2015

Algorithm Collections for Contest

A collection of some important algorithm for contest that might be helpful in 'On-site' programming contest.

CUET Black_Flags
Chittagong University Of Engineering & Technology
01-Jan-15

Contents

Math

❖ Geometry:

- Making a point data structure:

```
struct P
{
    double x,y;
    P(double x=0,double y=0){this->x=x;this->y=y;}
};
```

- Making a vector from two given points :

```
P MV(P a,P b){return P((b.x-a.x),(b.y-a.y));}
```

- Dot product:

```
double DP(P a,P b){return a.x*b.x+a.y*b.y;}
```

- Cross product:

```
double CP(P a,P b){return a.x*b.y-a.y*b.x;}
```

- Distance between two points:

```
double DIST(P a,P b)
{
     return sqrt(sqr(MV(a,b).x)+sqr(MV(a,b).y));
}
```

- Check if a point c is in middle of point a and b:

```
bool is_in_middle(P a,P b,P c)
{
    if(min(a.x,b.x)<=c.x&&c.x<=max(a.x,b.x)&&
       min(a.y,b.y)<=c.y&&c.y<=max(a.y,b.y))
        return true;
    return false;
}
```

- Check if tow segment collide with each other:

```
bool is_collided(segment a,segment b)
{
    double v1=CP(MV(a.a,a.b),MV(a.a,b.a));
    if(!v1&&is_in_middle(a.a,a.b,b.a)) return true;
    double v2=CP(MV(a.a,a.b),MV(a.a,b.b));
    if(!v2&&is_in_middle(a.a,a.b,b.b)) return true;
    double v3=CP(MV(b.a,b.b),MV(b.a,a.a));
    if(!v3&&is_in_middle(b.a,b.b,a.a)) return true;
    double v4=CP(MV(b.a,b.b),MV(b.a,a.b));
    if(!v4&&is_in_middle(b.a,b.b,a.b)) return true;
    if(v1*v2<0&&v3*v4<0) return true;
    return false;
}
```

- Comparison function for sorting points:

```
bool comp(P a,P b)
{
    double c=CP(MV(pivot,a),MV(pivot,b));
    if(c)
        return c>0;
     return sqrt((sqr(MV(pivot,a).x)+
     sqr(MV(pivot,a).y)))<sqrt((sqr(MV(pivot,b).x)+sqr(MV(p
     ivot,b).y)));
}
```

- Finding points that constitute a Convex Hall:

```
vector<P> convex_hall(vector<P> p)
{
    vector<P> s;
    s.push_back(p[0]);
    s.push_back(p[1]);
    s.push_back(p[2]);
    for(int i=3;i<p.size();i++)
    {
        P x=s.back();s.pop_back();
        P y=s.back();
        s.push_back(x);
        while(CP(MV(y,x),MV(y,p[i]))<0)
        {
            s.pop_back();
            if(s.size()>1){
               x=s.back();s.pop_back();
               y=s.back();s.push_back(x);
            }
            else{
               x=s.back();y=s.back();
            }
        }
        s.push_back(p[i]);
    }
    return s;
}
```

- Finding area of polygon if points are given:

```
double find_area(vector<P> p){
    p.push_back(p[0]);
    double ans=0;
    for(int i=0;i<p.size()-1;i++)
        ans+=0.5*(p[i].x*p[i+1].y-p[i+1].x*p[i].y);
    return ans;
}
```

- Projection of Vectors:
- Converting a vector into 'l' length vector:

❖ Number theory:

- CSOD:

Cumulative-sum of sum of divisors means $\sum_{i=1}^{n}\sum_{j=1}^{k} Dj$; $Where\ Dj\ is\ divisor\ of\ i$

```
ll CSOD(ll n)
{
    ll sum = 0,root = sqrt(n)+1;
    for(ll i=1;i<=root;i++)
    {
        ll temp = n/i; ///আমদের কত গুলা i আছে...
        sum += (temp*(temp+1))/2;
        /*
          আমি যেহেতু root পর্যন্ত লুপ চালাব সুতরাং root সংখ্যক i
          আমি নিয়ে নিব । সুতরাং এর চাইতে বেশি যত গুলা i আছে
          আমাকে নিতে হবে। এর চাইতে বেশি i আছে (temp-root)
          সংখ্যক ।
       */
        sum += max((ll)(0),(temp-root))*i;
    }
    return sum;
}
```

- Maximal algorithm for pre calculating prime in O(N) time:

```
/*
This algorithm can be used to calculate prime up to 10^7 in
O(N) time
```

```
(Actually a small time more than O(N) but it is
significantly lower..)
*/

#define SZ 10000010

ll lp[SZ];  //Contains lowest prime factor of i
vector<ll>prime;

void maximalAlgo(ll N)
{
    memset(lp,0,sizeof lp);
    for(ll i=2;i<=N;i++)
    {
        if(lp[i]==0)
        {
            lp[i] = i; //i is a prime number so it's lowest
prime factor is itself.
            prime.push_back(i);
        }
        for(ll j=0;j<prime.size()&&prime[j]<=lp[i]&&
                                        i*prime[j]<=N;j++)
            lp[i*prime[j]] = prime[j]; //Multiples of
prime[j] are composite and its lp is prime[j].
    }
}
```

- Catalan number formula:

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\,n!} = \prod_{k=2}^{n} \frac{n+k}{k} \qquad \text{for } n \geq 0.$$

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \frac{2(2n+1)}{n+2} C_n,$$

- Max Power of M that can divide N! :

আমরা যদি B বেস সিস্টেম এ N! এর ট্রেইলিং জিরো কয়টা আছে জানতে চাইতাহলে N! কে B এর সর্বচ্চ কত ঘাত দ্বারা নিঃশেষে ভাগ করা যায় সেটা জানাটাই যথেষ্ট। কারন যেকোনো নাম্বার সিস্টেম এ ওই সংখ্যার মান '10' এবং ১০ দ্বারা যেকোনো সংখ্যাকে গুণ করলে তার শেষে অবশই একটা জিরো থাকবে।

```
ll maxPower(ll m, ll n)
```

```
{
    ll ans = numeric_limits<ll>::max();
    for(ll i=0;i<prime.size()&&prime[i]<=m;i++)
    {
        ll temp = m,freq = 0, cnt = 0;

          while(temp%prime[i]==0)
             freq++,temp/=prime[i];

          if(!freq) continue; //If prime[i] is not a factor
     of m

          temp = n;
          while(temp>0)
            cnt+=temp/prime[i],temp/=prime[i];

          ans = min(ans,cnt/freq);
    }
    return ans;
}
```

- Euclidian GCD & LCM function:

```
int gcd(int a, int b){
    return b == 0 ? a : gcd(b, a % b);
}

int lcm(int a, int b){
    return (a / gcd(a, b)) * b;
}
```

- Extended Euclidian:

```
ll extended_euclid(ll a,ll b,ll &x,ll &y)
{
    if(b==0)
    {
        x = 1;
        y = 0;
        return a;
    }
    ll d = extended_euclid(b,a%b,y,x);
    y-=(a/b)*x;
    return d;
```

```
}
```

❖ Probability:

- Bay's theorem:

$$P(A|B) = \frac{P(A)\,P(B|A)}{P(B)},$$

$$P(B) = \sum_j P(B \mid A_j)P(A_j),$$

$$\Rightarrow P(A_i \mid B) = \frac{P(B \mid A_i)\,P(A_i)}{\sum_j P(B \mid A_j)\,P(A_j)}.$$

In the special case where *A* is a binary variable

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B \mid A)P(A) + P(B \mid \neg A)P(\neg A)}.$$

Graph:

- Segment tree:

```
#define SZ 100000

struct info
{
    ll prop,sum;
    info(ll p=0,ll s=0)
    {
        this->prop=p;
        this->sum=s;
    }
};

info tree[SZ*3];
ll arr[SZ];

void init(ll node,ll b,ll e)
{
    if(b==e)
```

```
    {
        tree[node]=info(0,arr[b]);
        return;
    }
    ll Left=node*2;
    ll Right=node*2+1;
    ll mid=(b+e)/2;
    init(Left,b,mid);
    init(Right,mid+1,e);
    tree[node].sum = tree[Left].sum+tree[Right].sum;
}

void update(ll node,ll b,ll e,ll i,ll j,ll x)
{
    if (i > e || j < b) return;

    if (b >= i && e <= j) //নোডের রেঞ্জ আপডেটের রেঞ্জের ভিতরে
    {
        tree[node].sum+=((e-b+1)*x); //নিচে নোড আছে e-b+1
টি, তাই e-b+1 বার x যোগ হবে এই রেঞ্জে
        tree[node].prop+=x; //নিচের নোডগুলোর সাথে x যোগ হবে
        return;
    }

    ll Left=node*2;
    ll Right=(node*2)+1;
    ll mid=(b+e)/2;
    update(Left,b,mid,i,j,x);
    update(Right,mid+1,e,i,j,x);
    tree[node].sum=tree[Left].sum+tree[Right].sum+(e-
b+1)*tree[node].prop;
    ///উপরে উঠার সময় পথের নোডগুলো আপডেট হবে
    ///বাম আর ডান পাশের SUM ছাড়াও যোগ হবে নিচে অতিরিক্ত যোগ
হওয়া মান
}

ll query(ll node,ll b,ll e,ll i,ll j,ll carry=0)
{
    if (i > e || j < b) return 0;

    if(b>=i and e<=j) return tree[node].sum+carry*(e-b+1);
//সাম এর সাথে যোগ হবে সেই রেঞ্জের সাথে অতিরিক্ত যত যোগ করতে
বলেছে সেটা
```

```
    ll Left=node*2;
    ll Right=(node*2)+1;
    ll mid=(b+e)/2;

    ll ret1 = query(Left, b,mid, i, j,
carry+tree[node].prop); //প্রপাগেট ভ্যালু বয়ে নিয়ে যাচ্ছে carry
ভ্যারিয়েবল
    ll ret2 = query(Right, mid+1, e, i,
j,carry+tree[node].prop);

    return ret1+ret2;
}
```

- Dijkstra:

```
vector<int>g[100],cost[100];
struct node
{
    int u,w;
    node(int a,int b){u=a; w=b;}
    bool operator < ( const node& p ) const {      return w
> p.w;   }
};

rest of code same as BFS
```

- Sliding windows range minimum query:

```
deque< pair<int,int> >DQ;//we also need the index of the
number hence deque is pair type.
for(i=0;i<n;i++)
{
   while(!DQ.empty()&&DQ.back().first>=ar[i])
    DQ.pop_back();
   DQ.push_back(make_pair(ar[i],i));
   while(DQ.front().second<=i-k)
    DQ.pop_front();
   pf("%d ",DQ.front());
}
```

String:

- KMP:

```
ll f[10000];
void build_failure_function(string pattern)
{
    f[0] = f[1] = 0;

    ll i,j;
    for(i=2;i<=m;i++)
    {
        j = f[i-1];
        while(true)
        {
            if(pattern[i-1]==pattern[j])
            {
                f[i] = j+1;
                break;
            }

            if(j==0)
            {
                f[i] = 0;
                break;
            }
            j = f[j];
        }
    }
}

void KMP(string text,string pattern)
{
    build_failure_function(pattern);

    ll n = text.size();
    ll m = pattern.size();
    for(ll i=0;i<n;i++)
    {

    }
}
```

- Longest Palindromic Subsequence:

```
string s;
int t=0;

int rec(int i,int j)
{
    if(i>j)
        return 0;

    t+=rec(i+1,j);
    t+=rec(i,j-1);
    t-=rec(i+1,j-1);
    if(s[i]==s[j])
        t+=rec(i+1,j-1)+1;
    return t;
}
```

- Big Integer Class:

```
class Bigint{

    private:
    // representations and structures
    string a; // to store the digits
    int sign; // sign = -1 for negative numbers, sign = 1
otherwise
    // constructors
    public:
    Bigint() {} // default constructor
    Bigint( string b )
    {
        (*this) = b;    // constructor for string
    }

    // some helpful methods
    int size()   // returns number of digits
    {
        return a.size();
    }
    Bigint inverseSign()   // changes the sign
    {
        sign *= -1;
        return (*this);
    }
```

```
    Bigint normalize( int newSign )    // removes leading 0,
fixes sign
    {
        for( int i = a.size() - 1; i > 0 && a[i] == '0'; i-
- )
            a.erase(a.begin() + i);
        sign = ( a.size() == 1 && a[0] == '0' ) ? 1 :
newSign;
        return (*this);
    }

    // assignment operator
    void operator = ( string b )    // assigns a string to
Bigint
    {
        a = b[0] == '-' ? b.substr(1) : b;
        reverse( a.begin(), a.end() );
        this->normalize( b[0] == '-' ? -1 : 1 );
    }

    // conditional operators
    bool operator < ( const Bigint &b ) const    // less
than operator
    {
        if( sign != b.sign ) return sign < b.sign;
        if( a.size() != b.a.size() )
            return sign == 1 ? a.size() < b.a.size() :
a.size() > b.a.size();
        for( int i = a.size() - 1; i >= 0; i-- ) if( a[i]
!= b.a[i] )
                return sign == 1 ? a[i] < b.a[i] : a[i] >
b.a[i];
        return false;
    }
    bool operator == ( const Bigint &b ) const    //
operator for equality
    {
        return a == b.a && sign == b.sign;
    }
    // mathematical operators
    Bigint operator + ( Bigint b )    // addition operator
overloading
    {
        if( sign != b.sign ) return (*this) -
b.inverseSign();
```

```
        Bigint c;
        for(int i = 0, carry = 0; i<a.size() || i<b.size()
|| carry; i++ )
        {
            carry+=(i<a.size() ? a[i]-48 : 0)+(i<b.a.size()
? b.a[i]-48 : 0);
            c.a += (carry % 10 + 48);
            carry /= 10;
        }
        return c.normalize(sign);
    }
    Bigint operator - ( Bigint b )   // subtraction
operator overloading
    {
        if( sign != b.sign ) return (*this) +
b.inverseSign();
        int s = sign;
        sign = b.sign = 1;
        if( (*this) < b ) return ((b -
(*this)).inverseSign()).normalize(-s);
        Bigint c;
        for( int i = 0, borrow = 0; i < a.size(); i++ )
        {
            borrow = a[i] - borrow - (i < b.size() ? b.a[i]
: 48);
            c.a += borrow >= 0 ? borrow + 48 : borrow + 58;
            borrow = borrow >= 0 ? 0 : 1;
        }
        return c.normalize(s);
    }
    Bigint operator * ( Bigint b )   // multiplication
operator overloading
    {
        Bigint c("0");
        for( int i = 0, k = a[i] - 48; i < a.size(); i++, k
= a[i] - 48 )
        {
            while(k--) c = c + b; // ith digit is k, so, we
add k times
            b.a.insert(b.a.begin(), '0'); // multiplied by
10
        }
        return c.normalize(sign * b.sign);
    }
    Bigint operator / ( Bigint b )   // division operator
```

```
overloading
    {
        if( b.size() == 1 && b.a[0] == '0' ) b.a[0] /= (
b.a[0] - 48 );
        Bigint c("0"), d;
        for( int j = 0; j < a.size(); j++ ) d.a += "0";
        int dSign = sign * b.sign;
        b.sign = 1;
        for( int i = a.size() - 1; i >= 0; i-- )
        {
            c.a.insert( c.a.begin(), '0');
            c = c + a.substr( i, 1 );
            while( !( c < b ) ) c = c - b, d.a[i]++;
        }
        return d.normalize(dSign);
    }
    Bigint operator % ( Bigint b )   // modulo operator
overloading
    {
        if( b.size() == 1 && b.a[0] == '0' ) b.a[0] /= (
b.a[0] - 48 );
        Bigint c("0");
        b.sign = 1;
        for( int i = a.size() - 1; i >= 0; i-- )
        {
            c.a.insert( c.a.begin(), '0');
            c = c + a.substr( i, 1 );
            while( !( c < b ) ) c = c - b;
        }
        return c.normalize(sign);
    }

    // output method
    void print()
    {
        if( sign == -1 ) putchar('-');
        for( int i = a.size() - 1; i >= 0; i-- )
putchar(a[i]);
    }
};
```

STL:

- Upper bound & lower bound:

```
/**
    Suppose you have to count the number of elements in the
    range a to b in a sorted vector v */

vector<int> v;

ll li=lower_bound(v.begin(),v.end(),a)-v.begin();
ll ui=upper_bound(v.begin(),v.end(),b)-v.begin();

cout<<lu-li<<"\n";
```

- Vector:

```
// erase the 6th element
myvector.erase (myvector.begin()+5);
// erase the first 3 elements:
myvector.erase (myvector.begin(),myvector.begin()+3);
```

- Set:

```
for (int i=1; i<10; i++) //inserting element into set
    myset.insert(i*10); // insert 10 20 30 40 50 60 70 80
90

for (it=myset.begin(); it!=myset.end(); ++it)//accessing
set element
        std::cout << ' ' << *it;
//erasing elements from set
myset.erase (40);
it = myset.find (60);
myset.erase (it);
```

Bitwise operation:

- Some important bitwise function:

```
int Set(int N,int pos){return N=N | (1<<pos);}
int reset(int N,int pos){return N= N & ~(1<<pos);}
bool check(int N,int pos){return (bool)(N & (1<<pos));}
```

ASCII Character table:

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Symbol
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

ASCII	Hex	Symbol
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

ASCII	Hex	Symbol
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

ASCII	Hex	Symbol
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	□