$n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}$

19/08/2017

CODE ISM

Codeforces — tutorial (DS)

\# Time complexity :-

```
for(i=1; i<=n; i++)
    for(j=1; j<=n; j+=i)
    {
        ——
        ——
        ——
    }
```

$O\left(n + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n}\right)$

$= O(n \log n)$

$O\left(n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}\right)\right)$

$= O\left(n \sum_{x=1}^{n} \frac{1}{x}\right)$

$= O\left(n \int_{1}^{n} \frac{1}{x} dx\right)$

$= O(n \log n)$

Page No.
Date: / /

#

$n \leq 10 \longrightarrow O(n!)$

$n \leq 20 \longrightarrow O(2^n)$

$n \leq 500 \longrightarrow O(n^3)$

$n \leq 5000 \longrightarrow O(n^2)$

$n \leq 10^6 \longrightarrow O(n \log n)$ or $O(n)$

- $n \leq 10^8 \longrightarrow O(n)$

$n \geq 10^9$ or large $\longrightarrow O(1)$ or $O(\log n)$

- Merge sort — $O(n \log_2 n)$
- Binary search — $O(\log_2 n)$
- long long int z = (1LL << 50);
- int x = 10^9, y = 10^9;

long long int z = 1LL * x * y;

- Prefer scanf over cin if input taken ~~is~~ slowly

else; ios_base::sync_with_stdio(false);
cin.tie(NULL);

- D.S — Youtube — NPTL
- MIT — algorithm

Algorithm

BOSS Page No. Date: / /

- char ch = getchar();
  or
  char ch = getchar_unlocked();
  ↓
  only for linux

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

Q:- spoj.com/problems/ID

spoj → NSTEPS
GIRLSNBS
CANDY 3
Codechef → PALIN

Q to Ask:- register
long int
long long int
long long

- "I am a student"
  string S;
  getline(cin, S);
  or ~~get~~ cin.getline(S);

?? { int v[10]
memset(, 5,); ⇒ int v[10];
O(n) ≡ memset(v, 5, 10)

Page No.
Date:

// initialize all elements by 5 either (0) or (-1).

D 314 Rahul Sir
D 345 Anupam Sir
D 231 Ayush Sir
D 306 Sauvik Saha
D 338 Harman Sir

Templates :-

```
class A{
template<typename>K
sort (K)
{
  ___
  ___
}
};

int main( )
{ ___char K[10];
  ___sort<char>K
  ___;
}
```

} generic function

10

Vectors :- #include<vector>

- Constructors :-

vector<long long int> v;

v(5); => size

T.C ≡ O(n) ≡ v(5, 1);

- (i) vector<vector<int>> v;

⇓

like 2-D array

(ii) vector<vector<int>> v(5);

⇓ fixed 5 rows

(iii) vector<int> a(6, 2);

vector<vector<int>> v(5, a);

⇓

5 rows with 6 columns each with initial value of 2.

- v.push_back(5);

T.C ≡ O(1).

- v.begin(); } returns iterator
  v.end(); } T.C ≡ O(1)

- v.size(); T.C ≡ O(1)
- v.resize(n, 5); T.C ≡ O(n)
  ⇓
  increases ~~as well as~~ or decreases size to n and make new elements equal to 5.
- v.empty(); ⇒ returns bool
  T.C ≡ O(1)
- v.clear(); ⇒ empties the vector
  T.C ≡ O(n)
- v.erase( );

  v.erase(⇓ , ⇓);
  iterator of start & end
- vector<int>::iterator t;

  t = ve.begin();
  cout << *t;

Sort :- #include <algorithm>

- sort(v.begin(), v.end());

  T.C ≡ O(n log n)

  array
  vector
  list
  string

  struct v{=}; v a;

- sort(a, a+n, compare);

```
bool compare(v x, v y)
{


}
```

if you want x before return true

if you want y before return false.

map

map<type1, type2> m;

E.g:- map <char, int> m; // automatically sorts the map according to index.

```
m['a'] = 2;
m['b'] = 6;
m['c'];          ⟶ default value is 0.
m[make_pair(2,4)] = 6;
```

- ```
  for(auto it: m)
  {
      cout<<*it->first<<" "<<*it->second
                                  <<"\n";
  }
  ```

```
for(auto it=m.begin(); it!=m.end(); it++)
```

- insertion & deletion & accessing $O(\log(n))$?? (check)
```

unordered_map<int, int> um;

insertion & deletion & accessing } O(1) (But O(n) in worst case)

set :- // stores only distinct elements

- set<int> s;
- s.insert(5);
- s.erase(); ⇓ iterator or value

insertion, deletion & accessing } O(log n) // check.

- it = s.find(5); ⇓ returns iterator to 5 or returns s.end();

unordered_set<int> s;

insertion, deletion, accessing } O(1) (But O(n) in worst case)

String :-

- string S = "abc";
- S+= "de";
- if (S1 == S2) ⇒ returns bool
  >=
  <
- string S, S1 = "abcdefg";
  S = S1.substr(4,3); // S = "efg"
  S1.substr(3); // S = "defg" till end.

Stack :-

- vector <int> B = {1,2,3};
  stack <~~type~~ int> A = ;
  ~~A~~
  A(B);
- A.size();
  A.top();
  A.pop();
  A.empty(); ⇒ returns bool.

Queue :-

- queue <int> a;
  vector <int> b = {1, 3, 7};
  a(b);

- a.front(); ≡ O(1)
- a.push(10); ~~≡ O(1)~~
- a.empty(); ⇒ returns bool
- a.size();
- a.pop(); ≡ O(1)

Dequeue :-

- deque <int> a;
  vector <int> b = {1, 7, 7};
  a(b);
- a.push_front();
  a.push_back();
- a.pop_front();
  a.pop_back();

BOSS
Page No.
Date: / /

pair :-

- pair<int, int> p;
- pair<char, int> p;

pair<pair<int, char>, int> p;

- pair<int, int> p1={2,5};

pair<int, int> p1 = make_pair(2,5);

- p1.first // 2
  p1.second // 5

- pair<int, int> p[n];

sort(p, p+n);
// sort according to 1 element & then according to 2 element.

priority_queue<int> q;
// sort according to descending order

priority_queue<int, vector<int>, greater<int>> or smaller<int> or compare to func

Q:- Spoj :-

~~FACEFRND~~

~~RKS~~

~~HOMO~~

codechef :- ~~KGOOD~~

27/Aug/2017

Number Theory

Modular Arithmatic

1. $(a+b) \% m = [(a\%m) + (b\%m)] \% m$
2. $(a-b) \% m = [m + (a\%m) - (b\%m)] \% m$
3. $(a*b) \% m = [(a\%m) * (b\%m)] \% m$.
4. How to calculate x^n in $O(\log n)$

$$x^n = \begin{cases} 1 & \text{if } n == 0 \\ x^{n/2} * x^{n/2} & \text{if } n\%2 == 0 \\ x * x^{n/2} * x^{n/2} & \text{if } n\%2 == 1 \end{cases}$$

or iterative :- (include it in template)

```
pow(x, n)
{
    int res = 1;
    while (n > 0)
    {
        if (n & 1)
            res = res * x;   ==> res = (res * x) % m;
        n >>= 1;
        x = x * x;           ==> x = (x * x) % m;
    }
    return res;
}
```

x = 10 ; n = 5

101

res = 1, 10, 100000

x = x % m; n = 10, 1, 0

x = 100

x = 10000

$x = 10^8$

$5^2 \% 3 = 1 \quad \underline{1 \pmod m}$

#

Modular multiplicative inverse

- $aX \equiv 1 \pmod m$

  ↓

  M.M.I of a

  $X \in \{1, 2, 3, \ldots, m-1\}$

  X only exists when $\boxed{\text{gcd}(a, m) = 1}$

- Fermat's Little theorem :-

$(a^{m-1} \% m) \equiv 1 \pmod m$

when m is prime. and $\text{gcd}(a, m) = 1$

$\therefore$ M.M.I of a is a^{m-2}. when m is prime.

Note :- $\left(\frac{a}{b}\right) \% m \Rightarrow$ where m is prime.

$= \left[(a \% m) * (b^{-1} \% m)\right] \% m.$

⇓

M.M.I of b

gcd (int a, int b) // Euclidian Algorithm.

```
{
  if (b==0)
     return a;
  else
     return (b, a%b);
}
```

$O(\log[\min(a,b)])$.

$^{n}C_{r}$

(i) Recursion:-

$$^{n}C_{r} = {}^{n-1}C_{r-1} + {}^{n-1}C_{r}$$

Base case:-

$$^{0}C_{0} = 1 \; ; \; {}^{a}C_{b} = 0 \; (a<b) \; ; \; {}^{n}C_{0} = 1$$

(ii) $$^{n}C_{r} = \frac{n}{r} \times {}^{n-1}C_{r-1}$$.

$n = 10_8$ - $p = 3$

$r = 5$

$n = 1 \times 3^2 + 0 \times 3^1 + 1 \times 3^0$

$r = 0 \times 3^2 + 1 \times 3^1 + 2 \times 3^0$

** Lucas theorem :-

$$\binom{n}{r} \% p$$

$$n = n_k p^k + n_{k-1} p^{k-1} + \ldots\ldots + n_2 p^2 + n_1 p^1 + n_0 p^0$$

$$r = r_k p^k + r_{k-1} p^{k-1} + \ldots\ldots + r_1 p^1 + r_0 p^0$$

$$\therefore \binom{n}{r} \% p = \left(\binom{n_k}{r_k} \% p\right) * \left(\binom{n_{k-1}}{r_{k-1}} \% p\right) * \ldots\ldots * \left(\binom{n_0}{r_0} \% p\right).$$

$$O\left(\log_p \min(n, r)\right)$$

bool prime[10e6];

Sieve of Eratosthenes (int n)

```
{
memset(prime, true, sizeof(prime));

for(p=3; p*p<=n; p+=2)
{
    if(prime[p]== true)
    for(i=p*p; i<=n; i+= 2*p)
    {
        prime[p]= false;
    }
}
}
```

$O(n \log(\log n))$ #.

Q:-
- ZSUM —
- LCPCP2 —
- PRIMES2 —
- ~~TDK~~
- TDKPRIME —
- PRIME1 —

} SPOJ

- COOK82B — Codechef
- Project Euler → ϕ-20.

Smallest prime factor

```
int pri[10^6];
void spf()
{ for (int i=1; i<=10^6; i++)
        pri[i]=i;

  for (int j=i*i; j<=10^6; j+=i)
        pri[j]=i;
}
```

Segmented Sieve :-

```
vector<int> prime_root_n;

void query(int l, int r)
{
    bool arr[r-l+1]={};

    for(i=0; i<prime.size(); i++)
    {
        int j;
        if((l%prime[i])==0 && l!=prime[i])
            j=l;
```

else $j = \left(\frac{l}{prime[i]} + 1\right) * prime[i];$

for (; j<=n; j+= prime[i])

~~prime~~ arr[j] = false;

}

Euler totient function :-

$\Phi(n)$ = Number from 1 to n whose gcd with n is 1.

- $\phi(p) = p-1$; where p is prime.
- $\phi(a \times b) = \phi(a) * \phi(b)$

  where $gcd(a,b) = 1$.
- $\phi(p^l) = p^l - p^{l-1}$

  $= p^{l-1}(p-1)$

where p is prime.
l is any positive integer.

60
15 5

210
105 35

Page No.
Date: / /

Let $n = p_1^{l_1} \times p_2^{l_2} \times p_3^{l_3} \times \ldots\ldots$

$\phi(n) = \phi(p_1^{l_1}) \times \phi(p_2^{l_2}) \times \ldots\ldots$

$= p_1^{l_1} \times p_2^{l_2} \times p_3^{l_3} \times \left(1 - \frac{1}{p_1}\right) \times \left(1 - \frac{1}{p_2}\right) \times \ldots\ldots$

$= n \times \left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \ldots\ldots$

```
int etf(n)
{
    int result = n;
    for(int p=2; p*p<=n; p++)
    {
        if (n%p==0)
        ~~while~~
        {
            while (n%p==0)
                n = n/p;
            result -= result/p;
        }
    }
    if (n>1)
        result -= result/n;
    return result;
}
```

HSOQ EQ ETF GCDEX POWPOW2
ETFS LCMSUM PRIME
CEQU POUR DCEPEAOB

Etf till n :-

```
int phi[1000001];
void phical()
{
    for(int i=1; i<=1000000; i++)
        phi[i] = i;
    for(int i=2; i<=1000000; i++)
    {
        if(phi[i]==i)
        {
            for(int j=i; j<=1000000; j+=i)
            {
                phi[j] = phi[j]/i * (i-1);
            }
        }
    }
}
```

Note:- (i) $gcd(a,b) = c$

d is a common factor of a & b.

then $gcd\left(\frac{a}{d}, \frac{b}{d}\right) = \frac{c}{d}$.

$\sum_{d/n}$ = set of divisor of n.

(ii) $gcd(a, n-a) = gcd(a, n)$.

***

Note:- There are $\phi\left(\frac{n}{d}\right)$ possible values of i (1 – n) for which we get $gcd(i, n) = d$.

$\Downarrow$

here $(n \% d = 0)$

*** $\sum_{i=1}^{n-1} \frac{n}{gcd(i,n)}$

Note:- $\sum_{d/n} \phi(d) = n$.

- Euler's theorem (RSA)

$$\begin{cases} a^{\phi(n)} \\ a^{\phi(n)} \% n \equiv 1 \pmod{n} \end{cases} \longrightarrow \text{where } gcd(a,n) = 1$$

- $a^{b^c} \% m = \left(a^{(b^c) \bmod \phi(m)}\right) \% m$.

stable_sort 6×6=36

Extended Eucledian :-

```
int egcd(int a, int b, int *x, int *y)
{
    if (a==0)
    {
        *x = 0;
        *y = 1;
        return b;
    }
    else
    {
        int x1, y1;
        int gcd = egcd(b%a, a, &x1, &y1);
        *x = y1 - (b/a) * x1;
        *y = x1;
        return gcd;
    }
}
```

$ax + by = gcd(a,b)$

$(b\%a)x_1 + ay_1 = gcd(b\%a, a)$

$(b - a(b/a))x_1 + ay_1 = gcd(a,b)$

$a\left(y_1 - \left(\frac{b}{a}\right)x_1\right) + bx_1 = gcd(a,b)$

Codeforces — COMDIV — Codechef

633A — PTIME — Vinat and GCD

778A — ETFD

GCD2

Binary Search — O(log n)

```
int lo = 0, hi = n-1;
while (lo <= hi)
{
    int mid = (lo+hi)/2;
    if (arr[mid] == key)
    {
        return mid;
    }
    if (arr[mid] > key)
        hi = mid-1;
    else
        lo = mid+1;
}
```

- b_s (lo, hi, P) FFFFF(T)TT

```
{
    while (lo < hi)
    {
        mid = lo + (hi-lo)/2;
        if (p[mid] == true)
            hi = mid
        else
            lo = mid+1;
    }
```

FFFF(T)TT
TOPCODER - Binary Search — fair work load
Hackearth - Aman Goyel Binary Search

```
    if( p[lo] == false)
        complain...

    return lo;
}
```

- b-s(lo, hi, P) FFFF(F)TTT

```
{
    while (lo < hi)
    {
        mid = lo + (hi - lo + 1)/2;
        if (p[mid] == true)
            hi = mid - 1;
        else
            lo = mid;
    }
    if (p[lo] == true)
        complain...

    return lo;
}
```

Sajan sajja => sub sequence
sija => sub string/array

1101

lower_bound
upper_bound

equal_range => returns a pair of iterators.
binary_search

Bit-masking :- O(n.2^n)

```
for(i=0; i<(1<<n); i++)
{
    for(j=0; j<n; j++)
    {
        if(i & (1<<j))
            cout<<a[j]<<" ";
    }
    cout<<"\n";
}
```

__builtin_popc

Codechef PPXOR | Data Science Tutorials Topcoder | SPOJ ONOTOLE NEEDS YOUR HELP

CNOTOLR

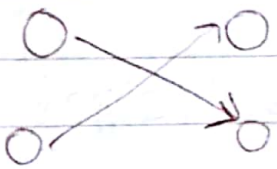
$-X = \sim X + 1$

$\Rightarrow -(X+1) = \sim X$

$\Rightarrow X + 1 = -(\sim X)$.

Graph — Representation
BFS
DFS

Graph Theory



DAG

Tree — every edge is connected but no cyclic path.

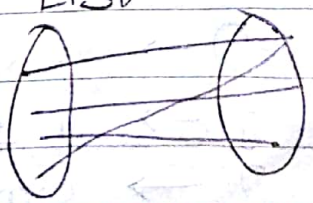
- maximally acyclic — adding of even one edge make it cyclic.

- minimally connected

Representation :-

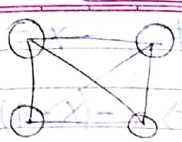
(i) Adjacency Matrix

(ii) Vector <= Adjacency List



Bipartile graph

Complete Graph:-



Note:- Number of Odd degree vertex are even.

In a graph there must be atleast 2 ~~edges~~ vertex with same degree.

DFS ⇒ Depth First Search

```
void DFS(int s)
{
    int j;
    for(j=0; j<G[s].size(); j++)
    {
        if(!VG[s][j])
        {
            VG[s][j] = true;
            DFS(G[s][j]);
        }
    }
    return;
}
```

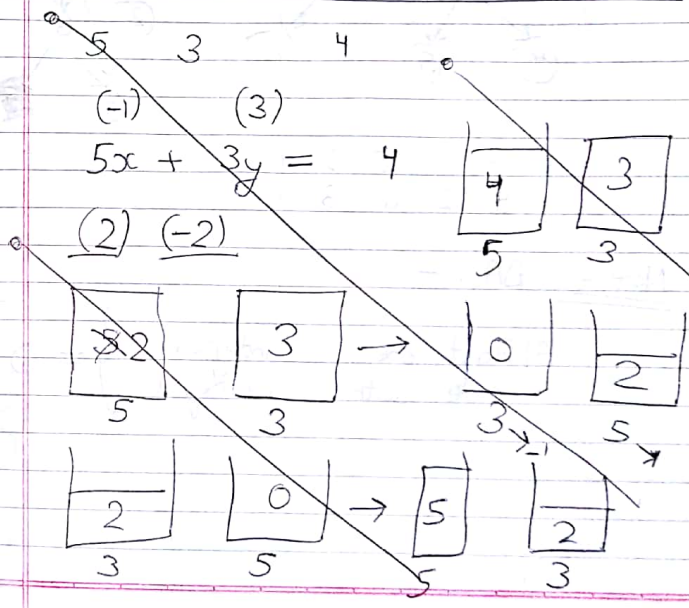
Note:- 0 → unvisited
1 → in stack
2 → completely visited.

Forward edge → 0—1 1—2
Backward edge → 1—1
Tree edge → 1—0

Q:- DFS / BFS Problem :-

POURI
PRIME PATH
Labyrinth ✓

Is it a tree?
Longest path in a tree. ✓

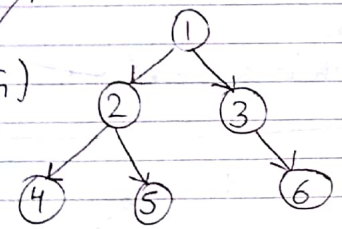


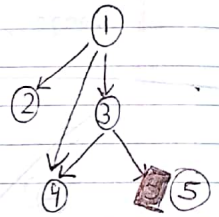
Time Complexity of DFS & BFS
$= O(V+E)$

Time Complexity of traversal in Tree
$= O(V+V-1)$
$= O(V)$

Topological Sorting :-

(DAG)





6 3 4 5 2 1
4 5 2 6 3 1

Note:- DAG :-

At least one node with { indegree / outdegree } = 0.

```
vector<int> ans;
queue q
for (i=1 to n)
   if (indegree[i]==0)
     {q.push(i);
      ans.push_back(i);}
while (!q.empty())
{
   u = q.top();
   q.pop();
   for (node v : adj[u])
   {
      indegree[v]--;
      if (indegree[v]==0)
      {q.push_back(v);
      ans.push_back(v);}
   }
}
```

Note:-
- Strongly connected = if we can go from node i to node j then we can also go from j to i.
- Strongly connected graph = all pairs of node are strongly connected.

Hackerearth

Kosaraju Algorithm :-

(1) Call DFS(a) and store finishing time of each vertex

(2). $G \rightarrow G^{rev}$

(3) Call DFS(G^{rev}) in decreasing order of finishing time.

Minimum spanning tree / Minimal :-

Spanning tree → some edges are removed from a graph so that it becomes tree.

parent[ ];

for (i=1 to) // while active set is not empty.

{

u = pop from active set having min key & include u in mst set ← prim's Algo for MST

for (each node v adjacent to u which are not in MST)

{

if (key[v] > ~~key[u]+~~ w(u,v))

{ key[v] = ~~key[u]+~~ w(u,v); parent[v] = u;

} active.push_back(v);

O(E log V)

Topological Sorting — lexicographically smallest.

- priority_queue<int, vector<int>, greater<int>>

Shortest path :-

- If DAG — then shortest path from a given source to all nodes using topological sorting — O(V+E)
- Graph with positive weight cycle and positive edges then Dijkstra.
- Graph with with positive weight cycle and not necessary positive edges then Bellman Ford.

S => source

Dijkstra:- $O(E \log V)$

```
dis[ ]
π[ ]

for (i=1; i<=V; i++)
    dis[i] = ∞;
    π[i] = ϕ;

dis[s] = 0;

set < pair <int,int> > ω;

ω.insert ( make_pair(0,s) );

while ( ! ω.empty() )
{
    pair <int,int> f = *(ω.begin());

    int y = f.second;
    if (d[y] == ∞) break;
for (i=1; i< Adj[y].size(); i++)
{
    int u = Adj[y][i];

    if (dis[u] > dis[y] + ω(y,u))
    {
        if(dis[u] != ∞)
            ω.erase (ω.find(make_pair(u,dis[u])));

        dis[u] = dis[y] + ω(u,v);

        π[u] = y;

        ω.insert (u, dis[u]);
    }
}
}
```

DISJOINT SET UNION — POSTERS

DIJIKSTRA — SHPATH, ~~SHAPTH~~, TSHPATH

SCC — CAPACITY, BOTTOM

MST — CSTREET, HIGHWAYS, BLINNET

spoj

Note:- $G \equiv (V, E)$
$G' \equiv (V', E')$

If $V' \subseteq V$ & $E' \subseteq E$ Then; $G' \subseteq G$

If $(V' == V)$ then spanning graph

Union of Disjoint Set :-

Priority_Queue

Kruskal ~~Algo. for in twing~~ MST :-

edge weight

```
pair<int, int pair<int,int>> p[e];

sort(p, p+e);
vector <pair<int,int>> ans;
for(i=0; i<e; i++)
{
    int u= p.second.first;
    int v= p.second.second;

    int root_u=root(u);
    int root_v=root(v);

    if (root_u != root_v)
    {
        ans.push_back(make_pair(u,v));
        union_find(u,v);
    }
}
```

A2OJ — coding site

codeforces.com/blog/entry/1622 ⇒ graph

↓

13529 ⇒ links

SPOJ => BUGLIFE MIT - 4 videos - DP

hackerearth => dp → intro → very easy

30

DP :-

Recursion → top-down

Iterative → bottom-up

Time Complexity of Recursion

```
Fibbo (int n)
{
    ========
    ========
}
```

weight

T.C = Tot. no. of ~~stack~~ states * O(weight)

= O(n) x O(1)

= O(n) ≡ fibonacci

Q :-

~~int kpath (int i, int j, int n)~~

f

1 → 1

2 → 5

3 → 8

4 → 9

5 → 10

6 → 17

7 → 17

8 → 20

$O(\phi^n)$ $x^2 - x - 1 = 0$ pseudo polynomial

Longest increasing sub-sequence :-

Longest common subsequence

0-1 Knapsack = O(n s)

n → no. of items, s → total weight

Subset sum

~~for (i=0; i<=S; i++)~~

{

~~for (j=0; j<n; j++)~~

{

~~if (i==0) (j==0)~~

Matrix Chain Multiplication ~~O(n^3)~~

Coin Change

• Spoj - Edit Distance (EDIST)

Edit Distance (Make string S1 to string2)

• Spoj - BABTWR

Box Stacking

- Two Robots – HackerRank
- Tourist – spoj

State-Space Reduction

↳ Reduces Timecomplexity as well as Space complexity

- Moody-Numbers — HackerEarth.
- Snake Ladder – HackerRank

Bellman Ford

$$DP^K(v) = \min_{u \in \text{vertices}} (DP^{K-1}(u) + w(u,v))$$

$$DP^K(u,v) = \min(DP^{K-1}(u,K) + DP^{K-1}(K,v),\ DP^{K-1}(u,v))$$

- $DP(K,v) = \min_{(adj\ v)} (dp(K-1,u) + wt(u-v))$

dp(K-1,v);

Floyd Warshall's algorithm:-

```
for K = 1 to n
    for i = 1 to n
        for j = 1 to n
            Dp(i,j) = min(DP(i,K) + DP(K,j), DP(i,j));
```

T.C ≡ $O(n^3)$
S.C = $O(n^2)$

H.W

- Bellman Ford & Floyd Warshall path print:

KQUERY ~~KQUERY2~~

Range Queries:-

(1). Segment Tree
(2). Fenwick Tree

Lazy Propogation:-

```
update(i,S,e,l,r,v)
{
    if(lazy[i] != 0)
    {
        tree[i] += (e-S+1) * lazy[i];
        lazy[
```

Article Codechef [MERGESORT TREE

- DQUERY
- KGSS
- BRCKTS
- POSTERS
- HORRIBLE
- LITE
- SEQSQRSS
- FLIPCOIN
- RENT

After Blog [FREQUENT ORDERSET

↓

- Ayush Jain - Segment trees
- Kartik Kukreja - segment tree
- Codeforces / entry / 18051, 15890, 13529
- ~~Spoj~~ => Count inversion
- Spoj => ASSIGN
- ~~LARRA~~
- LARMY #
- Inverse Count
- SUBLEX #

10010
10100

Catline
Bell
Stirling
} Numbers

BIT (Fenwick Tree) ⇒ can be updated
Sparse Table. ⇒ no update is possible.
SQRT Decomposition.

String Processing :-

Rabin Karp (Hashing) :-

```
d=1;
h=0;
for(i=m-1; i>=0; i--)
{
    h= (h+ (S[i]*d)%m)%m;
    d= (d*b)%m;
}
```

KMP :-

LCP ⇒ longest common proper prefix which is also a suffix.

a	b	c	d	a	b	c	d
0	0	0	0	1	2	3	4

P#T
pattern to be matched

Zee
Suffix Array

[_][_]
1, 2, 4

Fast Matrix Exponentation :-

$f_n = 1 \cdot f_{n-1} + 1 \cdot f_{n-2}$
$f_{n-1} = 1 \cdot f_{n-1} + 0 \cdot f_{n-2}$

$$\begin{bmatrix} f_n \\ f_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} f_{n-1} \\ f_{n-2} \end{bmatrix}$$

F_n F_{n-1}

Base case :- $n=2$

$$F_n = C^{n-2} F_2$$

Q:- $F_n = 3F_{n-1} + 3F_{n-3} + 2F_{n-2} + 3$

$$\begin{bmatrix} F_n \\ F_{n-1} \\ F_{n-2} \\ 1 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 3 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{n-1} \\ F_{n-2} \\ F_{n-3} \\ 1 \end{bmatrix}$$

1111001

$r=0$ $r=(r+q)\%3$

$a=1$

$a=(a*2)\%3=2$

Euler's Function $\phi(m)$

$a^{\phi(m)} \% m = 1$ where $\gcd(a,m)=1$

Chinese Remainder Theorem (CRT):-

$x \% m = a$

$x \% n = b$

there exists a unique x in (0 — $m*n-1$) which satisfies both equations,

inclusive range

where $\gcd(m,n) = 1$.

Proof:- $x \% n = a$

$x \% m = b$

m terms $\equiv a, n+a, 2n+a, \ldots\ldots, (m-1)n+a$

$i \quad j$

$in+a = im+b$

$jn+a = jm+b$

$(i-j)n = (i-j)m$

since $\Downarrow$ $< m$ & $\gcd(m,n)=1$

$\therefore$ Not possible.

$x \% n_1 = r_1$

$x \% n_2 = r_2$

$\vdots$

$x \% n_k = r_k$

where all n_i are pairwise co-prime.

$P = n_1 \cdot n_2 \cdot n_3 \ldots n_k$

$$X = \sum_{i=1}^{k} \left[\left(\frac{P}{n_i}\right) \left[\left(\frac{P}{n_i}\right)^{-1} \bmod n_i \right] r_i \right] \% P$$

$\Downarrow$ from extended Euclidean

Note:-

- $\left[a^{\phi(m)}\right] \% m = 1$ where $\gcd(a,m)=1$
- $\left[a^b\right] \% m = \left[a^{(b \% \phi(m))}\right] \% m$ where $\gcd(a,m)=1$.

d | e ≡ d is a divisor of e ~~Mobel's Function~~ Inclusion & ~~Exclusion~~

$\left(\frac{2n!}{n!\ n!}\right) \% p^k$

$\left(\frac{p^a \times c_1}{p^b \times c_2 \cdot p^b \times c_2}\right) \% p^k$

$\left(p^{a-2b} \cdot c_1 \cdot inv(c_2) \cdot inv(c_2)\right) \% p^k$

Tqie ~~Ariesta~~
Bit-masking DP ~~Walmart~~
DP ~~Amazon~~

Mobins Function & Inversion :-

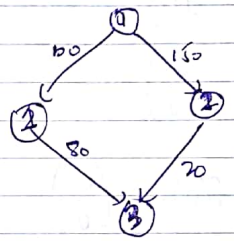
$$\mu[i] = \begin{cases} 1, & i=1 \\ 0, & \text{if } i \text{ is not square free} \\ -1, & \text{if } i \text{ has odd. no. of prime factors} \\ 1, & \text{if } i \text{ has even no. of prime factors} \end{cases}$$

$n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$

∴ not square free if $e_i > 1$ for some i
square free if $e_i = 1$ for all i.

$\mu[i] \equiv O(n \log n)$ Normally

$\mu[i] \equiv O(n)$ (using DP).



50
10
50

30 — 10
40 50

50
30 10
50

20 — 10
30 50
40

30 10
40 50

30 10
40 50

30 — 20
40 50

20 — 10
30 50
40

30 — ∞
40 50

30
40 20 → 20
50 → 50 — 30
40

20
40 30
50

20
30 — 50
40

Codechef ≡ coprime3

Application :-

If $g(n) = \sum_{d|n} f(d)$

Then $f(n) = \sum_{d|n} g(d) \cdot \mu\left(\frac{n}{d}\right)$

Proof :-

$$\sum_{d|n} g(d) \cdot \mu\left(\frac{n}{d}\right)$$

$$= \sum_{d|n} \sum_{e|d} f(e) \cdot \mu\left(\frac{n}{d}\right)$$

$$= \sum_{e|n} \left[f(e) \cdot \sum_{\substack{e|d \\ d|n}} \mu\left(n/d\right) \right]$$

$d \rightarrow n/d$

$$\sum_{e|n} \left[f(e) \sum_{e \,|\, (n/d)} \mu(d) \right]$$

$(n/d) \,|\, n$

$$= \sum_{e|n} \left[f(e) \cdot \sum_{d|(n/e)} \mu(d) \right] = f(n).$$

$$\sum_{d|x} \mu(d) = \begin{cases} 1 ; \text{ if } x = 1 \\ 0 ; \text{ if } x > 1 . \end{cases}$$

Q :- Coprime 3

$$G = \sum_{g=1}^{\text{MaxElement}} [g==1] \; \text{Count}(g)$$

Count(g) = no. of triplets having gcd = g

$$[n==1] = \sum_{d|n} f(d)$$

$$\Rightarrow f(n) = \sum_{d|n} [d==1] \cdot \mu\left(n/d\right)$$

$$= \mu(n/1) = \mu(n)$$

Now;

$$G = \left[\sum_{g=1}^{m} \left[\left(\sum_{d|g} \mu(d) \right) \text{count}(g) \right] \right]$$

$$= \sum_{d=1}^{m} \left(\mu(d) \cdot \sum_{d|g}^{m} \text{count}(g) \right)$$

$$= \sum_{i=1}^{m} \left(\mu(i) * {}^{d[i]}C_3 \right)$$

Dance with Mobins Function - Tutorial

Virat & GCD

$$G = \sum_{g=1}^{M} [g < k] \; count(g)$$

- $$[n < k] = \sum_{d|n} f(d)$$

$$f(n) = \sum_{d|n} [d < k] \cdot \mu\left(\frac{n}{d}\right)$$

```
for(i=1; i<k; i++)
{
    for(j=i; j<=M; j+=i)
        f(j) += u(j/i)
}
```

$$G = \sum_{g=1}^{M}$$

GCD Count (March Long)

- ~~$G = \sum_{g=1}^{M} [g$~~
- ~~$G = \sum_{K=1}^{M} [k == g] \; count(g)$~~
- $$G = \sum_{i=1}^{M} [gcd(i, g) == 1] \; count(i)$$
- $$[gcd(n, g) == 1] = \sum_{d|n} f(d)$$

$$f(n) = \sum_{d|n} [gcd(d, g) == 1] \mu\left(\frac{n}{d}\right)$$

Euler Path -

Starting a tour from a node, visiting each edge exactly once and then ending the tour at any node.

Directed graph —
Should be

(1). Connected graph with ignoring direction of edges.

(2). For all nodes out-degree should be equal to in-degree except at most one node with out_degree - in_degree = 1 and its corresponding one more node with in_degree - out_degree = 1.

Undirected graph –

(1). Graph should be connected.
(2). Degree for all nodes should be even ~~except atmost~~ ~~or~~ degree all nodes except two nodes should be even and the ~~two~~ other two should have odd degree.

Eules Circuit –

Starting a tour ~~at~~ from a node, visiting each edge exactly ~~one~~ once and then returning to the starting node.

Directed graph –

(1). ~~Graph should be a~~ All nodes should come under a strongly connected component.
(2). Out_degree = In_degree for all nodes.

Undirected graph –

(1). Graph should be connected.
(2). Degree of all nodes should be even.

Kth order Statistics

~~$T(n) = n - 3\left(\frac{n}{2} - 2\right)$~~

$$M = n - 3\left(\frac{1}{2}\left(\frac{n}{5}\right) - 2\right)$$

$$= n - 3\left(\frac{n}{10} - 2\right)$$

$$= \frac{7n}{10} + 6$$

$$T(n) = T\left(\frac{7n}{10} + 6\right) + T\left(\left\lceil \frac{n}{5} \right\rceil\right) + O(n)$$

$\Rightarrow T(n) = O(n)$

Prefix / Infix / Postfix

No. of unlabeled binary trees with n no. of nodes = N^{th} Catalan number = $\frac{(2n)!}{n!(n+1)!}$

No. of labeled binary trees with n nodes = N^{th} Catalan number * $n!$ = $\frac{2n!}{(n+1)!}$

Height of a red-black tree $\leq 2\log_2(n+1)$ where n is no. of nodes.

TSP :-

No. of subproblems = $O(n.2^n)$
Time complexity = $O(n^2.2^n)$

Hashing —

Collision Handling —

$O(1+\alpha)$
↖ (1). Chaining
$O\left(\frac{1}{1-\alpha}\right)$ ← (2). Open Addressing
(i) Linear Probing
(ii) Quadratic Probing
(iii) Double Hashing

# Load factor $\alpha = n/m$.
n = Numbers of keys to be inserted
m = Numbers of slots.

Time complexity to build a heap is $O(n)$ where n is no. of elements.

Maximum no. of nodes with height h $= \left\lceil \frac{n}{2^{h+1}} \right\rceil$

$$\therefore T(n) = O\left(\sum_{h=0}^{\log n} h \cdot \frac{n}{2^{h+1}}\right)$$

$$= O\left(\frac{n}{2}\sum_{h=0}^{\log n} \frac{h}{2^h}\right)$$

$$= O\left(\frac{n}{2} \cdot \sum_{h=0}^{\infty} h \cdot \left(\frac{1}{2}\right)^h\right)$$

$(x<1)$:- $\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$

Differentiating both sides :-

$$\sum_{n=0}^{\infty} n \cdot x^{n-1} = \frac{1}{(1-x)^2}$$

$$\Rightarrow \sum_{n=0}^{\infty} n \cdot x^n = \frac{x}{(1-x)^2}$$

$$\therefore \sum_{h=0}^{\infty} h \cdot \left(\frac{1}{2}\right)^h = \frac{\frac{1}{2}}{\left(1-\frac{1}{2}\right)^2} = 2$$

$\therefore T(n) = O(n)$.

Complete Binary Tree – all levels except possibly last are full. and if the last level is not full then all the nodes are as left as possible.

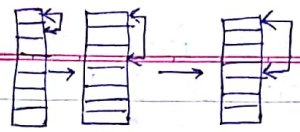
Full Binary Tree – each node has 2 children except leaf nodes.

B-Tree – It is a self balancing search tree. [t-2t]

(1). All leaves are at same level.
(2). A B-tree is defined by the term minimum degree 't'.
(3). All nodes except the root must contain atleast t-1 keys.
(4). Maximum possible no. of keys in a node can be 2t-1.
(5). No. of children of a node is one more than no. of keys.
(6). All keys of a node ~~is~~ are sorted in an increasing order.
(7). A child between two keys k1 & k2 contains all its keys in the range from k1 to k2.
(8). Time complexity is like other balanced binary search trees.

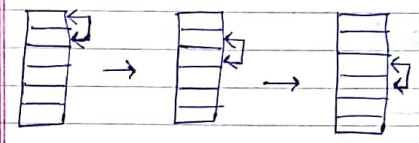
Multiway Search Tree – of order n has n-1 keys. and n children.

Selection Sort –

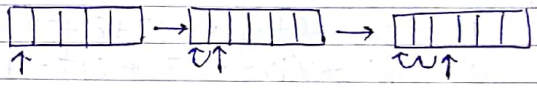


Comparison Sorting

Bubble Sort –



Insertion Sort –



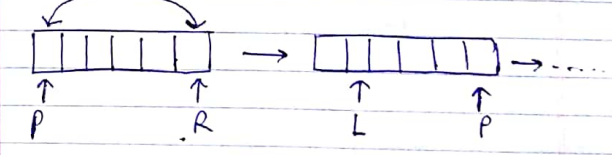
Bucket Sort – (like insertion sort)

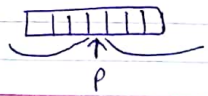
gap = $\lfloor n/2 \rfloor$

gap = $\lfloor gap/2 \rfloor$

till gap > 0

Quick Sort –





Non-comparison sorting

(1). Radix Sort — start from rightmost digit

(2). Bucket Sort — Divisor = Ceil$\left(\frac{Max+1}{No.\ of\ Buckets}\right)$

(3). Counting sort

Hashing —

(1). Folding K^2=07 89 23 41

(1). Pure folding = 07+89+23+41
(2). Fold shifting = 70+89+32+41
(3). Fold boundary = 70+89+23+14

(2). Mid square K = 1234
K^2 = 1522756
↓ ↓ ↓
525

(3). Division method
K ⟶ K % h

Valid postfix expression —

Rank	~~Oper~~
1	Operand
-1	Bin. Operator (+, -, *, /, %, ^)
0	Uni. Operator (!)

Sum of rank of all the characters in all valid prefix of a postfix expression must be greater than 0.
Sum of rank of all characters must be 1.

Infix to postfix

	Precedence
+, -	0
*, /, %	1
^	2
a, b, c	3

(1). Scan symbols one by one from left to right —

- while (stack == empty OR Pre(TOS) >= Pre(CS)
  POP(TOS) and Print
- PUSH (CS)
- If CS == '('
  PUSH (CS)
- If CS == ')'
  POP(TOS) while TOS ≠ '('

- At the end POP and Print all the symbols.

Infix to Prefix

- Reverse the infix expression and in that while reversing, make '(' to ')' and ')' to '('.
- Evalute Postfix
- Reverse evaluated expression

Postfix to Infix

- If it is operand push it in the stack.
- If it is operator pop top two elements, put operator inbetween them
  operand 2 operator operand 1
- Put the expression inside the parenthesis and again push it in the stack.

Prefix to Infix

- Process from right to left
- If it is ~~operator~~ operand push it in the stack.
- If it is operator pop top two elements and make string –
  '(' + operand 1 + operator + operand 2 + ')'
- Push the string st formed again at the top of the stack.

Postfix to Prefix conversion –

- If the symbol is an operand push it on the stack.
- If the symbol is an operator, pop top two elements from the stack and make string
  operator + operand 2 + operand 1
- Push the formed string back on the stack.

Prefix to Postfix conversion –

- Process the string from right to left.
- If the symbol is an operand push it on the stack.
- If the symbol is an operator, pop top two elements from the stack and make string –
  operand 1 + operand 2 + operator
- Push the formed string back on the stack.