Code Template for ACM-ICPC

CZWin32768 @ BIT

September 20, 2018

Contents

1 Number-Representation

1.1 BigDecimal

```
// methods
public static double add (double v1, double v2);
public static double sub (double v1, double v2);
public static double mul (double v1, double v2);
public static double div (double v1, double v2);
public static double div (double v1, double v2, int
    scale);
public static double round (double v1, double v2);

//example
double v1 = 14, v2 = 9;
BigDecimal b1 = new BigDecimal(Double.toString(v1));
BigDecimal b2 = new BigDecimal(Double.toString(v2));
BigDecimal res = b1.divide(b2, 10,
    BigDecimal.ROUND_HALF_UP);

/*
ROUND PROPERTIES:

ROUND_CEILING: If the BigDecimal is positive, behave as
    for ROUND_UP; if negative, behave as for
    ROUND_DOWN.
ROUND_DOWN: Never increment the digit prior to a
    discarded fraction (i.e., truncate).
ROUND_FLOOR: If the BigDecimal is positive, behave as
    for ROUND_DOWN; if negative behave as for ROUND_UP.
ROUND_HALF_DOWN: Behave as for ROUND_UP if the
    discarded fraction is > .5; otherwise, behave as
    for ROUND_DOWN.
ROUND_HALF_EVEN: Behave as for ROUND_HALF_UP if the
    digit to the left of the discarded fraction is
    odd; behave as for ROUND_HALF_DOWN if it's even.
ROUND_HALF_UP: Behave as for ROUND_UP if the discarded
    fraction is >= .5; otherwise, behave as for
    ROUND_DOWN.
ROUND_UNNECESSARY: This "pseudo-rounding-mode" is
    actually an assertion that the requested operation
    has an exact result, hence no rounding is
    necessary.
ROUND_UP: Always increment the digit prior to a
    non-zero discarded fraction.
*/
```

1.2 pydecimal

```
import decimal as D
D.getcontext().prec = 10
D.getcontext().rounding = D.ROUND_HALF_DOWN
print(D.Decimal(14) / D.Decimal("9"))
```

2 STL

2.1 next-permutation

```
// Example for Array:
int a[N];
sort(a, a+N);
next_permutation(a, a+N);

// Example for vector:
vector<int> ivec;
sort(ivec.begin(), ivec.end());
next_permutation(ivec.begin(), ivec.end());
//Example for loop:
vector<int> myVec;
sort(myVec.begin(),myVec.end());
do{
    for (i = 0 ;i < size;i ++ ) cout << myVec[i] << "
        \t " ;
    cout << endl;
}while (next_permutation(myVec.begin(), myVec.end()));
```

2.2 priority-queue

```
struct node
{
    int x,y;
};

struct cmp{
    bool operator()(node a,node b)
    {
        if(a.x==b.x) return a.y > b.y;
        return a.x < b.x;
    }
};

priority_queue<int,vector<int>,greater<int> > q;
priority_queue<node,vector<node>,cmp > qq;
```

2.3 nth-element

```
struct node
{
    int val,pos;
}A[10];

int cmp(node a,node b)
{
    return a.pos < b.pos;
}

int main()
{
    int a[10] = {-1,3,9,1,4,5,8,7,6,2};
    int i;
    while(0){
        cin >> i;
        nth_element(a+1,a+i,a+9+1);
        cout << a[i];
    }
    for(int i=1;i<=9;i++) A[i].pos = 10-i,A[i].val=i;
    nth_element(A+1,A+4,A+9+1,cmp);
    printf("%d\n",A[4].pos);
}
```

3 IntervalQuery

3.1 Number-of-Differnet-Numbers

```
int N,M;
const int SIZE = 50005;
int c[SIZE], A[SIZE], Next[SIZE], res[200005],
    show[1000005];
bool fir[SIZE];
struct Q{
  int l,r;
  int pos;
} q[200005];
int lowbit(int k) {return k&(-k);}
void modify(int n,int v){
  while(n <= N){
    c[n] += v;
    n += lowbit(n);
  }
}
int sum(int n) {
  int ans = 0;
  while(n > 0) {
      ans += c[n];
      n -= lowbit(n);
  }
  return ans;
}
int cmp(Q a, Q b){
  return a.l < b.l;
}
int main()
{
  scanf("%d",&N);
  for(int i=1;i<=N;i++) scanf("%d",&A[i]);
  for(int i=N;i>=1;i--){
    if(!show[A[i]]){
        show[A[i]] = i;
        fir[i] = true;
    }
    else{
        Next[i] = show[A[i]];
        fir[Next[i]] = false;
        fir[i] = true;
        show[A[i]] = i;
    }
  }
  scanf("%d",&M);
  for(int i=1;i<=M;i++){
      scanf("%d%d",&q[i].l,&q[i].r);
      q[i].pos = i;
  }
  sort(q+1,q+1+M,cmp);
  for(int i=1;i<=N;i++)
    if(fir[i])
    {
        modify(i,1);
    }
  int qtemp = q[1].l;
  int ptr = 1;
  for(int i=1;i<=M;i++){
    for(;ptr<q[i].l;ptr++){
      if(fir[ptr]){
        modify(ptr,-1);
        fir[ptr] = false;
        if(Next[ptr])
        {
          fir[Next[ptr]] = true;
          modify(Next[ptr],1);
        }
      }
    }
    ptr = q[i].l;
    qtemp = q[i].l;
    res[q[i].pos] = sum(q[i].r) - sum(q[i].l-1);
  }
  for(int i=1;i<=M;i++) printf("%d\n",res[i]);
  return 0;
}
```

4 Tree

4.1 Persistent-Segment-Tree

```
// calc number of different prefix in the string list
    [s_l, ..., s_i, ..., s_r]
#include<bits/stdc++.h>
using namespace std;

namespace Trie {
    const int SIZE = 100005;
    int node[SIZE][26];
    int tot, bel[SIZE];
    void Insert(string& str) {
        int cur = 0;
        for(int i = 0; i < str.size(); i++) {
            int p = str[i] - 'a';
            if(node[cur][p] == 0) {
                tot++;
                node[cur][p] = tot;
                memset(node[tot], 0, sizeof(node[tot]));
            }
            cur = node[cur][p];
            bel[cur] = 0;
        }
    }
    void init() {
        tot = 0;
        memset(node[0], 0, sizeof(node[0]));
    }
}

namespace PST {
    const int MAXN = 100005;
    const int M = MAXN * 40;
    int tot;
    int n;
    int T[MAXN],lson[M],rson[M],c[M];
    void init(int _n) {
        tot = 0;
        n = _n;
    }
    int build(int l,int r) {
        int root = tot++;
        c[root] = 0;
        if(l != r) {
            int mid = (l+r)>>1;
            lson[root] = build(l,mid);
            rson[root] = build(mid+1,r);
        }
        return root;
    }
    int update(int root,int pos,int val) {
        int newroot = tot++, tmp = newroot;
        c[newroot] = c[root] + val;
        int l = 1, r = n;
```

```
        while(l < r) {
            int mid = (l+r)>>1;
            if(pos <= mid) {
                lson[newroot] = tot++; rson[newroot] =
                    rson[root];
                newroot = lson[newroot]; root =
                    lson[root];
                r = mid;
            }
            else {
                rson[newroot] = tot++; lson[newroot] =
                    lson[root];
                newroot = rson[newroot]; root =
                    rson[root];
                l = mid+1;
            }
            c[newroot] = c[root] + val;
        }
        return tmp;
    }
    int query(int root,int pos) {
        int ret = 0;
        int l = 1, r = n;
        while(pos < r) {
            int mid = (l+r)>>1;
            if(pos <= mid) {
                r = mid;
                root = lson[root];
            }
            else {
                ret += c[lson[root]];
                root = rson[root];
                l = mid+1;
            }
        }
        return ret + c[root];
    }
}
string s[PST::MAXN];

int main() {
    int N;
    while(~scanf("%d",&N)) {
        PST::init(N);
        Trie::init();
        for(int i = 1; i <= N; i++) {
            cin >> s[i];
            Trie::Insert(s[i]);
        }
        PST::T[N+1] = PST::build(1, N);
        for(int i = N; i >= 1; i--) {
            int cur = 0;
            PST::T[i] = PST::T[i+1];
            for(int j = 0; j < s[i].size(); j++) {
                int p = s[i][j] - 'a';
                cur = Trie::node[cur][p];
                if(Trie::bel[cur]) {
                    //Eliminate the influence of appeared
                        prefix
                    PST::T[i] = PST::update(PST::T[i],
                        Trie::bel[cur], -1);
                }
                Trie::bel[cur] = i; //record the last
                    position of prefix
            }
            PST::T[i] =
                PST::update(PST::T[i],i,s[i].size());
        }
        int Q;
        scanf("%d",&Q);
        int Z = 0;
        while(Q--) {
            int l, r;
            scanf("%d%d",&l,&r);
            l += Z; l %= N;
            r += Z; r %= N;
            if(l > r) swap(l, r);
            Z = PST::query(PST::T[l+1],r+1);
            printf("%d\n",Z);
        }
    }
}
```

4.2 LvTrie

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

using namespace std;

const int SIZE = 1005;
int a[SIZE];

const int maxn = 1000005;

int chd[maxn*32][2],nn = 0,num[maxn*32];

void init(){
    nn = 1;
    memset(chd[0],-1,sizeof(chd[0]));
    //memset(num,0,sizeof(num));
    num[0] = 0;
}

void insert(int x){
    int p = 0;
    //printf("--insert\n");
    for(int i = 31;i >= 0;i--){
        int t = (x>>i)&1;
        //printf("%d\n",p);
        if(chd[p][t]==-1){
            chd[p][t] = nn;
            memset(chd[nn],-1,sizeof(chd[nn]));
            num[nn] = 0;
            nn++;
        }
        p = chd[p][t];
    }
    num[p]++;
    //printf("num = %d\n",num[p]);
}

int remove(int x,int p = 0,int d = 31){
    if(d == -1){
        num[p]--;
        if(num[p] == 0){
            return 1;
        }
        return 0;
    }
    int ret;
```

```
    int t = (x>>d)&1;
    int s = chd[p][t];
    ret = remove(x,s,d-1);
    if(ret) chd[p][t] = -1;
    if(chd[p][0]==-1&&chd[p][1]==-1) return 1;
    return 0;
}

int find(int x){
    int p = 0;
    int ret = 0;
    //printf("find---\n");
    for(int i = 31;i >= 0;i--){
        //printf("%d\n",p);
        int t = (x>>i)&1;
        if(chd[p][t]==-1) t^=1;
        p = chd[p][t];
        ret <<= 1;
        ret |= t;
    }
    //printf("num = %d\n",num[p]);
    return ret;
}
// find the most distant number
int f(int n) {
    int p = 0;
    int ret = 0;
    for(int i = 31; i >= 0; i--) {
        int t = (n >> i) & 1 ^ 1;
        if(chd[p][t] == -1) t ^= 1;
        p = chd[p][t];
        ret <<= 1;
        ret |= t;
    }
    //printf("ret=%d\n",ret);
    return ret ^ n;
}
```

4.3 Barycenter-of-Tree

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

using namespace std;

const int SIZE =300005;

int n, q;
vector<int> g[SIZE];
int sz[SIZE];
int res[SIZE];
int fa[SIZE];

void dfs(int u) {
    sz[u] = 1;
    res[u] = u;
    int mx = 0;
    for(int v: g[u]) {
        dfs(v);
        sz[u] += sz[v];
        if(sz[v] > sz[mx]) mx = v;
    }
    if(sz[mx] * 2 > sz[u]) {
        int t = res[mx];
        while((sz[u]-sz[t]) *2 > sz[u]) t = fa[t];
        res[u] = t;
    }
}

int main() {
    scanf("%d%d",&n, &q);
    for(int i = 2; i <= n; i++) {
        int a;
        scanf("%d",&a);
        fa[i] = a;
        g[a].push_back(i);
    }
    dfs(1);
    while (q--)
    {
        int x;
        scanf("%d", &x);
        printf("%d\n", res[x]);
    }
    return 0;
}
```

5 IO

5.1 FastIO

```
namespace fastIO {
    #define BUF_SIZE 100000
    //fread -> read
    bool IOerror = 0;
    inline char nc() {
        static char buf[BUF_SIZE], *p1 = buf +
            BUF_SIZE, *pend = buf + BUF_SIZE;
        if(p1 == pend) {
            p1 = buf;
            pend = buf + fread(buf, 1, BUF_SIZE, stdin);
            if(pend == p1) {
                IOerror = 1;
                return -1;
            }
        }
        return *p1++;
    }
    inline bool blank(char ch) {
        return ch == ' ' || ch == '\n' || ch == '\r' ||
            ch == '\t';
    }
    inline void read(int &x) {
        char ch;
        while(blank(ch = nc()));
        if(IOerror)
            return;
        for(x = ch - '0'; (ch = nc()) >= '0' && ch <=
            '9'; x = x * 10 + ch - '0');
    }
    #undef BUF_SIZE
};
```

6 String

6.1 Suffix-Automaton

```
const int CHAR = 26;
```

```
const int MAXN = 250010;
struct SAM_Node
{
    SAM_Node *fa,*next[CHAR];
    int len;
    int id,pos;
    SAM_Node(){}
    SAM_Node(int _len)
    {
        fa = 0;
        len = _len;
        memset(next,0,sizeof(next));
    }
};
SAM_Node SAM_node[MAXN*2], *SAM_root, *SAM_last;
int SAM_size;
SAM_Node *newSAM_Node(int len)
{
    SAM_node[SAM_size] = SAM_Node(len);
    SAM_node[SAM_size].id = SAM_size;
    return &SAM_node[SAM_size++];
}
SAM_Node *newSAM_Node(SAM_Node *p)
{
    SAM_node[SAM_size] = *p;
    SAM_node[SAM_size].id = SAM_size;
    return &SAM_node[SAM_size++];
}
void SAM_init()
{
    SAM_size = 0;
    SAM_root = SAM_last = newSAM_Node(0);
    SAM_node[0].pos = 0;
}
void SAM_add(int x,int len)
{
    SAM_Node *p = SAM_last, *np = newSAM_Node(p->len+1);
    np->pos = len;
    SAM_last = np;
    for(;p && !p->next[x];p = p->fa)
    p->next[x] = np;
    if(!p)
    {
        np->fa = SAM_root;
        return;
    }
    SAM_Node *q = p->next[x];
    if(q->len == p->len + 1)
    {
        np->fa = q;
        return;
    }
    SAM_Node *nq = newSAM_Node(q);
    nq->len = p->len + 1;
    q->fa = nq;
    np->fa = nq;
    for(;p && p->next[x] == q;p = p->fa)
    p->next[x] = nq;
}
void SAM_build(char *s)
{
    SAM_init();
    int len = strlen(s);
    for(int i = 0;i < len;i++)
    SAM_add(s[i] - 'a',i+1);
}
```

```
//topological-sort:
char str[MAXN];
int topocnt[MAXN];
SAM_Node *topsam[MAXN*2];
int n = strlen(str);
SAM_build(str);
memset(topocnt,0,sizeof(topocnt));
for(int i = 0;i < SAM_size;i++)
topocnt[SAM_node[i].len]++;
for(int i = 1;i <= n;i++)
topocnt[i] += topocnt[i-1];
for(int i = 0;i < SAM_size;i++)
topsam[--topocnt[SAM_node[i].len]] = &SAM_node[i];
```

6.2 Extended-KMP

```
//next[i]: longest common prefix of x[i..m-1] and
    x[0...m-1]
//extend[i]: longest common prefix of y[i...n-1] and
    x[0...m-1]
void pre_EKMP(char x[],int m,int next[])
{
    next[0]=m;
    int j=0;
    while(j+1<m && x[j]==x[j+1])j++;
    next[1]=j;
    int k=1;
    for(int i=2;i<m;i++)
    {
        int p=next[k]+k-1;
        int L=next[i-k];
        if(i+L<p+1)next[i]=L;
        else
        {
            j=max(0,p-i+1);
            while(i+j<m && x[i+j]==x[j])j++;
            next[i]=j;
            k=i;
        }
    }
}
void EKMP(char x[],int m,char y[],int n,int next[],int
    extend[])
{
    pre_EKMP(x,m,next);
    int j=0;
    while(j<n && j<m && x[j]==y[j])j++;
    extend[0]=j;
    int k=0;
    for(int i=1;i<n;i++)
    {
        int p=extend[k]+k-1;
        int L=next[i-k];
        if(i+L<p+1)extend[i]=L;
        else
        {
            j=max(0,p-i+1);
            while(i+j<n && j<m && y[i+j]==x[j])j++;
            extend[i]=j;
            k=i;
        }
    }
}
```

6.3 ACAutomaton

```
const int MN,Z;

struct AC
{
    int next[MN][Z],fail[MN],end[MN];
    int root,L;
    int newnode()
    {
        for(int i = 0;i < Z;i++)
            next[L][i] = -1;
        end[L++] = 0;
        return L-1;
    }
    void init()
    {
        L = 0;
        root = newnode();
    }
    void insert(char buf[])
    {
        int len = strlen(buf);
        int now = root;
        for(int i = 0;i < len;i++)
        {
            if(next[now][buf[i]-'a'] == -1)
                next[now][buf[i]-'a'] = newnode();
            now = next[now][buf[i]-'a'];
        }
        end[now] = 1;
    }
    void build()
    {
        queue<int>Q;
        fail[root] = root;
        for(int i = 0;i < Z;i++)
            if(next[root][i] == -1)
                next[root][i] = root;
            else
            {
                fail[next[root][i]] = root;
                Q.push(next[root][i]);
            }
        while( !Q.empty() )
        {
            int now = Q.front();
            Q.pop();
            for(int i = 0;i < Z;i++)
                if(next[now][i] == -1)
                    next[now][i] = next[fail[now]][i];
                else
                {
                    fail[next[now][i]]=next[fail[now]][i];
                    Q.push(next[now][i]);
                }
        }
    }
    int query(char buf[])
    {
        int len = strlen(buf);
        int now = root;
        int res = 0;
        for(int i = 0;i < len;i++)
        {
            now = next[now][buf[i]-'a'];
            int temp = now;
            while( temp != root )
            {
                res += end[temp];
                temp = fail[temp];
            }
        }
        return res;
    }
    void debug()
    {
        for(int i = 0;i < L;i++)
        {
            printf("id = %3d,fail = %3d,end = %3d,chi =
                 [",i,fail[i],end[i]);
            for(int j = 0;j < Z;j++)
                printf("%2d",next[i][j]);
            printf("]\n");
        }
    }
};
```

6.4 Manacher

```
const int MAXN=110010;
char Ma[MAXN*2];
int Mp[MAXN*2];
void Manacher(char s[],int len)
{
    int l=0;
    Ma[l++]='$';
    Ma[l++]='#';
    for(int i=0;i<len;i++)
    {
        Ma[l++]=s[i];
        Ma[l++]='#';
    }
    Ma[l]=0;
    int mx=0,id=0;
    for(int i=0;i<l;i++)
    {
        Mp[i]=mx>i?min(Mp[2*id-i],mx-i):1;
        while(Ma[i+Mp[i]]==Ma[i-Mp[i]])Mp[i]++;
        if(i+Mp[i]>mx)
        {
            mx=i+Mp[i];
            id=i;
        }
    }
}
/*
* abaaba
* i:
0 1 2 3 4 5 6 7 8 9 10 11 12 13
* Ma[i]: $ # a # b # a # a $ b # a #
* Mp[i]: 1 1 2 1 4 1 2 7 2 1 4 1 2 1
*/
char s[MAXN];
int main()
{
    while(scanf("%s",s)==1)
    {
        int len=strlen(s);
        Manacher(s,len);
        int ans=0;
        for(int i=0;i<2*len+2;i++)
```

```
        ans=max(ans,Mp[i]-1);
        printf("%d\n",ans);
    }
    return 0;
}
```

6.5 KMP

```
const int SIZEP = 10005;
const int SIZET = 1000005;
char pat[SIZEP];

int Next[SIZEP];
char text[SIZET];
int lt,lp;

void getNext()
{
    for(int i=0,j=-1;i<=lp;i++,j++)
    {
        Next[i] = j;
        while(j!=-1 && pat[i]!=pat[j]) j = Next[j];
    }
}

int query()
{
    int cnt = 0;
    for(int i=0,j=0;i<=lt;i++,j++)
    {
        if(j==lp) cnt++;
        while(~j && text[i] != pat[j]) j = Next[j];
    }
    return cnt;
}
```

6.6 Suffix-Array

```
const int MAXN=100010;
int t1[MAXN],t2[MAXN],c[MAXN];

//sa[1...n]->[0,N) rank[0...n-1]->[1,N] height[1...n]
bool cmp(int *r,int a,int b,int l)
{
    return r[a] == r[b] && r[a+l] == r[b+l];
}
void da(int str[],int sa[],int Rank[],int height[],int
    n,int m)
{
    n++;
    int i, j, p, *x = t1, *y = t2;
    for(i = 0;i < m;i++)c[i] = 0;
    for(i = 0;i < n;i++)c[x[i] = str[i]]++;
    for(i = 1;i < m;i++)c[i] += c[i-1];
    for(i = n-1;i >= 0;i--)sa[--c[x[i]]] = i;
    for(j = 1;j <= n; j <<= 1)
    {
        p = 0;
        for(i = n-j; i < n; i++)y[p++] = i;
        for(i = 0; i < n; i++)if(sa[i] >= j)y[p++] =
            sa[i] - j;
        for(i = 0; i < m; i++)c[i] = 0;
        for(i = 0; i < n; i++)c[x[y[i]]]++;
        for(i = 1; i < m;i++)c[i] += c[i-1];
        for(i = n-1; i >= 0;i--)sa[--c[x[y[i]]]] = y[i];
        swap(x,y);
        p = 1; x[sa[0]] = 0;
        for(i = 1;i < n;i++)
            x[sa[i]] = cmp(y,sa[i-1],sa[i],j)?p-1:p++;
        if(p >= n)break;
        m = p;
    }
    int k = 0;
    n--;
    for(i = 0;i <= n;i++)Rank[sa[i]] = i;
    for(i = 0;i < n;i++)
    {
        if(k)k--;
        j = sa[Rank[i]-1];
        while(str[i+k] == str[j+k])k++;
        height[Rank[i]] = k;
    }
}
int Rank[MAXN],height[MAXN];

char str[MAXN];
int r[MAXN];
int sa[MAXN];
int A[MAXN];

int main()
{
    while(~scanf("%s", str)) {
        int n = strlen(str);
        for(int i = 0; i < n; i++) r[i] = str[i];
        r[n] = 0;
        da(r,sa,Rank,height,n,128);
        int res = n;
        int temp = n;
        for(int i = Rank[0]; i > 1; i--) {
            temp = min(temp, height[i]);
            res += temp;
            res %= 256;
        }
        temp = n;
        for(int i = Rank[0] + 1; i <= n; i++) {
            temp = min(temp, height[i]);
            res += temp;
            res %= 256;
        }
        res %= 256;
        printf("%d\n",res);
    }
    return 0;
}
```

7 Others

7.1 Date

```
//Constructors
//As of JDK version 1.1, replaced by Calendar.set(year
    + 1900, month, date) or GregorianCalendar(year +
    1900, month, date).
Date(int year, int month, int date)

//Allocates a Date object and initializes it to
    represent the specified number of milliseconds
```

```
    since the standard base time known as "the epoch",
    namely January 1, 1970, 00:00:00 GMT.
Date(long date)

//Methods
boolean after(Date when)
boolean before(Date when)
void    setTime(long time)
```

7.2 Calendar

```
//Constructors:
GregorianCalendar(int year, int month, int dayOfMonth)
GregorianCalendar(int year, int month, int dayOfMonth,
    int hourOfDay, int minute)
GregorianCalendar(int year, int month, int dayOfMonth,
    int hourOfDay, int minute, int second)

//Fields
//Field number for get and set indicating the day of
    the month.
static int     DAY_OF_MONTH
//Field number for get and set indicating the day of
    the week.
static int     DAY_OF_WEEK
//Field number for get and set indicating the day
    number within the current year.
static int     DAY_OF_YEAR
//Field number for get and set indicating the hour of
    the day.
static int     HOUR_OF_DAY
//Field number for get and set indicating the week
    number within the current month.
static int     WEEK_OF_MONTH
//Field number for get and set indicating the week
    number within the current year.
static int     WEEK_OF_YEAR
//Field number for get and set indicating the second
    within the minute.
static int     SECOND
//Field number for get and set indicating the minute
    within the hour.
static int     MINUTE
//Field number for get and set indicating the hour of
    the morning or afternoon.
static int     HOUR
//Field number for get and set indicating the day of
    the month.
static int     DATE
//Field number for get and set indicating the month.
static int     MONTH
//Field number for get and set indicating the year.
static int     YEAR
//The currently set time for this calendar, expressed
    in milliseconds after January 1, 1970, 0:00:00 GMT.
protected long time

//Methods:
//Converts calendar field values to the time value
    (millisecond offset from the Epoch).
protected void computeTime()
//Returns the number of weeks in the week year
    represented by this GregorianCalendar.
int     getWeeksInWeekYear()
//Returns the week year represented by this
    GregorianCalendar.
int     getWeekYear()
//Determines if the given year is a leap year.
boolean isLeapYear(int year)
//Sets the GregorianCalendar change date.
void setGregorianChange(Date date)
//Returns whether this Calendar represents a time after
    the time represented by the specified Object.
boolean after(Object when)
//Returns whether this Calendar represents a time
    before the time represented by the specified
    Object.
boolean before(Object when)
//Compares the time values (millisecond offsets from
    the Epoch) represented by two Calendar objects.
int     compareTo(Calendar anotherCalendar)
//Returns a Date object representing this Calendar's
    time value (millisecond offset from the Epoch").
Date getTime()


//Examples:
import java.util.GregorianCalendar;
import java.util.Calendar;

// Accessing Year, Month, Day etc.
Calendar calendar = new GregorianCalendar();

int year      = calendar.get(Calendar.YEAR);
int month     = calendar.get(Calendar.MONTH);
int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);
    // Jan = 0, not 1
int dayOfWeek = calendar.get(Calendar.DAY_OF_WEEK);
int weekOfYear = calendar.get(Calendar.WEEK_OF_YEAR);
int weekOfMonth= calendar.get(Calendar.WEEK_OF_MONTH);

int hour      = calendar.get(Calendar.HOUR);     // 12
    hour clock
int hourOfDay = calendar.get(Calendar.HOUR_OF_DAY); //
    24 hour clock
int minute    = calendar.get(Calendar.MINUTE);
int second    = calendar.get(Calendar.SECOND);
int millisecond= calendar.get(Calendar.MILLISECOND);

Calendar calendar = new GregorianCalendar();

//set date to last day of 2009
calendar.set(Calendar.YEAR, 2009);
calendar.set(Calendar.MONTH, 11); // 11 = december
calendar.set(Calendar.DAY_OF_MONTH, 31); // new years
    eve

//add one day
calendar.add(Calendar.DAY_OF_MONTH, 1);
//calendar.add(Calendar.DAY_OF_MONTH, -1);

//date is now jan. 1st 2010
int year      = calendar.get(Calendar.YEAR); // now 2010
int month     = calendar.get(Calendar.MONTH); // now 0
    (Jan = 0)
int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);
    // now 1

// set gregorian change at another date
calendar.setGregorianChange(new Date(92, 12, 10));
```