n-1 + S(i) \}$$

$$S(n) = n-1 + S(n-1)$$

$$S(n) = \frac{n(n-1)}{2}$$

Best case

$n-1$

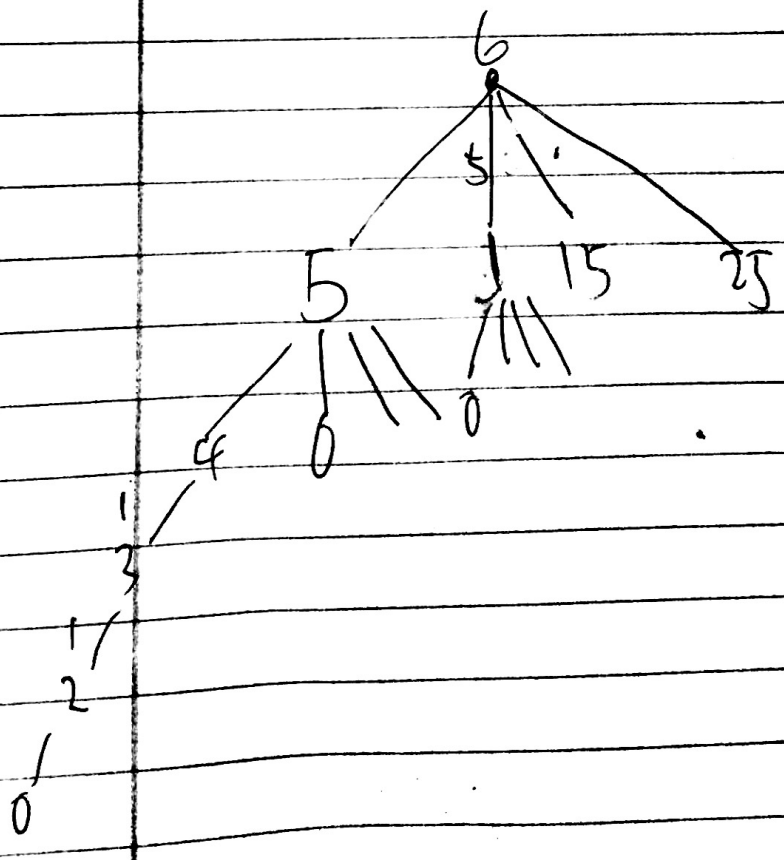
Ave. case : $\theta(n)$

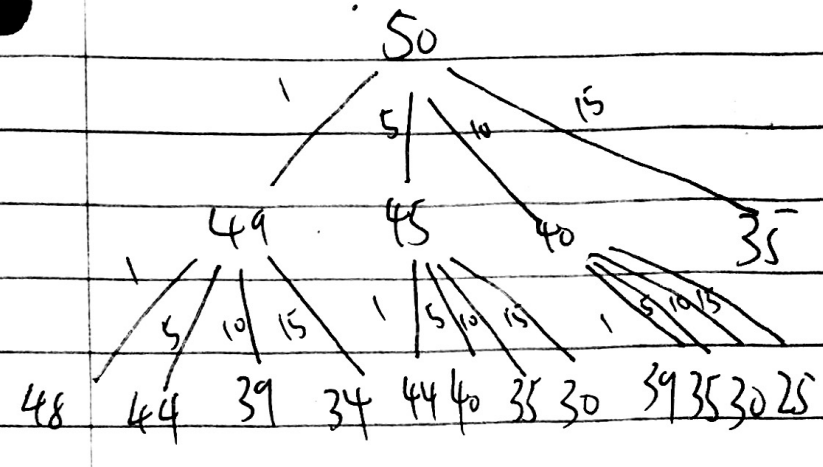
General coin change

INPUT: amount n d[1..k]

```
gc (n, d[1..k])
  if (n==0)
    return 0
  ans=∞
  for (i=1; i<=k; ++i);
  {
      ans = min(ans, 1+gcc(n-d[i], d));
  }
  return ans;
}
```

n=6 d=[1,5,10,25]





0, -1, -1 ...

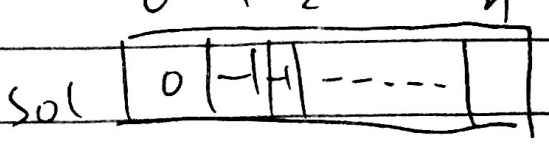
Memorize sol[0...n] = {-1}

```
mgcc (n, d[1...k])
{
  if (sol[m] != -1)
      return sol[n];
  ans = ∞,
  for (i=1; i≤k; ++i)
  { if (n - d[i] > 0)
      ans = min (ans, mgcc (n-d[i], d);
  }
  sol[m] = ans;
  return ans;
}
```

$O(kn)$

iteration version of mgcc

igcc (m, d[1...k])



iterative version of mgcc

```
igcc (m, d[1---k])
{
  sol[0]=0;
  for (i=1; i<=n; ++i)
  {
    ans=∞
    for (j=1; j<=k; ++j)
    {
      if (n-d[j]>=0)
        ans = min (ans, sol(n-d[j]))
    }
    sol [i] = ans;
  }
  return sol [n];
}
```

$\Theta(nk)$

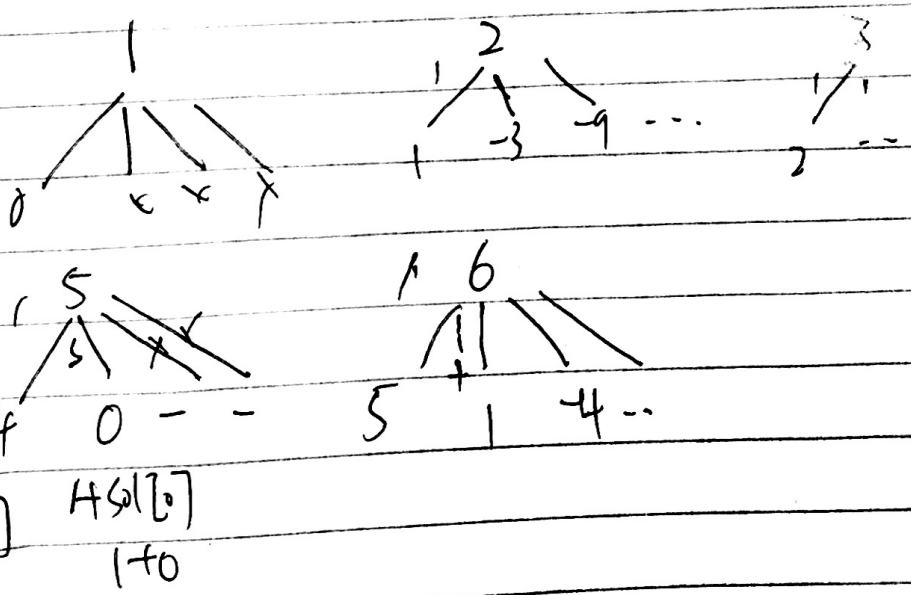
$d(i) = \min\{d(i-v_j)+1\}$

$i-v_j >= 0$,

v_j 表示第 j 个

sol

0	1	2	3	4	5	...
0	1	2	3	4	1	



Brute force
Greedy
Divide-conquer : Merger NOT overlapping

Dynamic programming: overlapping

☆ MATRIX CHAIN MULTIPLICATION (MCM)

Input n matrices

$M^1_{d_0 \times d_1}$, $M^2_{d_1 \times d_2}$, $M^3_{d_2 \times d_3}$ ... $M^i_{d_{i-1} \times d_i}$ x $M^n_{d_{n-1} \times d_n}$

rows, columns

$M_{2\times3}$ $\begin{pmatrix} 1 & 3 & 6 \\ 2 & 6 & 7 \end{pmatrix}$

$M_{3\times1}$ $\begin{pmatrix} 1 \\ 9 \\ -12 \end{pmatrix}$

$M_{1\times4}$ $(1\ 2\ 3\ 4)$

OUTPUT: compute $M^1 \times M^2 \times M^3$
using the minimum of scalar multiplication

$A_{x,y} \cdot B_{y,z}$

$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}_{2\times3} \cdot \begin{pmatrix} 8 \\ 9 \\ 10 \end{pmatrix}_{3\times1} = \begin{pmatrix} 56 \\ 131 \end{pmatrix}_{2\times1}$

Costs 6 scalar * j

# of scalar multiplications is: xyz

because xy elements in the results and each element request z multiplication.

$2 \times 3 = 3 \times 2$: but not commutate for matrix.

$M_{2\times3} \cdot M_{3\times1} \cdot M_{1\times4}$

2 ways to complete product

1) $(M_{2\times3} \cdot M_{3\times1})_{2\times1} \cdot M_{1\times4} = M_{2\times4}$

$2 \cdot 3 \cdot 1 + 2 \cdot 1 \cdot 4 = 6 + 8 = 14$

2) $M_{2\times3} \cdot (M_{3\times1} \cdot M_{1\times4})_{3\times4} = M_{2\times4}$

$24 + 12 = 36$

New input: $d_0, d_1, d_2, \cdots d_{n-1}, d_n$

for example: 2, 3, 1, 4

output: $2\times3 + 2\times4 = 14$

Brute: try all possible ways to multiply the chain of n matrices

But how many ways are there?

$n = 2, 1$

$n = 3, 2$

$n = 4, 5$ $\quad M_1, M_2, M_3, M_4$

1) $(M_1 \cdot M_2 \cdot M_3) \cdot M_4$ 2 way

2) $(M_1 \cdot M_2)(M_3 \cdot M_4)$ 1 way

3) $M_1 (M_2 \cdot M_3 \cdot M_4)$ 2 way

5

$n = 5 = 14$ $\quad M_1, M_2, M_3, M_4, M_5$

1×5 $\quad 1 \times 2$ $\quad 2 \times 1$ $\quad 5 \times 1$ (when $n=4$)

$C(n)$ = # of ways to multiply a chain of n matrices.

$$C(n) = \sum_{i=1}^{n-1} C(i) C(n-i) \quad \text{Catalan \# } j$$

Fact: $C(n) \geq 2^{n-2}$

Brute: Pick position of last $*$ and recurse.

Catalan # $\geq 2^{n-2}$ exp.

Greedy: input $d_0, d_1, \cdots d_{n-1}$

pick the least index to break into 2 chains

$d_0\ d_1\ d_2\ d_3$

$1 \times 100 \times 1 \times 1$: 分成 1×100, 100×1, 1×1

1×100 100×1 1×1 $1\times100 \times 100\times1$

$100^2+1 + 100 =$

$= 100^2 + 100$

DP:

Given problem

$M_1 * M_2 * \cdots M_n$

left

( ) × ( ) × ( )

$i < j$ right

$M_i \times M_{i+1} \times M_{j+2} \cdots M_j$

xy, yz

$M_1 \times M_2 \times M_3 \times M_4$

$M_{1-1} = 0$

$M_{2-2} = 0$

$M_{3-3} = 0$

$M_{nn} = 0$

	1	2	3	4
1	0			
2		0		
3			0	
4				0

$= d_i d_j$

$M_i \quad M_j$

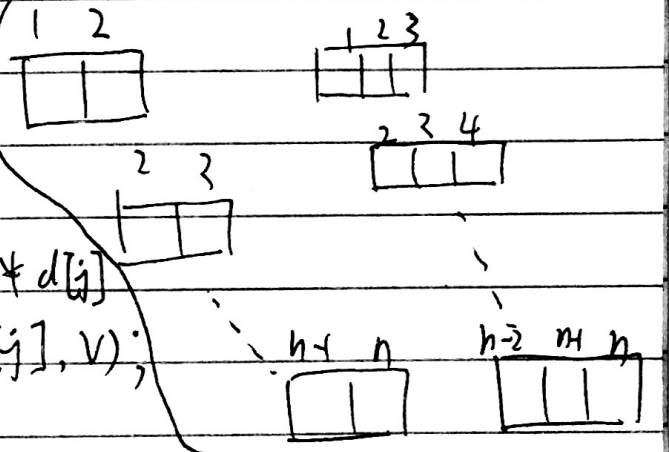
$d_{i-1} d_i \quad d_{j+1} d_j$

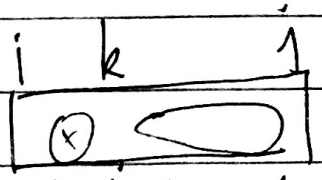
$c[1][3] + c[4][4]$

$+xyz$

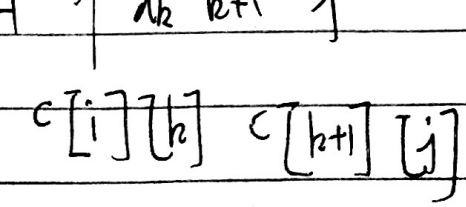
```
MCM(d[0...n]) // n matrices with dimension
                d0 x d1 x d1 x d2 ... dn-1 x dn
{ // compute costs of subchains of length 1
  for (i=1; i<=n; ++i)
      c[i][i] = 0;                    //  [ o o o o ]

  for (l=2; l<=n; ++l)                //  [ o o o o o ]   1-2, 2-3, 3-4, n-1-n
    for (i=1; i<=n-l+1; ++i)
    {
      j = i+l-1; c[i][j] = ∞;
      for (k=i; k<j; ++k)
      {
        v = c[i][k] + c[k+1][j]
            + d[i-1] * d[k] * d[j]
        c[i][j] = min(c[i][j], v);
      }
    }
  return c[1][n];
}
```






```
{
  v = c[i][k] + c[k+1][j] + d[i-1]*d[k]*d[j]
  if (v < c[i][j])
      c[i][j] = v;
      b[i][j] = k;
{ B[i][j]
```



Example: $d[] = [10, 100, 5, 50]$ $n=3$

$= M_{10\times100}, M_{100\times5}, M_{5\times50}$ 2200 + 5000

	1	2	3
1	0	5000	7500
2		0	25000
3			0

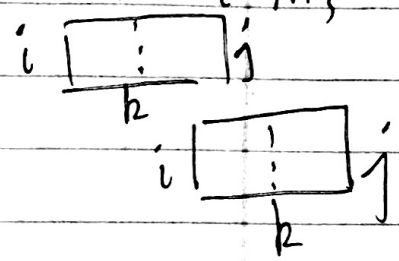
$l = 2$

$k=1 \quad i=1 \quad j=2$

$i=2 \quad j=3$

$d_0 \times d_1 \quad d_1 \times d_2 \quad d_2 \times d_3$

$M_1 \quad M_2 \quad M_3$



$k=1$

$M_1 \times M_2$

$M_1 \times M_1, \quad M_2 \times M_2$

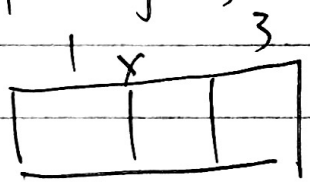
$0 \quad 0$

$d_0 \times d_1 \times d_2 = ?$

$= 10 \times 100 \times 5$

$l = 3$

$i=1 \quad j=3$

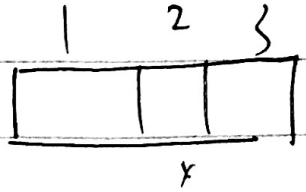


$c[1][1] + c[2][3]$

$+ d_0 \times d_1 \times d_3$

$= 2206 + 50000$

update

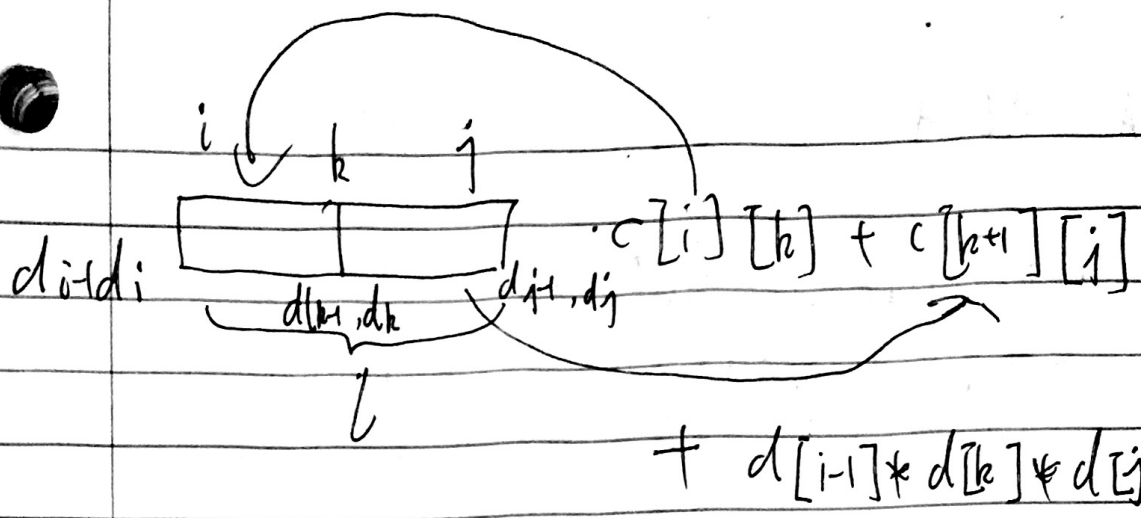


$c[1][1] + c[3][3]$

$d_0 \times d_2 \times d_3$

$+ = 5000 + 2500$

$= 7500$



$+ d[i-1] * d[k] * d[j]$

just like brute-force, but save intermidate value to the table, so not prove.

running time: $\sum_{l=2}^{n} \sum_{i=1}^{n-l+1} \sum_{l=i}^{i+l-1} \Theta(1)$

不仅得到最小值，还需要知道步骤

B[i][j] the best k for this subchain

```
Print MCM ( B[][], i, j)      print MCM (B, 1, n)
{
    print  i , B[i][j] , j
    if ( i<j ) // recursive
    {
      print MCM ( B, i , k)
      print  MCM ( B, k+1 , j)
    {
}
```

DNA MATCH

[x] x [xx] A, T, G, C

ATCGCGAT

A subsequence of $A_1, A_2 \dots A_k$
is obtained by removing a subsets of the symbols, pressing the relate ordering.

Ex. AB$\overset{x}{C}$B$\overset{x}{D}$AB
BBAB is a subsequence of s.

Note: a substring must form a contiguous block
this is not requr for subsequence.

Def. Let s, t be two sequence
The Longest common subsequence problem asks for a maximum length subsequence comme to s and t.

S = ABCBDAB
t = BDCABA

A
AB
BCA
BCAB BDAB

Note: these may be multiple common subsequence of maximum length

LCS

```
brute_LCS (S[1..n], t[1...m])
{
    for each subsequence α of S
        if α is a subsequence of t
            ans = max (ans, |α|)
    return ans;
}
```

running time exp

$X = x_1 x_2 \cdots x_m$

$Y = y_1 y_2 \cdots y_n$

Let Z be a LCS of X, Y

$Z = z_1 z_2 \cdots z_k$

Case 1: Symbol x_m is in Z, y_n is in Z $\Rightarrow$ $x_m = y_n = z_k$

Case 2: $x_m \in Z$ but $y_n \notin Z$

Case 3: $x_m \notin Z$ but $y_n \in Z$

Case 4: $x_m \notin Z$ $y_n \notin Z$

Case1: $x_m \in Z$, $y_n \in Z$

$x_m = y_n = z_k$

So $LCS(x[1...m], y[1...n])$

$= LCS(x[1...m-1], y[1...n-1]).x_m$

Case2: $x_m \in Z$, $y_n \notin Z$

$LCS(x[1...m], y[1...n]) = LCS(x[1...m], y[1...n-1])$

Case3: $x_m \notin Z$, $y \in Z$

$LCS(x[1...m], y[1...n]) = LCS(x[1...m-1], y[1...n])$

Case4: $x_m \notin Z$, $y_n \notin Z$

$LCS[x[1...m], y[1...m]) = LCS(x[1...m-1], y[1...n-1])$

$x[1...m]$

$y[1...n]$

i

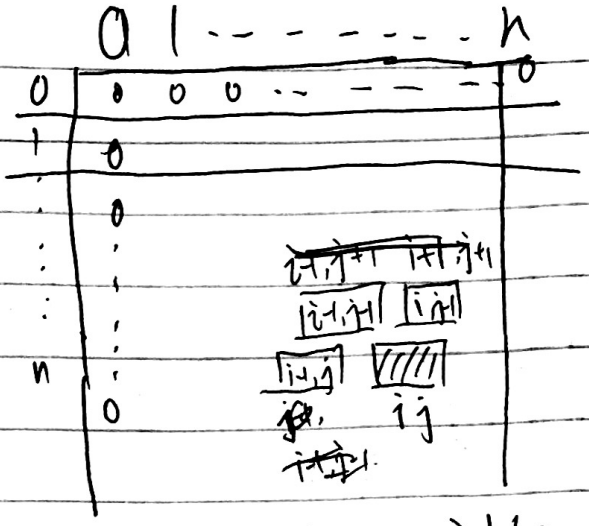
j

There are $m \times n$ subsolution

store in a $m \times n$ table.

$(m+1) \times (n+1)$

Base Case: when string has 0 symbol



from left → right
Top — bottom

$\theta(m \times n)$

Only 4 possible value

```
LCS (X[1.... m], Y[1...n])
{
   for (i=0 ; i≤n ; ++i)
     L[0][i] = 0;
   fo (i=0 ; i≤m ; ++i)
     L[i][0] = 0 ;
   for (i= 1 ; i≤m ; ++i)
     for (j=1 ; j≤n , ++j)
     {
        L[i][j] = max ( L[i-1][j], L[i][j-1] );
        if ( X[i] == Y[j] )
             L[i][j] = max (L[i][j], 1+L[i-1][j-1]);
     }
   return L[m][n];
}
```

O(n)
← O(m)
θ(m·n)

Case 3 Case 2

Case 1

for small size problem

For example: X = ABCBDAB

Y = BDCABA

		y	0	1 B	2 D	3 C	4 A	5 B	6 A
	x		0	0	0	0	0	0	0
A	1		0	0	0	0	1	1	1
B	2		0	1	1	1	1	2	2
C	3		0	1	1	2	2	2	2
B	4		0	1	1	2	2	3	3
D	5		0	1	2	2	2	3	3
A	6		0	1	2	2	3	3	4
B	7		0	1	2	2	3	4	4

... X = AB

Y = BDCABA

```
if (L[i][j-1] > L[i-1][j])
    L[i][j] = L[i][j-1];
    D[i][j] = ←
else L[i][j] = L[i-1][j];
    D[i][j] = ↑

if (x[i] == y[j] && (1+L[i-1][j-1] >
    L[i][j]))
{
    L[i][j] = 1+L[i-1][j-1];
    D[i][j] = ↖;
}
```

Exp: S = CHARACTER

Final longest subsequence of S
that is a padnelrome 对称
↳ CARAC

Reverse S, S^R = RETCARAHC

Example

S = 1 5 7 4 3 2 5 8
find longest subsequence that is nondecreasing ↑
1 5 7 8
1 4 7 8 Convert idea!
1 2 7 8

T = 1 2 3 4 5 6 7 8 find T and S longest same

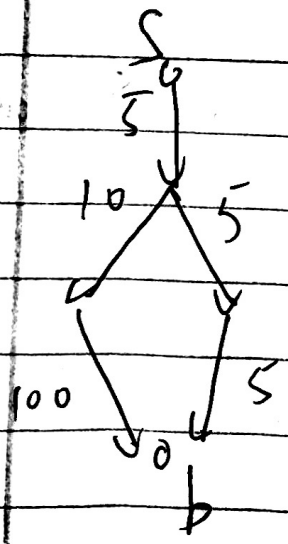
Example
3 common mutation A C T G
- delete: A C ~~T~~ G
- insert: A C (C) T G
- replace: (G) C T G

ROD CUTTING

1 $ - #5
2 $ - #15
3 $ - #6
6 $ - #15

Input : a rod of n $

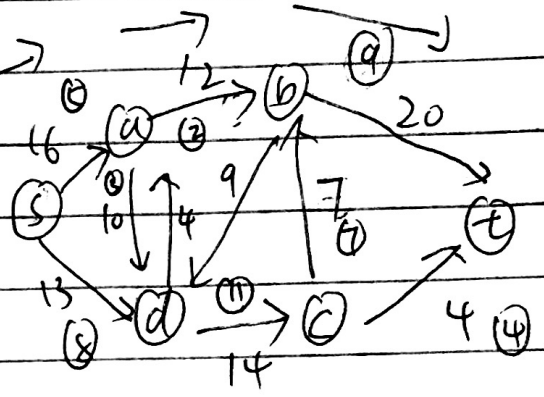
Output : cut rocks into segments to ~~open~~ optimal price



FLOW Problem

INPUT: A directed weighted graph
$G = (V, E, c)$
with 2 special vertices
(called the source (s)
and the sink (t)
where $c: V \times V \to \mathbb{R}$ is a nonnegative function

INPUT:



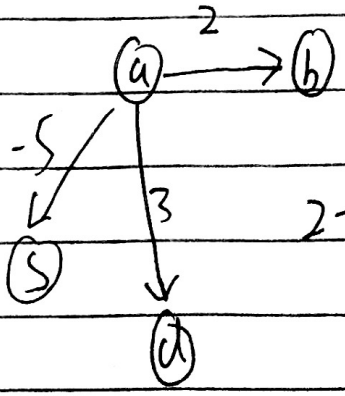
OUTPUT: A flow $f: V \times V \to R$ subject to the following

1) capacity constraint: no sent more
$f(a,b) \leq c(a,b), \quad a,b \in V$

2) skew symmetry: $f(a,b) = -f(b,a)$

3) Conservation: $\sum_{b \in V} f(a,b) = 0$
for $a \in V - \{s, t\}$
mean a not {source / sink}

with maximum value
where value(f)
writen as $|f|$.
is defined to
$|g| = \sum_{b \in V} f(s,b)$



$2 + 3 + (-5) = 0$

Note: How on an example of a linear programming problem, where the object is to maximum a linear combination of a some variable subjected to some constraints:

$\sum_{b \in V} V_{a,b} = 0$ $a \in V - \{s,t\}$	$0 \le V_{s,a} \le 16$ $0 \le V_{a,b} \le 12$	$V_{a,b} \neq (V_{b,a})$ negative	maximum $\sum_{a,b \in V} V(a,b)$

Solution:

Contents of finding AUGMENTS paths start with the empty flow

$f(a,b) = 0, \forall a,b \in V$ (all 3 constrain

while (not done)

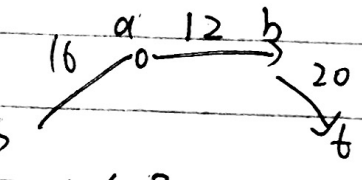
{

step1: pick any path p from s to t

step2: define $c(p) = \min_{(a,b) \in p} C(a,b)$

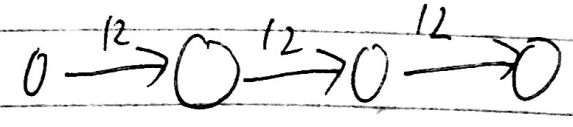
$f_p = f_{low}$

bottom



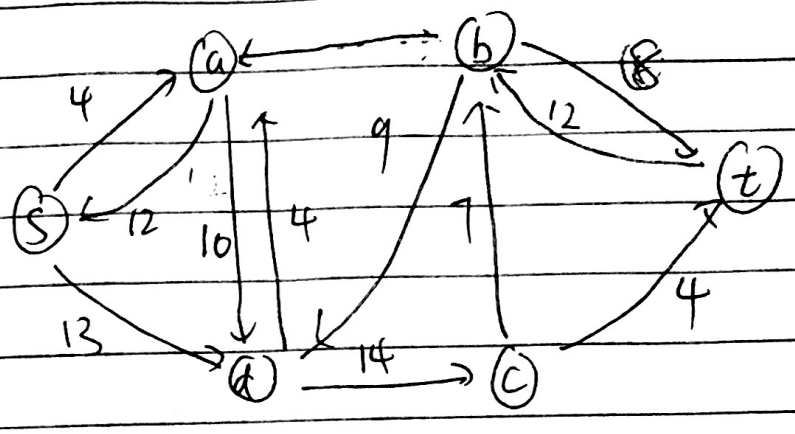
$$f(a,b) = \begin{cases} c(p) \text{ for } (a,b) \in p \\ 0 \text{ otherwise} \end{cases}$$

step3: $f = f + f_p$



Step4 compute residual retreat

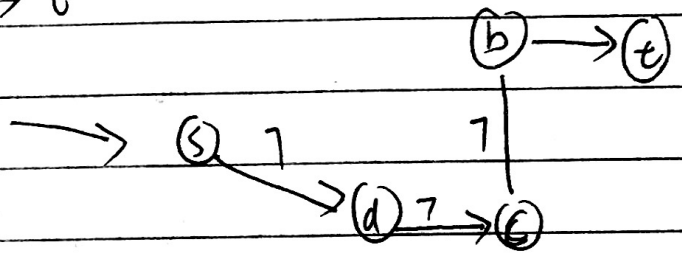
Residual Network ← 剩余网络

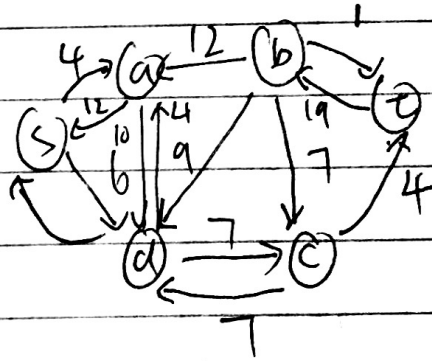


s → d → c → b → t

13 14 7 8

min





s →(4) a →(10) d →(7) c →(4) t

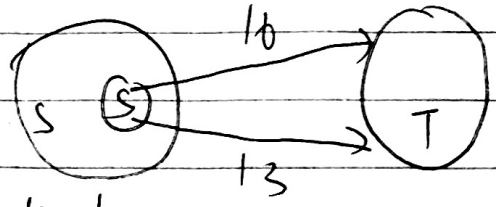
(S,T)

A CUT of a flow network is just a partition.

if V into $S,T \Rightarrow S = S \cup T$

$S \cap T = \emptyset$

such that : $\begin{cases} \text{Source } s \in S \\ \text{sink } t \in T \end{cases}$



Theaom (MAX_FLOW/MIN CUT)

Let $G = (V, E, C, S, t)$ be a flow network.

the following are equivalent;

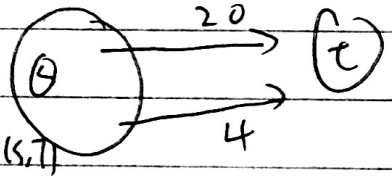
1) f is a max flow of G

2) G_f has no augmentary path

3) $|f|$ is the capacity of some cut (S,T) of V

↑ cut capacity = 24

looking different cut



Cut capacity $= 20 + 4 = 24$

Claim: Let f be a flow

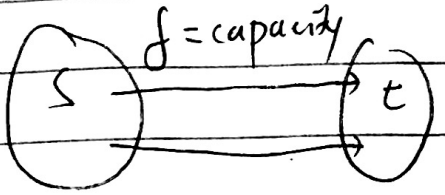
$|f|$ is $\leq c(S,T)$ for any cut

Proof: Intuitively clear

(1) ⇒ 2) if there is an augmenting path. then we can sent some more along this path from s to t, contradictory assumption that f is a maximum flow.

(2) ⇒ (3)

if G has augemoty path.



$S = \{v, V : v$ reachable from s in Graph $G\}$ S T

$T = V - S$

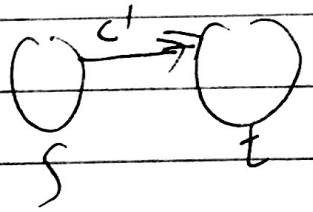
This is a cut, since $t \in T$

claim: $c(S,T) = |f|$

claim

Prof: If not than there is edge (a,b), $a \in S$, $b \in T$

$c(a,b) > f(a,b)$



But $c'(a,b) > 0$ in G_f

So b in reachable from s

contradiction.

3) ⇒ 1)

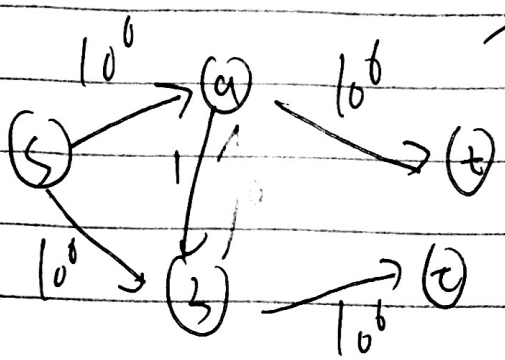
$|f| = c(S,T)$ for some cut (S,T)

But $|f| \leq (S,T)$ for any cut (S,t) $|f|$ must be optimal

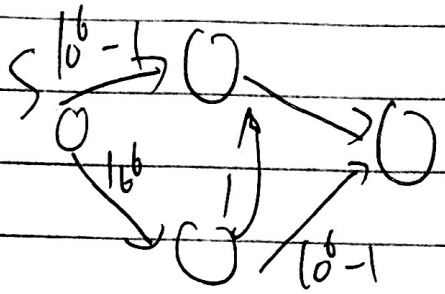
KARK-EDMONDS

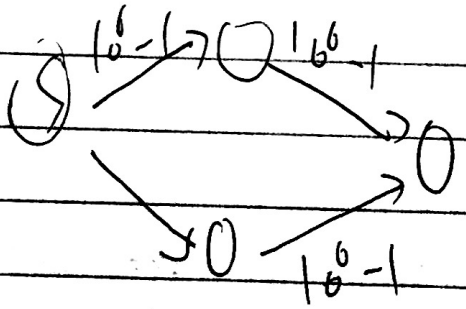
FORD-FULLERSON

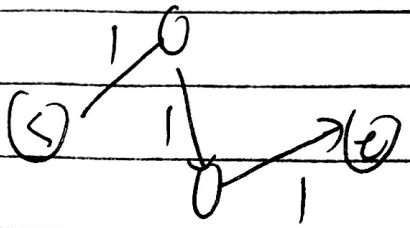
Let user pick → worst $G(|V|^3)$

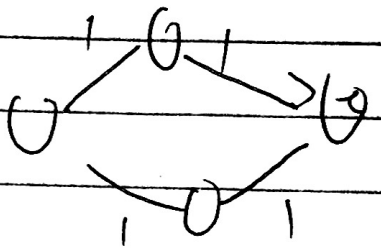


residue









$||ogM||V|$

SATISTABILITY

INPUT: a Boolean expression
n variable $x_1 x_2 \cdots x_n$
Boolean Operator $\wedge, \vee, \neg, ()$

Ex. ① x_1

② $x_1 \vee x_2 \wedge \neg x_3$

③ $(x_1 \vee x_2) \wedge x_3$

$x_1 \vee x_2 \wedge x_3 \Rightarrow x_1 + x_2 \cdot x_3$

$\Rightarrow x_1 \cup (x_2 \wedge x_3)$

Precede rule
$\neg$
$\wedge$
$\vee$

OUTPUT: [Yes]. if there is a truth argument to $x_1 \cdots x_n$ such that the formula evalue to true

[No]. if not

④ $x_1 \wedge \neg x_1$

① $x_1 \to T$: $x_1 = T$

② $x_1 \vee x_2 \wedge \neg x_3$: $x_1 \to T$ $x_1 = T$
$x_2 \to T$ $x_2 = F$
$x_3 \to F$ $x_3 = F$

③ $(x_1 \vee x_2) \wedge x_3$

$x_1 = T$
$x_2 = T$
$x_3 = T$

④ False

Brute Force

```
SAT(F, X1 ... Xn)
{
    for each truth assignment to X1 ... Xn {2^n different ways}
    {
        if (a(F) == T)
            return Yes
    }
    return NO
}
```

{using stack to compute operation cost: O(n)}

$\Omega(2^n \cdot n)$

VERIFICATION

TO CONVINCE SOMEONE That problem H is hard

Show this:

If H is easy

{ H has p poly-time alg,

then SATISFIY is also

(has p poly time solution)

SAT

Restricted version of SAT

1) CNF-SAT

Input: a Boolean expression
in conjunctive normal Form
on n variable $x_1 \dots x_n$

Output: Yes if there is a truth assignment to $x_1 \dots x_n$ that make X true.
No if not

Def: A clause is a Bool expression of the

Def: A LITERAL is a variable x of the
of variable $\neg x_i$

$l_1 \vee l_2 \vee \dots \vee l_k$
where each l_i is a literal.

Ex. $(x_1 \vee x_2)$

$(\neg x_1 \vee \neg x_2 \vee x_4)$

$(\neg x_1 \vee \neg x_2 \vee x_3 \vee x_4 \vee \neg x_5)$

Def: A boolean formula $\mathcal{F}$ is in CNF
if it has the form
$C_1 \wedge C_2 \wedge \dots \wedge C_m$ where each C_i is a clause

where each C_i is a clause

E_x: $(x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee x_3 \vee \neg x_4 \vee x_4) \wedge (\neg x_1 \vee \neg x_2)$

2) 3-SAT

Input: a boolean expression X in 3-CNF (each clause has exactly 3 different literals)

output: Yes / No

3 3 3

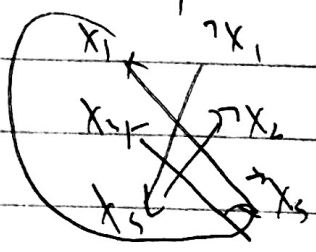
E_x: $(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (x_2 \vee x_4 \vee \neg x_5)$

3

3) 2-SAT (time)

Input: a boolean expression in 2-CNF (each clause has exactly 2 different literal.

output: Yes / No



$C_1 = (x \vee y)$ $(x_2 \vee x_2) (\Rightarrow \neg x_2 \rightarrow x_3$

$\Downarrow$

$\neg x \rightarrow y$

(1971) COOK'S THEOREM

If SAT is easy, then
Every problem in NP is easy.

Def: P is the class of problem (Whole problem)
which has a polynomical time algorithm

SORTING $O(n \lg n)$
MEDIAN $O(n)$
MATRIX $O(n)$
LCS $O(n^2)$

$\Rightarrow$ even $O(n^{100000})$

P (polynomical time)

In other words, a problem is in P if its solution can be FOUND in polynomial time

① 3 4 1 5 2 → 1 2 3 4 5

② 1 2 ③ 4 5

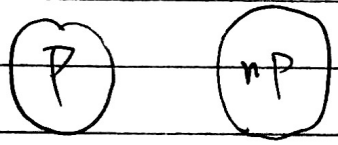
③ A B B A C B A
B A B A C B

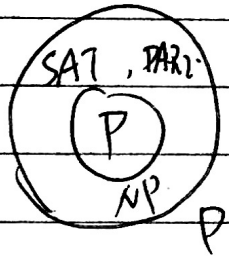
Def: A problem is NP if its solution can be CHECKED in polynomical time

INPUT: $w_1, \ldots w_n$

OUTPUT: There is a partition, x, y of the integers such that SUM(x) = SUM(y)

(3), 1, (7), (2), 11 $\begin{cases} 3+7+2=12 \\ 1+11=12 \end{cases}$





P vs NP

$P \subseteq NP$

IDEA behind Cook's Theorm

Rephrase each problem in NP as a Boolean expression.

Example: Another NP problem

3-COLORING problem

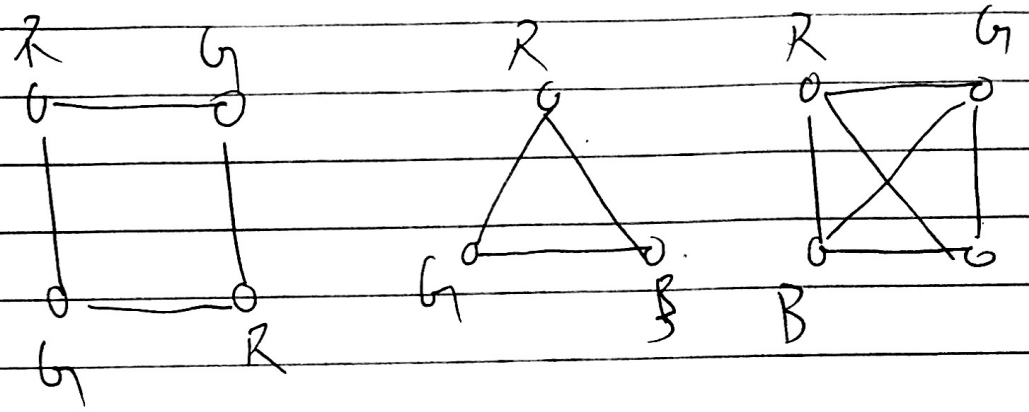
INPUT an Undirected graph $G = (V, E)$

OUTPUT: yes there is a coloring

$c: V \to \{R, G, B\}$

such that $c(a) \neq c(b)$ if $(a,b) \in E$

No, otherwise

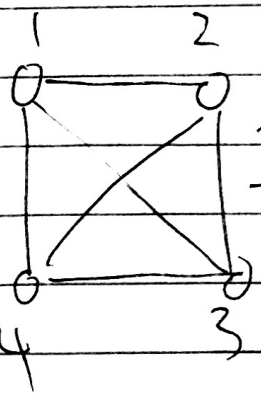


3-COLORING is in NP

Given a coloring $c: V \to \{R, G, B\}$

```
for each edge (a,b) ∈ E
    if c(a) == c(b)
        return FALSE
return true,
```

$O(|E|)$



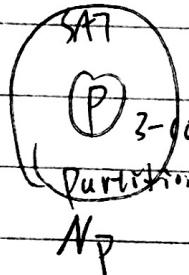
translate → to a Boolean expression with SAT

for each vertex: R_i, G_i, B_i

$$(R_1 \vee B_1 \vee G_1) \wedge (R_1 \to \neg B_1) \wedge (R_1 \to \neg G_1)$$
$$\wedge (B_1 \to \neg R_1) \wedge (B_1 \to \neg G_1)$$
$$\wedge (G_1 \to \neg R_1) \wedge (G_1 \to \neg R_1)$$

$$\wedge (R_1 \to \neg R_2) \wedge (R_1 \to \neg R_3) \wedge (R_1 \to \neg R_4)$$
$$(G_1 \to \neg G_2) \wedge \cdots$$

Preview.



KARP → 2 problem.

→ NP-complete.

- Divid
- DP
- Flow-network

P-NP: easy / hard } show P → easy

P-NP → hard → NP → hard

If SAT $\in$ P, then P = NP

In fact cook sh

Something more precise

every pooblem on NP

REDUCTION TO SAT

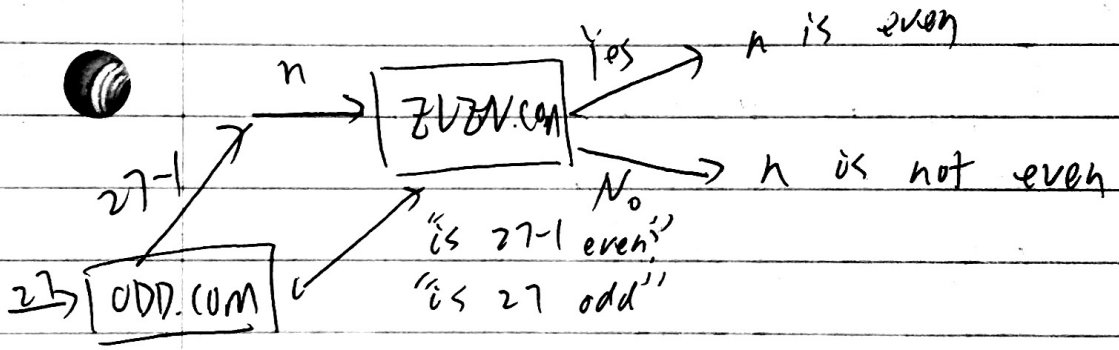
REDUCTION

| Even |

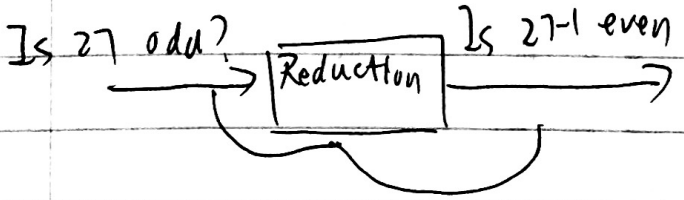
Input: integer n

Output: yes of n is divide by 2

No. of n is not dividable.



is equvilent to (is n-1 even)



n is odd $\Rightarrow$ n-1 is even

n is not odd $\Rightarrow$ n-1 is not even

① Same answer.

② Translation is easy to cover.

Reduction function f(n) = n-1

we ~~substitute~~

ODD $\leq$ EVEN

gcd(9,6)=3

gcd(a,b)

a,b → lcm.com —a,b→ gcd(a,b)

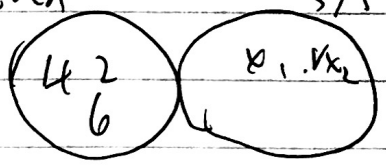
gcd(a,b) —a b→ lcm(4,6)=12

Reduction

Def:

Note: A yes-no problem can be as a s:

Even SAT



Def: Let A and B be Yes-No problem (decision) ODD or EVEN

we say "A reduces to B"

and write "$A \leq B$"

if there is a polynomical-time fuction f()

such that 1) $x \in A \Rightarrow f(x) \in B$

2) $x \notin A \Rightarrow f(x) \notin B$

LOGICAL EQUIVANCE

$X \Rightarrow Y \equiv \neg Y \Rightarrow \neg X$

1) $x \in A \Rightarrow f(x) \in B$

2) $f(x) \in B \Rightarrow x \in A$

$x \in A \iff f(x) \in B$ every

Cooks Theorem

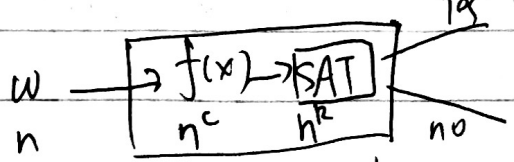
Let $X \in NP$ then $X \leq 3\text{-SAT}$

SAT

Corollary: If SAT $\in P$, then $X \in P$ for $X \in NP$

i.e $P = NP$

Proof: Let $X \in NP$. Construct the following alg for X



$x \quad (n^c)^k$

Running time

$O((n^c)^k) = O(n^{ck})$

polynomial

Proof:

Suppose 3-SAT $\in P$, so there is an alg A 3-SAT to solve 3-SAT in time $O(n^k)$ for some constant k

construct an Alg A_X for X as

input w

return $A_{3\text{-SAT}}(f(w))$; $\left.\begin{matrix}\\\\\end{matrix}\right\} O((n^c)^k) = O(n^{ck})$

where f is the polynomial-has reduction from X to 3-SAT, whose running time is $O(n^c)$ for some constant c.

Def. If $X \leq H$ for $X \in NP$
then we say H is NP-Hard

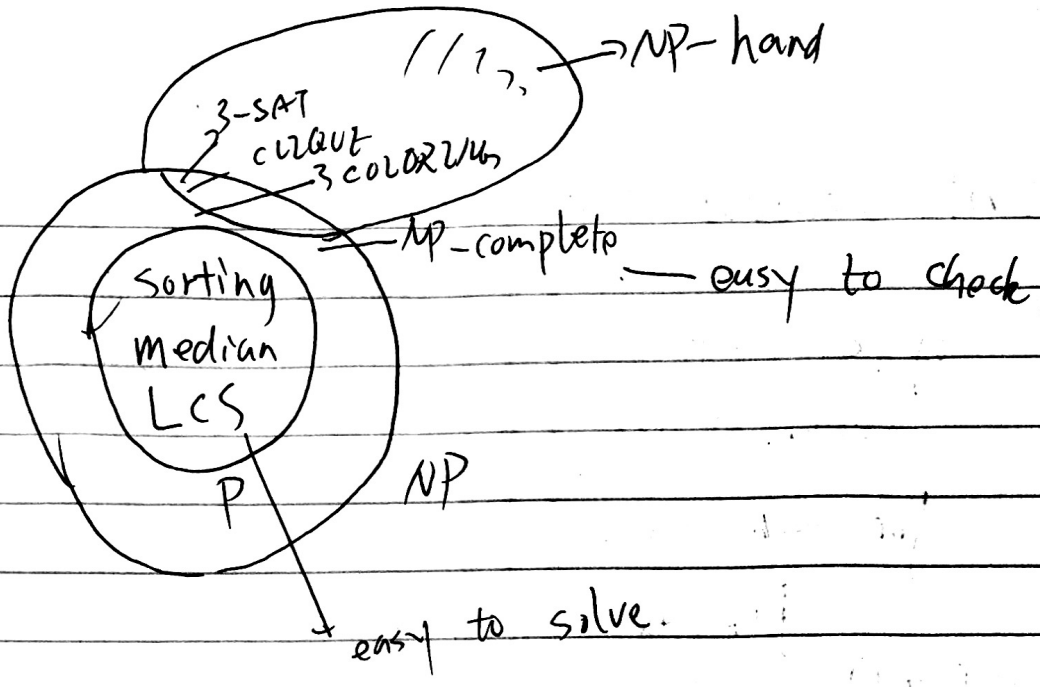
Def: If H is NP-hard and $H \in NP$
then we say H is NP-complete.

KARP THEORZM
other problems are NP-complete

CLIQUE
HAM-CYC
VERTEX-COVER
3-coloring
3D-MATCHING
KNAPSACK

CLIQUE $\leq$ 3-SAT
3-SAT $\leq$ CLIQUE

CLIQUE $\equiv$ 3-SAT



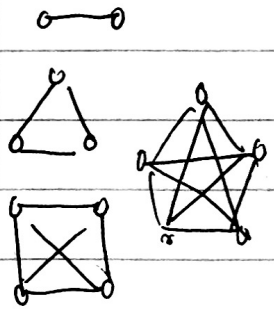
NPH:

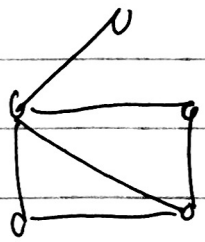
CLIQUE:

INPUT: - undirected graph $G=(V,E)$
- positive integer B.

PUT: yes if there is a complete subgraph of G on B (all the edge are present)
No if not

Ex:





$B=4$ NO

$B=2$ Yes

$B=3$ Yes

Theorm: CLIQUE $\in$ NP

Prof: We need to · a poly-time alg to verify its solution

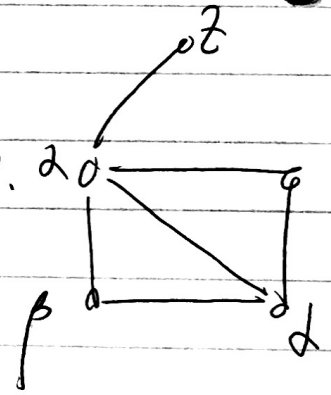
set of B vertices

verify-CLIQUE-solving (V, E, B, C)
question proposed solution

{
  if |C| != B
    return false;
  for each x in C
    for each y in C
      if (x != y && (x,y) ∉ E)
        return false;
  return true;
}

$O(|V|^2|E|)$
$O(|V|^4)$

(α, β) ∈ E
(α, α) ∈ E c = {α, β, δ}
(β, δ) ∈ E



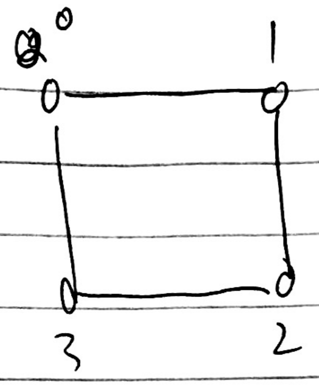
INDEPENDENT SET (IND)

INPUT: undirected graph G = (V, E)
a positive integer β |I| = β

output: Yes if there is a set I ⊆ V
such that if x ∈ I and y ∈ I, then (x, y) ∉ E

No. otherone

Ex.



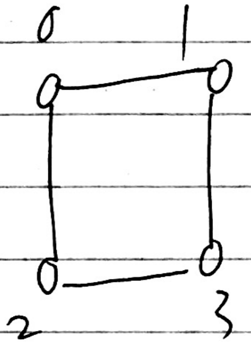
$B = 2$ (Yes)
$B = 3$ (No) (1, 3) no edge connect.

VERTEX COVER (VC) (NPC)
(NODE COVER) (NC)

INPUT: undirected graph $G = (V, E)$
positive integer B

output: Yes if $C \subseteq V$, $|C| = B$
such that if $(x, y) \in E$
then $x \in C$
or $y \in C$
or both

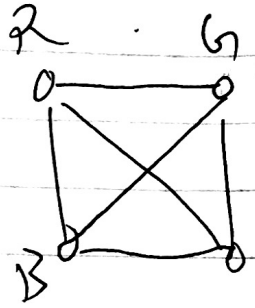
Ex:



$B = 1$ No
$B = 2$ Yes
$B =$

cover, [illegible]

3-COLORING
INPUT: an undirected grap
output: Yes if $C: V \to \{R, G, B\}$
such that if $(x, y) \in E$
then $C(x) = C(y)$



2-1) MATCHING

INPUT: ① a set B

② a set G

$|B| = |G|$

③ $L \subseteq B \times G$

(Bob, Alice)

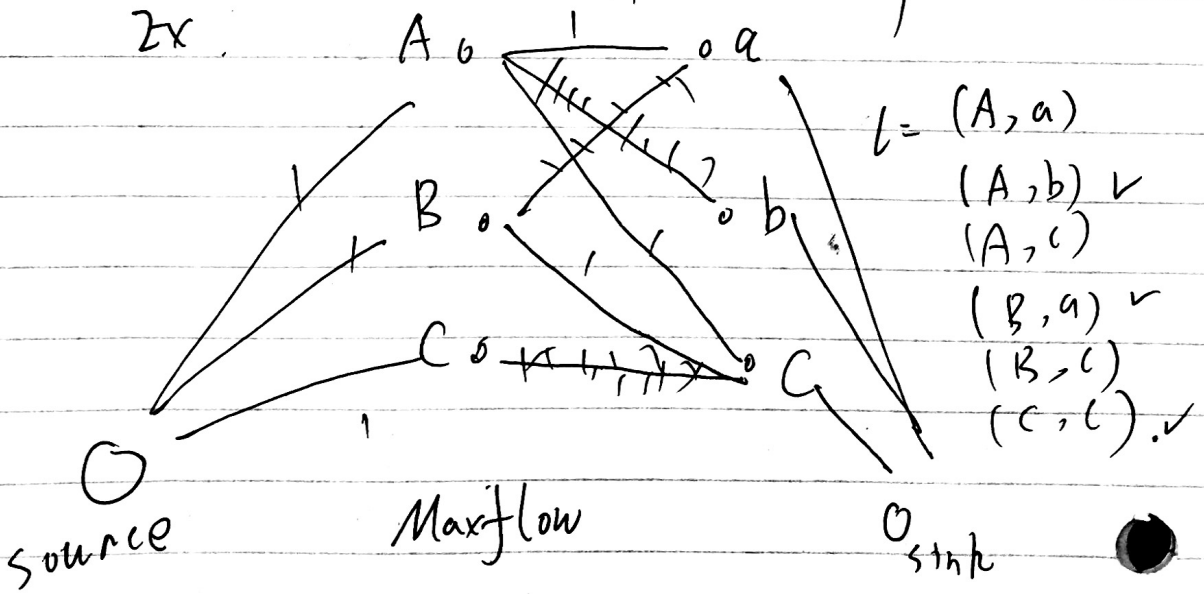
output: Yes if there is a set $S \subseteq L$

such that each $b \in B$

and each $g \in G$

appead exactly one in S

Ex.



3-D matching (NPC)

Input 3 sets, B, G, H, $|B| = |G| = |H| = n$.

list of m i

$$L = \{(b_1, g_1, h_1), (b_2, g_2, h_2)$$

output: Yes if

$S \subseteq L$

such that each $b \in B$

$g \in G$ in S

$h \in H$

SUBSET-SUM positive (NPC)

INPUT: - a set of n integer $a_1, \ldots a_n$

- a positive integer B

output: Yes if there is a subset of $a_1, \ldots a_n$ whose sum is B

Ex: $\{1, 2, 7, 9, 25\}$

$B = 29$.

PARTITION (NPC)

INPUT: n positive integer $a_1 \ldots a_n$

output: Yes if the integer can be partition 2 sets of equal sum

$\{1, 3, 27, 18, 1\}$

(NPC)

HAMILTON PATH

Acyclic is easy

INPUT: a directed graph $G=(V,E)$

OUTPUT: Yes if there is a simple path that visits each vertex once.

Yes.

TRAVELING SALESMAN

INPUT: a weighted graph (V,E,W)
a positive integer B.

Output: Yes if there is a Hamilton path whose total weight is B.

Theorm: If H is NP-Hard. and $H \leq X$, Then X is also NP-Hard

Cook: 3-SAT is NP-Hard

kap: $3SAT \leq IND$

IND is NP-hard.

$X \in NP \quad X \in H$

Proof: We need to show that for every $S \in NP$, $S \leq' X$,

is that in a poly- function

$w \in S \iff f_S(w) \leq X$

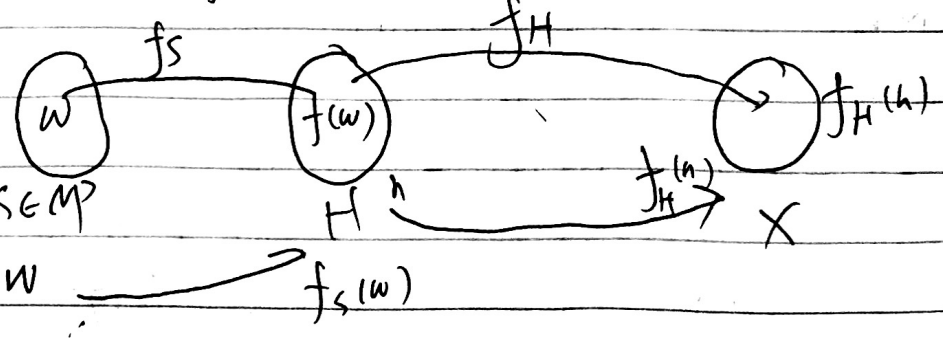
Hypoisi: Since H is NP-Hard, there exists f_H such that for every $S \in NP$

for every S in NP there exists a poly-time function g_S such that $w \in S \iff g_S(w) \leq H$

Hypi: $H \leq X$

There exists poly-time function f_H such that

$f \in H \iff f_X(h) \leq X$



$s \in S \Longleftrightarrow g_s(w) \in H$

$\Leftrightarrow f_H(g_s(w)) \in X$

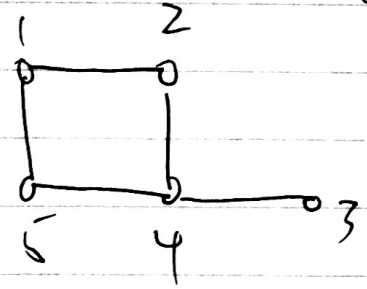
Def: $f_s(w) = f_H(g_s(w))$

$w \in S \Leftrightarrow f_s(w) \in Y$

IND Pch

INPUT. undirected graph $G=(V,E)$
a positive categer B

OUTPUT: Yes if there exists $I \subseteq V$,
$|I| = B$
such that if $a \in I$, $b \in I$ $a \neq b$
then $(a,b) \notin E$



$B=3$

$(2,5,3)$

互不相连 (任两个)

Theoram: IND is NP-complete

Prof:

Step: (1) IND is NP-completed
solution is a subset I of V
Verify-IND-SO((V,E,B,I) / Q) — solution

无向图，E=2V. 图

你们要的 Question 和 验证的答案.

```
{
  if |I| != B)
      return false
  for (a∈I)
    for (b∈I)
      if (a != b && (a,b)∈E)
          return false
  return true
```

$O(|V|^2|E|) = O(|V|^4)$

3) 3-SAT ≤ IND

knew ≤ New

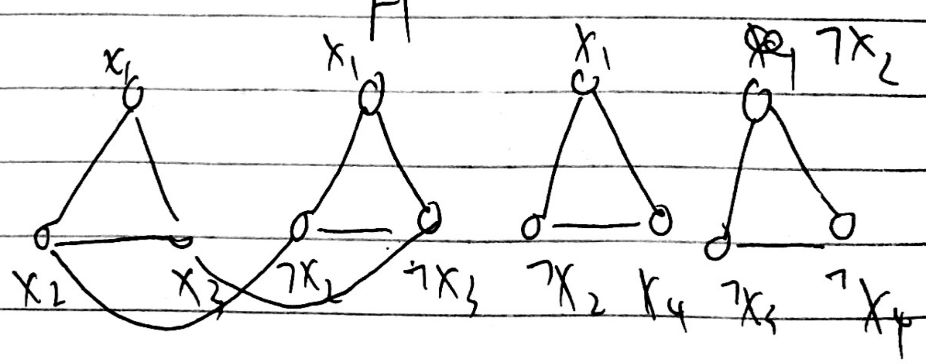
Polynomic reduction f

Reduction example

$X: (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4)$

G

B = # of H



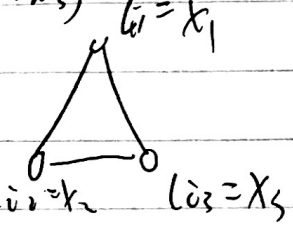
$x_1 \vee x_2 \cdot \neg x_1 \to x_2$

$f(X = C_1 \wedge C_2 \wedge \cdots \wedge C_m) \to$ satisfy?

$C_i = C(l_{i1} \vee l_{i2} \vee l_{i3})$

27 is odd $\xrightarrow{f}$ 27+1 is even

$V = \{V_{11}, V_{12}, V_{13}, \ldots, V_{m1}, V_{m2}, V_{m3}\}$



$O(m)$ {
$V = \phi$
for ($i=1$, $i \le m$; ++i)
$V = V \cup \{V_{i1}, V_{i2}, V_{i3}\}$
label(V_{i1}) = l_{i1},
label(V_{i2}) = l_{i2} label(V_{i3}) = l_{i3}
{ return (V, E, R))

$E = E + \{(V_{i1}, V_{i2}), (V_{i2}, V_{i3}), (V_{i3}, V_{i1})\}$ (m)

$O(9m^2)$ {
for ($i=1$; $i<m$, ++i)
for ($j = i+1$; $j \le m$; ++j)
for ($x=1$, $x \le 3$; ++x)
for ($y=1$; $y \le 3$; ++y)
if (label(V_{ix}) $\neq$ label(V_{jy}))
$E = E \cup \{(V_{ix_1}, V_{jy})\}$

$O(9m^3)$

Is 27 odd $\xrightarrow{f}$ Is 27+1 even)
Yes $\longrightarrow$ Yes
No $\longleftrightarrow$ Yes

$x_1 = T$

$x_2 = F$

$x_3 = T$

step 3) $x \in 3\text{-SAT} \Rightarrow f(x) = (V, Z, B) \in IND$

suppose there is a truth assignment

$a: \{x_1, x_2, \ldots x_n\} \to \{T, F\}$ that satisfied X

Then each clause i has a literal l_{ij} such that $a(l_{ij}) = T$

Let $I = \{v_{ij} : 1 \le i \le m\}$

Then $|I| = B$, I is independe since

$a(x) = a(y) = T$ for $x, y \in I$, so $(x,y) \notin Z$

there I is an inde set of graph, ie $(V, Z, B) \in IND$

step 4. $f(x) = (V, Z, B) \in IND \Rightarrow X \in 3\text{-SAT}$

let I be and ind set of size $B = m$ of (V, Z) Clearly each triangle contains exactly one vertex in I (is I is independent) let these vertices be $I = \{v_{ij}, 1 \le i \le m\}$

Assign True to l_{ij},

and True to remaind, unassigned vertices.

Note that this is a consistent truth assignment (ie. we did not assign true to x_i and $\neg x_i$)

x_i —— $\neg x_i$

Further, this truth assignment make every clause T so it makes x true. Independly, $x \in 3\text{-SAT}$.

exam

5-COLORING

CLIQUE

INPUT : undirected graph (V, E)
positive integer B

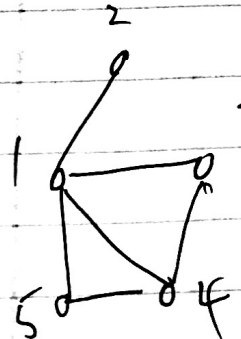
output . yes, if there exists $C \subseteq V$

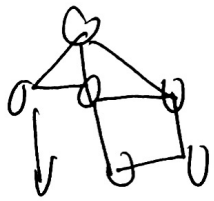
$|C| = B$ such that

$x \in C$

$y \in C \Rightarrow (x,y) \in E$

$x \neq y$





Theorm : CL2QUZ → NP-C

Proof :

Step1. 1) CL2QUZ ∈ NP

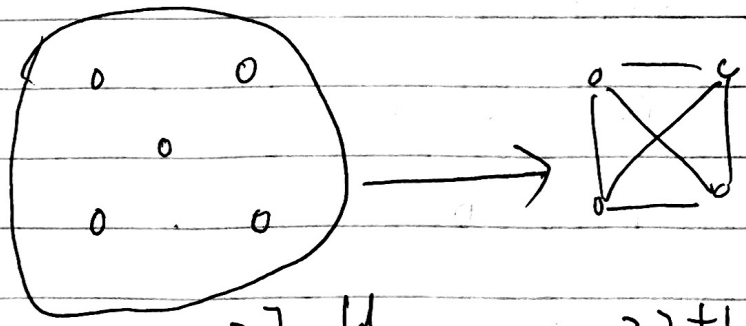
Solution. C a subset of V, |C|=B

verify-CL2QUZ-SQ (V,Z,B,C)

{ if |C|!=B
return false;
for x∈C
$O(|V|^2|Z|)$ for y∈C
if (x≠y && (v,y)≠Z)
$=O(|V|^3)$ return false
}
return FALSE.
}

know new

Step2 2) IND ≤ CL1QUZ



27 odd 27+1 even?

(V,Z,B) ⟶ (V,Z,B)

reduction

$f(V, E, B)$

{ $E' = \emptyset$

for $x \in V$

for $y \in V$

$O(|V|^2 |E|)$ → 可 $(x,y) \notin E$

$E' = E' \cup \{(x,y)\}$

} return (V, E', B)

3) $(V, E, B) \in IND \Rightarrow (V, E', B) \in CLIQ$

suppose (V, E) has an ind set

I of size B

$x \in I$, $y \in I$, $x \neq y$ $\Rightarrow$ $(x,y) \notin E$

$\Rightarrow (x,y) \in E'$

$\Rightarrow I$ is a clique of (V, E')

of size B

4) $(V, E', B) \in CLIQ \Rightarrow (V, E, B) \in IND$

suppose (V, Z') has a clique C of size B

$x \in C$
$y \in C \Rightarrow (x, y) \in Z'$
$x \neq y$

$\Rightarrow (x, y) \notin Z$

$\Rightarrow C$ is an independent of (V, Z) of size B