JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

Algorithms

Week 3rd - 14th Day

THURSDAY 14 JANUARY

1) sort (v.begin(), v.end()); → theoretical element after last element
first iterator — last iterator

sort (first, last) (open interval)

2) reverse (fi, li)

3) * max_element (fi, li), * min_element
↳ to find max element
without *, pointer to the max element

4) count (fi, li, x)
count the occurence of x in vector
fi → first iterator
li → last iterator

5) find (fi, li, x)
points to v.end() if element not found else returns pointer to the first element = x

6) binary_search (fi, li, x); (returns 1 if found)
vector should be sorted

7) lower_bound (fi, li, x) (array must be sorted)
return pointer to first element in (first, last) with value ≥ x

8) upper_bound (li, ri, x):
. . . . . > x (strictly greater)

Important

eg) V → {1, 2, 10, 10, 5, 7, 30}
it1 it2

auto it1 = lower_bound (v.begin(), v.end(), 10);
auto it2 = upper_bound (" , " , 10);

it2 - it1 = 3 (three 10)
it1 - v.begin() = 2 (index of it1)

2016

FEB MAR APR

FRIDAY
15
JANUARY

Week 3rd - 15th Day

DECEMBER

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

9) v.erase (v.begin() + 1)
to delete 2nd element

v.erase (unique (v.begin(), v.end()), v.end());
deletes duplicate elements in a sorted vector

10) distance (first_element, desired_position)
↳ returns the distance of desired position from first.

eg: distance (v.begin(), max_element (v.begin(), v.end()));
(returns the index of maximum element

11) next_permutation (fi, li) 1,2,3 → 1,3,2 → 2,1,3 → 2,3,1 → 3,1,2 → 3,2,1
prev_permutation (fi, li)

12) all_of() , any_of(), none_of()
v.begin() v.end()

eg: any_of (ar, ar+6, [] (int x) { return x<0 ; }) ?
cout << "There exists a -ve element" ; :
cout << " All are +ve elements";

13) Stable_partition (v.begin(), v.end(), [] (int x) { return (x%2 == 0); });
2,1,5,6,8,4 → 2,6,8,4,1,5,7
(returns iterator to the beginning of elements where condition is false.) ↑ it

14) search (fi, li, f2, l2);
Important
searches for subsequence [first2, last2) in [first1, last1)
returns iterator to the first element of occurrence
or last1 if no occurrence is found.

15) issorted (v.begin(), v.end())
returns true if sorted in ascending order

2016

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

Vectors :

Week 3rd - 16th Day

SATURDAY
16
JANUARY

Functions :

begin() , end() , rbegin() , rend()
↳ theoretical element before first elems

size(), v1.empty() == true, at(), front(), back(),
push_back(), pop_back(), clear(), v1.swap(v2);
↳ removes all elements

insert (position (iterator), value)

or v2.insert (v2.begin() , v1.begin() , v1.end())
(inserting at beginning of v2 elements in range v1.begin to v1.end)

erase (position) or erase (first position , last position)
↳ iterators

eg: vector <int> v1 { 1, 2, 3, 4, 5, 6, 7, 8, 9 }

i) for (auto i = v1.begin() ; i != v1.end() ; i++) // 1, 3, 5, 7, 9
{ if (i%2 == 0) { v1.erase (i) ; i--; } }

ii) v1.erase (v1.begin() + 2 , v1.begin() + 5)
// 1, 2, 6, 7, 9
↳ not included in [ ) SUNDAY 17

2D Vector

cin>>r>>c;
vector <vector <int>> v (r, vector<int> (c));
↳ 2D vector of r rows and c columns

Important

FEB
MAR
APR

2016

MONDAY
18
JANUARY

List → non contiguous location
(doubly linked list)

Week 4th - 18th Day

DECEMBER

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Functions: front(), bak(), begin(), end()
push_front(), push_back(), pop_front(), pop_bak
empty(), insert(), erase(), reverse(), size(),
mylist.sort(), list.remove(val);

Dequ ≈ list
(doubly ended queue)

Stack

size(), empty(), top(), push(g), pop() → O(1)

void display (stack<int> s)
{ while (!stempty()) stack<int> st = s;
{ cout << stop();
stpop(); }}

Priority Queue

empty(), size(), top(), push(), pop(), swap(),
returns first element (greatest) — maintain order (descending) — from begin

priority_queue<int> p;
p.push(10);
20, 20, 5, 1
// 30, 20, 10, 5, 1
to display, similar to stack & queue

Important

2016

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

Set

Week 4th - 19th Day

TUESDAY
19
JANUARY

↳ each element is unique
↳ stored in ascending order
↳ elements cannot be modified directly (use iterator to access element)

Functions :

begin(), end(), size(), empty(), clear(), swap()

9 i) s1.insert(value); // log N — N → no. of elements in the set

10 ii) itr = s1.insert(itr, 4);
11 itr = s1.insert(itr, 2); — searching begins at position itr to insert the element

iii) s2.insert(s1.find(3), s1.end());

12 s1 = {1, 2, 3, 4, 5}
s2 = {3, 4, 5}

s1.erase(position) — not in vector
1 s1.erase(it, it) = operands.
erase(const g)
find(const g), count(const g) ⇒ 1 or 0 only
set s2 = s1;
2

Multiset → multiple elements can have same value (order is maintained)

Unordered set → randomize insertion (no order)

Unordered multiset → store duplicate elements also

Important

multiset<int, greater<int>> s2; // sorted in descending order

for(auto x : s1) // short hand
cout<<x;

2016

WEDNESDAY
20
JANUARY

Map

Week 4th · 20th Day

DECEMBER

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

↳ Key-value pairs (unique key values)
↳ sorted in ascending order of keys

Functions: begin(), end(), size(), empty(), erase (iterator), erase (key), swap(), clear(), insert(), count(key K) → 0 or 1 (unique key) erase (it1, it2)

eg. map <int, int> m1;
• m1.insert (pair<int, int> (1, 40));
map <int, int> :: iterator it;

for (it = m1.begin(); it != m1.end(); it++)
{ cout << it->first << it->second;

• map <int, int> m2 (m1.begin(), m1.end()); // m2 = m1
• m2.insert ({2, 20});
auto it2 = m1.find (2);
• m1.insert (it2, {3, 60}); // inserts before element with key 2

[] operator → map <int, string> m1;
m1[1] = "Hi";
m1[2] = "Alex";
cout << m1[2]; // O(log N)

Important

Multimap → multiple elements can have same keys

Unordered map → map with random order // O(1) on avg
Unordered multimap → multimap + unordered map (duplicate & no order) // for map O(log N)

for (auto x : m1)
cout << x.first << x.second;
↳ operator (if pointer is not null)

2016

JANUARY

M	T	W	T	F	S	S
			1	2	3	
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

Pair

Week 4th - 21st Day

THURSDAY
21
JANUARY

```
pair <int, int> p1;
p1.first = 1; p1.second = 'A';
pair p2 = make_pair (2, 'B');
cout p2.second; // B
```

Operators : =, ==, !=, >=, <=, swap()

```
eg: vector < pair <int, int> > v;
for (i=0; i<n; i++)
    { v.push_back ( make_pair (x,y) ) }
    sort ( v.begin (), v.end() );
for (i=0; i<n; i++)
    { cout << v[i].first << v[i].second;
```

5 10
5 20
10 5
20 10
20 30
30 60

Map Application : frequency array

```
string s = "geeksforgeeks";
unordered_map < char, int > m;
for (i=0; i<s.size(); i++)
    { m [ s[i] ]++; }
for (auto x : m) { cout << x.first << x.second; }
```

Output : O(n)

```
01
f1
r1
g2
e4
k2
s2
```

Important

2016

FEB
MAR
APR

FRIDAY
22
JANUARY

Strings

Week 4th - 22nd Day

DECEMBER

M	T	W	T	F	S	S
		1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

i) getline (cin, str); (reads a line including spaces)
cin >> str; (reads until space is encountered)

ii) str.pushback ('s');
str.popback();
=, >, <, ⊕ operator (concatenation)

iii) str.resize(13) → change size of string
length() / size()
shrink_to_fit()
capacity() (capacity → size)

iv) iterator f^n : begin(), end(), rbegin(), rend()

v) str1.swap(str2);
str.clear() // empties the string
front(), back(), empty() (whether string is empty)
char * charstr = str1.c_str(); // string → null terminated char array

vi) str1.find(str2), rfind() (searching from end)
↳ returns index where string 2 is found in string 1
if pattern not found returns -1.

vii) str.substring(a,b)
returns substring of length b starting from index a.

Important viii) str.erase(a,b)
deletes b characters starting from index a.
or str.erase(str.begin(), str.begin()+5)

ix) ~~str.erase~~ str.replace(a, b, "hello"); str2
replace b characters from index a by "hello"

2016

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

SATURDAY
30
JANUARY

Week 5th - 30th Day

x) str . insert (pos, "hello") → str2
↳ index ↳ string to insert

9

10

11

12

1

2

SUNDAY 31

Important

2016

WEDNESDAY
03
FEBRUARY

TRICKS

Week 6th - 34th Day

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

```
#include <bits/stdc++.h>

i) Checking even or odd:
   if (num & 1) cout<< "ODD"; else even

ii) division by 2: n = n >>1 ; mult: n = n <<1

iii) Xor swap:  a ^=b; b ^=a; a ^=b;

iv) emplace_back() is faster than push_back

v) __gcd(x,y) → inbuilt

vi) max size of array:
   in main() → 10^6        in global → 10^7

vii) No. of digits in a no.n = floor(log10(N))+1;

viii) To check if a no. is a power of 2:
   bool ispowertwo(int x)
   { return(x && !(x&(x-1))); }

ix) faster input/output ; // can't use printf/scanf
   main()
   { ios_base::sync_with_stdio(false);
     cin.tie(NULL);
   }

x) auto a = 100;        // a will become int
   auto d = "hello";    // a will become string
```

x:1000
x:0111
first x-1
for x=0

Important

2016

char s[] = "Hello"; int x = 2048;

xi) int to string : string s = to_string(x);
string to int : int y = stoi(s);
= atoi(s2); // C type string

xii) vector<int> v = {0, 1, 2, 3, 4}
for (auto & value : v)
cout << value << " "; } // if we not used & then changes not reflects
// 0 2 4 6 8

xiii) swap(), __builtin_popcount(x), __builtin_clz(x)

viii) Precision of floating point nos:
floor(), ceil(), round(), trunc()
<= x <= x >= x -> removes decimal
float pi = 3.14159;
cout << fixed << precision(3) << pi << endl; // 3.142
(2) // 3.14
(5) // 3.14159

ix) Comparing floats :
if (abs(a-b) < 1e-9) cout << "nos are equal"
// (0.3*3 + 0.1) != 1 (internal precision error)

Important

FRIDAY
05
FEBRUARY

Mathematics

Week 6th - 36th Day

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
				1	2	3
8	9	10	11	12	13	
15	16	17	18	19	20	
22	23	24	25	26	27	
29						

i) **Sieve of eratosthenes:**

```
void sieve (int n)
{ bool isprime [1e6];
  for (i=0; i<=n; i++)
      isprime [i] = true;
  isprime [0] = isprime [1] = false;
  for (i=2; i*i <= n; i++)
      { if (isprime [i])
          { for (j = i*i; j<=n; j = j+i)
              isprime [j] = false; }
      }
}
```

ii) **Modulo arithmetic:**

$(a \pm b) \% c = ((a \% c) \pm (b \% c)) \% c$

$(a - b) \% c = ((a \% c) - (b \% c) + c) \% c$

Generally $c = 1e9 + 7$

iii) **Modulo exponentiation:**

$(a \hat{} b) \% m$

```
ll expo (ll a, ll b, ll m)
{ if (b == 0) return 1;
  ll p = expo (a, b/2, m) % m;
  p = (p * p) % m;
  return (b % 2 == 0) ? p : (a * p) % m;
}
```

eg: factorial (n) % m n is large

```
unsigned long long factorial (int n)
{ unsigned ll f = 1;
  for (i=1; i<=n; i++)
      f = (f * i) % m;
  return f; }
```

Important

2016

Similarly $(a+b+c)\%m = (((a+b)\%m)+c)\%m$
(perform %m after each step)

$-5\%3 = -2$ // actually $\in [0 \text{ to } n-1]$

⇒ int mod (int a, int m)
return $(a\%m + m)\%m$

Modulo multiplicative inverse & MMI

MMI of int a is an int x such that
$ax \equiv 1 \pmod m$
or $ax \%m = 1$

$x \in \{0 \text{ to } m-1\}$

$z = (x/y)\%m$ &
$y\_inv = MMI(y, M)$
$z = (x * y\_inv)\%M$

MMI of a modulo m exists iff $\gcd(a, m) = 1$.

Fermat's little theorem:

if p is a prime no., a is any integer, then

SUNDAY 7

$a^p \equiv a \pmod p$
or
$(a^p - a)\%p = 0$ or
$a^p - a$ is an integer multiple of p

Important

if a and p are coprime // $\gcd(a,p) = 1$
then, $a^{p-1} \equiv 1 \pmod p$ // $a^{p-1}\%p = 1$

$a^{-1} \equiv a^{p-2} \pmod p$

$a^{-1} = a^{p-2}\%p$

$a(a^{p-1}-1) = kp$
$\Rightarrow a^{p-1} - 1 = kp$
(a does not divide p)

2016

MONDAY
08
FEBRUARY

Week 7th - 39th Day

JANUARY						
M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY						
M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

Congruence:

$n > 1$, two integers are said to be congruent modulo n if n is a divisor of their difference:

$a - b = kn$

$a \equiv b \pmod{n}$

$\Rightarrow a = kn + b$ (k may be $-ve$)

* a and b have the same remainder when divided by n

$a = pn + r$
$b = qn + r$
$\Rightarrow a - b = kn$

eg: $38 \equiv 14 \pmod{12}$ $\rightarrow 38 - 14 = 24 \mid 12$ ✓
$-3 \equiv 7 \pmod{5}$ $(-8\%5 = 7\%5 = 2)$
$2 \equiv -3 \pmod{5}$
$-3 \equiv -8 \pmod{5}$

* equivalence relation:
reflexive: $a \equiv a \pmod{n}$
symmetric: $a \equiv b \pmod{n} \Rightarrow b \equiv a \pmod{n}$
transitive: if $a \equiv b \pmod{n}$ & $b \equiv c \pmod{n}$
$\Rightarrow a \equiv c \pmod{n}$

Important * If $a_1 \equiv b_1 \pmod{n}$, $a_2 \equiv b_2 \pmod{n}$, $a \equiv b \pmod{n}$

$a \pm K \equiv b \pm K \pmod{n}$ $(K \in Z)$

$a_1 \pm a_2 \equiv b_1 \pm b_2 \pmod{n}$

$a^K \equiv b^K \pmod{n}$ $(K \in Z^+)$

2016

a if $Ka \equiv Kb \pmod n$
then $a \equiv b \pmod n$ if $\gcd(K, n) = 1$

* Wilson's theorem:
if p is a prime no. $\Leftrightarrow (p-1)! \equiv -1 \pmod p$
or $(p-1)! \equiv p-1 \pmod p$
$\Rightarrow (p-1)! + 1 = Kp$

9
10
11 ${}^nC_r$: $\frac{n!}{r!(n-r)!} = \frac{n(n-1)(n-2)\cdots[n-(r-1)]}{r(r-1)(r-2)\cdots 1} = \frac{n(n-1)(n-2)\cdots[n-(r-1)]}{1 \cdot 2 \cdot 3 \cdots r}$

12 i) binomial coeff (int n, int k)
{ int res = 1;
1 if (K > (n-K)) K = n-K; // ${}^nC_K = {}^nC_{n-K}$
2 for (i=0; i<K; i++) // O(K) time
{ res = res * (n-i); O(1) span
res = res / (i+1);
}
return res;
}

ii) ${}^nC_r = {}^{n-1}C_{r-1} + {}^{n-1}C_r$

Important code:
int dp[100000][100000] = {0};
int bincoeff (int n, int k)
{ if (dp[n][k] != 0) return dp[n][k];
if (k == 0 || k == n)
dp[n][k] = 1; return dp[n][k];
dp[n][k] = bincoeff(n-1, k-1) + bincoeff(n-1, k);
return dp[n][k];
}

WEDNESDAY
10
FEBRUARY

Week 7th - 41st Day

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

modified sieve // prime[i] contains largest prime factor of i.

define size 1e7+1

vector<int> prime(size); (global so initialized to 0)

void sieve();

```
for (i=2; i<=size ; i++)
{ if (prime[i] != 0) continue;
  else
  { prime[i] = i;
    for (j = 2*i ; j<= size ; j = j+i)
    { prime[j] = i ;
    }
  }
}
```

Prime factorization of n: map<int, int> m ;

```
int n; cin >> n;
while (n != 1)
{ m[prime[n]]++;
  n = n / prime[n];
}
for (auto x : m)
cout << x.first << x.second ;
```

eg: n = 84

2 2
3 1
7 1

$(2^2 \times 3^1 \times 7^1)$

Important

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

MARCH

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Week 7th - 42nd Day

THURSDAY
11
FEBRUARY

Eulers theorem : $x^{\phi(m)} \% m = 1$

$\phi(m)$ → no. of numbers btw 1 and m which are coprime to m.
Euler totient func

eg: $\phi(12) = 4$ (1, 5, 7, 11 are coprime to 12)

$\phi(m) = \prod_{i=1}^{k} p_i^{\alpha_i - 1}(p_i - 1)$

$12 = 3 \times 2^2$ $\phi(12) = 3^0(3-1) + 2^1(2-1)$
$= 4$

If m is prime $\phi(m) = m^0(m-1) = m-1$

∴ If m is prime, $x^{m-1} \% m = 1$ → fermat's th

Important

2016

FRIDAY
12
FEBRUARY

Week 7th - 43rd Day

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

Sort ()

```
bool sortbysec ( pair<int,int> &a, pair<int,int> &b)
{ return (a.second < b.second); }
        ↓
// if condition is true then a comes before b in sorted seq.

vector <pair<int,int>> v ;
sort (v.begin(), v.end(), sortbysec);
```

Important

Divide & Conquer

Master theorem : Big O of recursion algorithm

If $T(n) = aT\left(\left[\frac{n}{b}\right]\right) + O(n^d)$

$a > 0$, $b > 1$, $d \geq 0$, then

$$T(n) = \begin{cases} O(n^d) & \text{if } d > \log_b a \\ O(n^d \log n) & \text{if } d = \log_b a \\ O(n^{\log_b a}) & \text{if } d < \log_b a \end{cases}$$

eg: Merge sort : $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

$\log_2 2 = 1 = d = 1$

$\Rightarrow T(n) = O(n^1 \log n)$

Binary search : $T(n) = T\left(\frac{n}{2}\right) + O(1)$

$\log_2 1 = 0 = d$ $\quad O(n^0)$

$\Rightarrow T(n) = O(n^0 \log n) = O(\log N)$

Important

MONDAY
15
FEBRUARY

Hashing

Week 8th - 46th Day

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

↳ unordered_set and unordered_map
  ↳ implemented through hash table
  ↳ O(1) average time complexity for find(), erase(), insert()
    O(n) in worst case

map, set → implemented through self balancing binary search tree
  ↳ O(log N) for find(), erase(), insert()
    as order is maintained

eg1:

```
unordered_set <int> s;
for(i=0; i<n; i++)
    s.insert(v1[i]);
for(i=0; i<m; i++)
{   if(s.find(v2[i]) != s.end())
    {   s2.insert(v2[i]);
    }
}
```

// to find common elements of two vectors v1 and v2
// unordered_set<int> s2;
// Now s2 contains intersection of both vectors
// O(n+m) on avg.

eg2:

Important

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

MARCH

M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

$12 = 3 \times 2^2$

$12\left(1-\frac{1}{3}\right)\left(1-\frac{1}{2}\right)$

$12 \times \frac{2}{3} \times \frac{1}{2} = 4$

Week 8th - 47th Day

TUESDAY 16 FEBRUARY

Mathematics

$\phi(n) = n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\cdots\left(1-\frac{1}{p_n}\right)$

* Euler's totient function: $\varphi(n) = \prod_{i=1}^{k} p_i^{(d_i-1)}(p_i-1)$

eg: $\phi(12) = 2^{(2-1)} \cdot 3^{(3-1)} = 4$

$\phi(n)$ → (no. of coprime no.s to n, b/w 1 and n)

* For prime no.s, $\phi(n) = n-1$

Euler's th^m: $x^{\phi(m)} \bmod m = 1$

where x and m are coprime

* If m is prime then $x^{m-1} \bmod m = 1$

$a^{p-1} \equiv 1 \pmod p$

or $a^p \equiv a \pmod p$

$a^{\phi(n)} \equiv 1 \pmod n$

$a^n \equiv a^{n \bmod \phi(m)} \bmod n$

$\phi(n) = n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\cdots\left(1-\frac{1}{p_n}\right)$

Modular Inverse:

$x x^{-1} \equiv 1 \pmod m$

$x^{-1} \equiv x^{\phi(m)-1}$

Important

eg: $6^{-1} \bmod 17 = 6^{17-2} \,\%\, 17$

$= 6^{15} \,\%\, 17$

$= 3$

$6^{16} \equiv 1 \pmod{17}$

$6^{16-1} \equiv 1^8 \pmod{17}$

$6^{15} \equiv 3 \pmod{17}$

(6, 17 are coprime)

$\left(\frac{36}{6}\right) \% 17 = \left(36 \cdot 6^{-1}\right) \% 17 = (36 \times 3) \% 17$

$= (2 \times 3) \% 17$

$= 6$

MAR

APR

2016

WEDNESDAY
17
FEBRUARY

Week 8th - 48th Day

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

$$\therefore \left(\frac{a}{b}\right) \% m = (a \cdot b^{-1}) \% m$$

Extended Euclid's Algo:

$ax + by = \gcd(a, b)$ $\quad x, y, a, b \in \mathbb{Z}$

(we can always find pair (x, y) for a and b)

↳ To find modular inverse using extended euclids Algo

Let $b = m$

$ax + my = 1$ (m, a are coprime for modular inverse to exist)

Taking mod both sides

$ax + my \equiv 1 \pmod m$

$ax \equiv 1 \pmod m$

$x = a^{-1}$ → found using the algo

Chinese Remainder Theorem:

Important

$x \equiv a_1 \pmod{m_1}$

$x \equiv a_2 \pmod{m_2}$

$\vdots$

$x \equiv a_n \pmod{m_n}$

$X_k = \frac{m_1 m_2 \cdots m_n}{m_k}$

2016

Then $x \equiv \sum a_k X_k X_{k_{m_k}}^{-1} \pmod{m_1, m_2 \cdots m_n}$

where, $X_K X_K^{-1}{}_{m_K} \equiv 1 \pmod{m_K}$

Explain: $x \bmod m_1 = a_1 X_1 X_1^{-1} \bmod m_1$ (all other term contain m_1)

$= a_1$ ($X_1 X_1^{-1} \bmod m_1 = 1$)

eg: $x \equiv 3 \pmod 5$
$x \equiv 4 \pmod 7$
$x \equiv 2 \pmod 3$

$\therefore x = 3 \cdot 21 \cdot 1 + 4 \cdot 15 \cdot 1 + 2 \cdot 35 \cdot 2$

$= 263$

$\Rightarrow x = 23 + 105k \quad k \in Z$

Euler totient function: $n = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$

$\phi(n)$ = no. of coprimes from 1 to n coprime to n

$= n \left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right)$

$\phi(p) = p - 1$ ($p\left(1 - \frac{1}{p}\right) = p - 1$)

$\phi(p^k) = p^k - p^{k-1}$ where p is prime

Important

$\phi(a,b) = \phi(a) \cdot \phi(b)$ if $\gcd(a,b) = 1$

Code:

```
phi[i] = i  ∀ i from 1 to N
for (i = 2 ; i <= N ; i++)
  { if (isprim[i]) {
      for (j = i ; j <= N ; j += i)
        phi[j] -= phi[j] / i ;
```

2016

FRIDAY
19
FEBRUARY

Binomial Coefficients

Week 8th - 50th Day

JANUARY

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

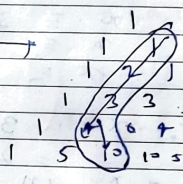
FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

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

$\sum_{i=0}^{n} i\binom{n}{i} = n2^{n-1}$

Hockey Stick formula: $\sum_{m=k}^{n} \binom{m}{k} = \binom{n+1}{k+1}$



$\sum_{i=0}^{k} \binom{n}{i}\binom{m}{k-i} = \binom{n+m}{k}$ } Vandermonde's Identity

$\downarrow$

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

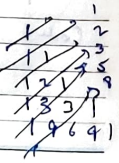
$n_1 + n_2 + n_3 + \cdots + n_r = n$

Important

$0 \le n_i \le r$

$\binom{n+r-1}{r-1}$

$\sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-1}{k} = Fib(n+1)$



2016

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

MARCH

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Week 8th - 51st Day

SATURDAY
20
FEBRUARY

Lucas Theorem :

$\binom{n}{m} \bmod p$ for N large, p small prime

$$\binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor} \cdot \binom{n \% p}{m \% p} \bmod p$$

↳ recursively till small enough

Catalan Numbers :

C_n : 1, 1, 2, 5, 14, 42

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

SUNDAY 21

Recurrence: $C_{n+1} = \sum_{i=0}^{n} C_i \, C_{n-i}$ for $n \geq 0$

$C_0 = 1$

Important

↳ No. of paths from $(0,0)$ to (N,N) not going above the diagonal

$$\binom{2n}{n} - \binom{2n}{n+1} = \frac{1}{n+1}\binom{2n}{n}$$

→ 2n length balanced parenthesis (no. of)

2016

MONDAY
22
FEBRUARY

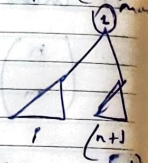
Week 9th - 53rd Day

JANUARY
M T W T F S S
1 2 3
4 5 6 7 8 9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

FEBRUARY
M T W T F S S
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29

= No. of binary trees with n nodes = C_n

↳ $f(n+1) = \sum_{i=0}^{n} f(i)\, f(n-i)$.



= No. of triangulation ways for a convex polygon of $(n+2)$ sides = C_n

3, 4

~~$f(n+2) = f(n) + f(n-1) f(n)$~~

~~(n+2 sides)~~ ~~+ f(n)~~

~~∴ for prime 1 ~ N ~ $\sum_{\log_2 N}$~~

~~(b-a) log log (1-a) f(0) f(n) + f(1) f~~

= No. of ways to connect 2n points on circle so that no two intersect

Important

016

HAVELLS

FEBRUARY

M	T	W	T	F	S	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29						

MARCH

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

$\lfloor \frac{n}{i} \rfloor = \lfloor \frac{n}{x} \rfloor$ last value of $x = \lfloor \frac{n}{\lfloor \frac{n}{i} \rfloor} \rfloor$ $\left(\lfloor \frac{n}{k} \rfloor\right)$ for i from 1 to n ($2\sqrt{n}$ distinct values)

Week 9th - 54th Day

TUESDAY

23 FEBRUARY

Fibonacci Numbers

$F_n = F_{n-1} + F_{n-2} = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$

$= \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$

To calculate nth fibonacci no. when $n \le 10^{18}$

$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix}$ ✓✓ use matrix exponentiation $O(\log n)$

$\left(\begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix}\right)$ $n \le 10^{18}$

Properties:

i) $\sum_{i=0}^{n} F_i = F_{n+2} - 1$

ii) $\sum_{i=0}^{n} F_i^2 = F_n \cdot F_{n+1}$

iii) $F_n = F_m \cdot F_{n-m+1} + F_{m-1} F_{n-m}$

Important

iv) $F_n \mid F_{km}$

or $F_n \mid F_m \Rightarrow n \mid m$; $n > 2$

v) $\gcd(F_n, F_m) = F_{\gcd(n,m)}$

2016

MAR

APR

CS-204

Algorithms

Time Complexity:

Insertion sort :

* Best case : elements already sorted → $\theta(n)$
* Worst case : $\frac{n(n+1)}{2}$ → $\theta(n^2)$
* Avg case → $\theta(n^2)$

$\theta(g(n)) = f(n)$, $\exists c_1, c_2 > 0, n_0 \geq 0, \forall n > n_0$

$$c_1 g(n) \leq f(n) \leq c_2 g(n)$$

$\frac{1}{2}n^2 - 3n = \theta(n^2) = 3n^2 + 7n + 6$

$O(g(n)) = f(n)$: $f(n) \leq c\, g(n)$

(upper bound)

$f(n) = \theta(g(n)) \Rightarrow f(n) = O(g(n))$

$O(n^2) = 7n^2 + 5n - 2$

Also $an + b = O(n^2)$ (loose upper bound but still true)

$\Omega(g(n)) = f(n)$: $f(n) \geq c\, g(n) \geq 0$

(lower bound)

$f(n) = \theta(g(n)) \Rightarrow f(n) = \Omega(g(n))$

$7n^2 + 2 = \Omega(n^2)$

Also, $an^2 + bn + c = \Omega(n^2)$

o - Notation:

$o(g(n)) : \forall c > 0, \exists n_0 > 0 : \forall n > n_0 \quad f(n) < c\,g(n)$

strict inequality

loose upper bound

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

eg: $2n = o(n^2)$ but $2n^2 \neq o(n^2)$

ω - Notation

$\omega(g(n)) = f(n) : f(n) \gg c\,g(n)$

loose lower bound

$2n^2 = \omega(n)$ but $2n^2 \neq \omega(n^2)$

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

$f(n) = O(g(n))$, $\Omega(g(n))$, $\Theta(g(n))$

→ transition

↺ reflexive

Θ → symmetric $\quad f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$

$f(n) = \underset{o}{\Theta}(g(n)) \Leftrightarrow g(n) = \underset{\omega}{\Omega}(f(n))$

$f(n) = O(g(n))$ & $f(n) = \Omega(g(n)) \Leftrightarrow f(n) = \Theta(g(n))$

Master's theorem:

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

$$T(n) = \begin{cases} d > \log_b a \longrightarrow \Theta(n^d) & \text{(weight decreases down the tree)} \\ d < \log_b a \longrightarrow \Theta(n^{\log_b a}) & \text{(weight increases geometrically)} \\ d = \log_b a \longrightarrow \Theta(n^{\log_b a} \log n) & \text{weight is same at each level} \end{cases}$$

Sorting

Heap: almost complete binary tree
Max heap: parent (x) $\geq$ x
implemented through array

Operations:

i) max_heapify (A, i, n) → restores the max heap property
condition: left subtree and right subtree of index i are max heap
↳ O(log n) (take max (child nodes), then swap)

ii) build_max_heap (A, n) → create max heap from unordered array
for (i from n/2 to 1) max_heapify (A, i, n);
↳ O(n ~~log n~~)

iii) heapsort (A, n)
↳ first build max heap
↳ then sort (swap root with last then heapify discard the last node)
↳ $\Theta(n \log n)$

Priority Queue:

↳ key with highest value/priority extracted first
↳ implemented using max heap.

Operations:

i) Heap_maximum (A) — returns largest element → O(1)

ii) Heap_extract_max (A, n) → returns the max value & removes that element from heap
↳ O(log N)

iii) Heap_increase_key (A, i, key) → increases the key of an element
↳ O(log N)

iv) max_heap_insert (A, key, n) → inserts new element in max heap
↳ O(log n)

* Stable sort: not suitable for if array contains -ve integers
Insertion, merge, counting O(n), radix O(d*n), bucket ↳ O(n) ($i^{32} \approx 1$)

Unstable: quick, heap

Efficiency: Merge sort > Quick sort > heap sort
additional storage req.

* Linear selection: worst case → $O(n^2)$
expected / avg → $O(n)$

DP → overlapping subproblems
→ optimal substructure

i) Matrix multiplication → $O(n^3)$

$$m[i,j] = \begin{cases} 0 & i = j \\ \min_{i \le k < j} [m[i,k] + m[k+1,j] + p_{i-1} p_k p_j] & i < j \end{cases}$$

ii) Rod cutting → $O(n^2)$

iii) Longest Common Subsequence: $O(mn)$

$$c[i,j] = \begin{cases} 0 & i=0 \text{ or } j=0 \\ c[i-1,j-1] + 1 & x_i = y_j \\ \max(c[i,j-1], c[i-1,j]) & x_i \neq y_j \end{cases}$$

iv) Optimal BST: → $O(n^3)$
$1 + \sum \text{depth}(k_i) \cdot p_i$ → minimize

Greedy → Optimal substructure
greedy property

GRAPH

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

* $|E| = |V| - 1 \rightarrow$ tree
If Graph is connected, 然 $|E| \geq |V| - 1$

* Adjacency List → storage → $O(V+E)$
Adjacency matrix → $O(V^2)$

BFS :

```
vector <int> adj[n];
queue <int> Q;
Q.push(s);                    // source node        visited[s] = true;
while (!Q.empty())                                  dist[s] = 0;
{   x = q.front(); q.pop()
    for (auto v : adj[x])
    {   if (!visited[v])
            visited[v] = true;
            dist[v] = dist[x] + 1;
            q.push(v);
    }
}
```

* Runtime → $O(V+E)$
* If u is enqueued before v then $u.d \leq v.d$
* dist[v] in bfs = shortest path distance from s to v.

DFS:

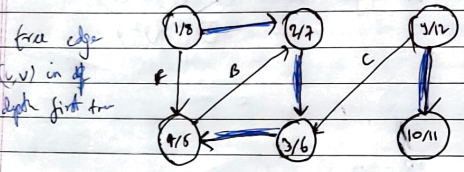
```
void dfs(int s)
{   if (visited[s]) return;
    visited[s] = true;
    for (auto v : adj[s])
        dfs(v);
}
```

* Running time = $O(V+E)$

* v is a proper descendant of u iff

$$d[u] < d[v] < f[v] < f[u]$$

d[v] : discovered time
f[v] : finish time



Back edge (B) → (u, v) : v is the descendant of u in the depth first tree

Forward edge (F) → (u,v) : v is the descendant of u but not a tree edge

Cross edge (C) : any other edge

* Undirected graph → only tree and back edge

Topological Sort :

* $(u,v) \in E \Rightarrow u$ must be done before v in the directed graph

ordering of vertices 1, 2, — , |V| :
for every edge (u,v),
u appears before v in ordering

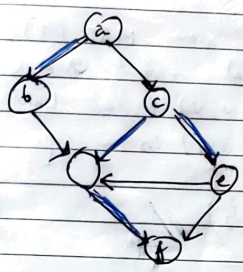
* There should not be any cycles (otherwise deadlock)
↳ DAG (directed acyclic graph)

∵ no cycle ⇒ no back edge

Algorithm:

i) Call DFS to compute $f[v]$ for each vertex v
ii) As a vertex is finished, insert in front of linked list
iii) return linked list

eg:



$a \to b \to c \to e \to d \to f \to \phi$
(decreasing order of finish time)

Run time: $O(V+E)$
dfs $\to O(V+E)$ in worst case
insertion in linked list $\to O(1)$

(u,v): $f[v] < f[u]$

Strongly Connected Component:
max set of vertices: $\forall\ u,v \in C$, $u \rightsquigarrow v$, $v \rightsquigarrow u$

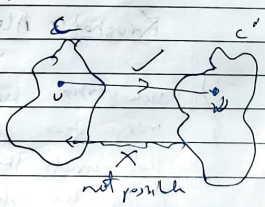
transpose of SCC $\to$ edge direction reversed
Creation of $G^T \to \Theta(V+E)$ (adjacency list)

Algo to determine SCCs:

i) call dfs(G) to compute $f[u]$ $\forall u$ // $O(V+E)$
ii) compute G^T // $O(V+E)$
iii) call dfs(G^T) considering vertices in the order of decreasing $f[u]$ // $O(V+E)$
iv) Output vertices of each tree formed (of the depth first forest) in second dfs as a separate SCC.

$\therefore$ $\boxed{O(V+E)}$

Thm: $f(c) > f(c')$ if there is a edge from c to c'.



not possible otherwise both will be counted

(if $f(c) > f(c')$ then there are no edges from c to c' in G^T)

Minimum Spanning Tree

↳ tree containing all vertices of the graph & minimum sum of weights
(tree has no cycles (|E| = |V| - 1))

Th^m: Let A be a subset of some MST (T) (S, V-S) be a cut that respects A (doesn't cut vertex of A) and (u, v) be a light edge (min weight), then (u, v) is a safe edge for A.

Prim's Algo: (greedy)

Q → min heap
contain vertices not yet included in the tree
stores vertices with weight key[v]
↓ vertex with
picks the minimum key[v], adds to the tree & then adjacent nodes key value is updated.
↳ (while ∈ Q)

```
while (Q ≠ ϕ)                                   // V
    do u = extract_min(Q)                       // log V
    for each v ∈ Adj[u]                         // E
        if (v ∈ Q && w(u,v) < key[v] // dec)
            {π[v] = u
            decrease_key (Q, v, w(u,v))}        // log V
```

Time: O(V log V + E log V) = O(E log V)

Kruskal's Algo: // greedy

i) each vertex being its own component
merge two components by choosing light edge that connects them
(scan the set of edges in increasing order of weight)

Algo:

$A \rightarrow \phi$

for each vertex $v \in V$ } $O(V)$
    make set (v)

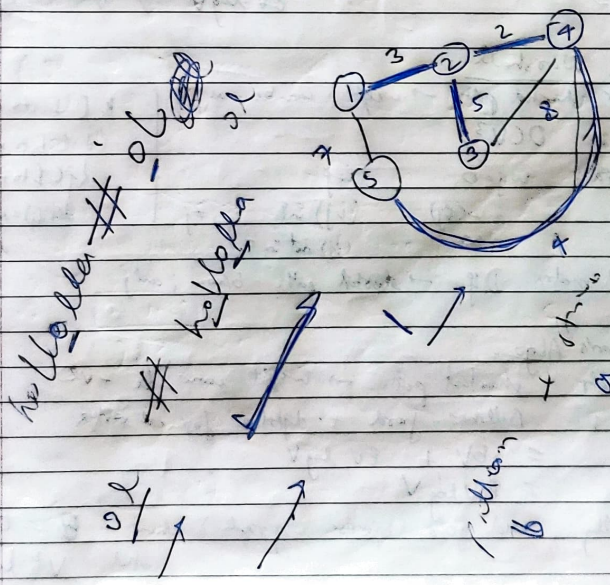
sort E in non-decreasing order by weight // $O(E \log E)$

for each (u,v) in sorted list // $O(E)$
    if (findset (u) ≠ findset (v)) // returns first element of set
        $A \rightarrow A \cup \{(u,v)\}$ } // $O(\log V)$
return A

Run time: $O(V + E \log E + E \log V) = O(E \log E)$

$(E = O(V^2) \Rightarrow \log E = 2 \log V)$

$\Rightarrow$ Run time $= O(E \log V)$ // same as Prim



Shortest Path :

i) Dijkstra :
- non-negative edges
- shortest path from source to all other nodes
- min heap / set
- Time complexity :
  - adjacency list : $E \log V$ $(V \log V + E \log V)$ (extract-min, decrease-key)
  - adjacency matrix : V^2

ii) Bellman ford Algo :
- can handle -ve edges
- same as dijkstra
- relax each edge $(n-1)$ times
- Time complexity : $EV \simeq V^3$ $(E = O(V))$

* All pair shortest path → run both algo for each vertex as source one by one
  $EV \log V$ / V^4

Floyd Warshall :
- all pair shortest path → weight can be -ve
- Time : $O(V^3)$

Initialize : $w_{ij} = 0$ $i = j$
$= w(ij)$ $(ij) \in E$ $\}$ $i \neq j$
$= \infty$ (ij) not in E

k (1 to n)
{i (1 to n)
{j (1 to n)
$d[i][j] = \min(d[i][j], d[i][k]+d[k][j])$

find matrix D_{ij} → shortest path b/w i and j

Johnson's Algo
- all pair shortest path → weight can be -ve
- Time : Bellman ford + dijkstra for each vertex
  $= EV + EV \log V$
  $= EV \log V$
- Generally used for sparse graph when $E = O(V)$
  so $V^2 \log V$

- $\hat{w}(u,v) = w(u,v) + h(u) - h(v)$
  ≥ 0
  run dijkstra for each vertex to obtain $\hat{\delta}(u,v)$ for all $v \in G.V$
- Now $d_{uv} = \hat{\delta}(u,v) + h(v) - h(u)$

$h(v) = \delta(s,v)$
s → dummy source connected to each vertex with weight 0

MAXIMUM FLOW

- While there is an augmenting path (path with all edge > 0) do augment flow.
  Total flow = sum of each flow
- Time complexity : DFS → $O(Ex)$
  x → max flow

  DFS → Edmonds Karp → $O(E^2V)$ (for augmenting path finding)
  DFS · (E · EV)
  1 1
  DFS Edmonds

* Max flow = minimum cut
- Push relabel → V^2E

$\delta(u)$ u ⇝ v

$\delta(v) \leq \delta(u) + 1$
$h(v) \leq h(u) + 1$