Algorithm Library

September 4, 2018

Contents

1 头文件

```
#include<iostream>
#include<cstdio>
#include<cmath>
#include<cstdlib>
#include<cstring>
#include<string>
#include<vector>
#include<stack>
#include<queue>
#include<set>
#include<map>
#include<algorithm>
using namespace std;
typedef long long  ll;
typedef unsigned long long ull;
```

2 暴力

2.1 枚举

```
do{
        for(int i=0;i<n;++i) cout<<num[i]<<endl;
    }while(next_permutation(num,num+n));   //prev_permutation
```

2.2 子集生成

```
void print_subset(int *A,int *B,int n,int cur){
    if(cur==n){
        for(int i=0;i<n;++i) if(B[i]) printf("%d ",A[i]);
        ↪  printf("\n");
        return;
    }
    B[cur]=0; print_subset(A,B,n,cur+1);
    B[cur]=1; print_subset(A,B,n,cur+1);
}
```

2.3 回溯

```
int vis[maxn];
void bt(int cur) {// (int x,int y)
    if(cur==n){
```

```
    }
    for(int i=0;i<4;++i){
        //改变

        bt(cur+dx[i])

        //恢复
    }
}
```

3 搜索

3.1 dfs

```
int cnt,vis[maxn][maxn];
void dfs(int x,int y,int id){
    // 判读边界
    dfs(nx,ny,id);
}

for(int i=0;i<n;++i){
    for(int j=0;j<n;++j){
        if(!vis[i][j]){
            vis[i][j]=1;
            dfs(i,j,++cnt);
            vis[i][j]=0;
        }
    }
}
```

3.2 bfs

```
queue<int> que;
int vis[maxn];
while(que.size()) que.pop();
memset(vis,0,sizeof(vis));
que.push(0); vis[0]=1;
void bfs(){
    while(que.size()){
        int x=que.front(); que.pop();
        for( ){
            vis[ ]=1; //每加入队列就标记
            que.push( );
        }
```

```
    }
}
```

4 数据结构

4.1 并查集

```
const int maxn=1010;
int fa[maxn];  //memset(fa,-1,sizeof(fa));
findroot(int x){
    if(fa[x]==-1) return x;
    return fa[x]=findroot(fa[x]);
}

void Union(int x,int y){
    x=findroot(x); y=findroot(y);
    if(x!=y) fa[x]=y;
}
```

4.2 heap

```
vector<int> vec;
make_heap(vec.begin(),vec.end(),greater<int>());
        for(int i=k;i<n;++i){
            int x;cin>>x;
            if(*vec.begin()<x){

                ↪ pop_heap(vec.begin(),vec.end(),greater<int>());{
                    cout<<"before pop "<<*(vec.end()-1)<<endl;
                    vec.pop_back();
                    cout<<"after pop "<<*(vec.end()-1)<<endl;
                }
                cout<<"before push "<<*(vec.end()-1)<<endl;
                vec.push_back(x);
                ↪ push_heap(vec.begin(),vec.end(),greater<int>());
                cout<<"after push "<<*(vec.end()-1)<<endl;
            }
        }
        sort_heap(vec.begin(),vec.end(),greater<int>());
        cout<<*(vec.end()-1)<<endl;
```

4.3 二叉树

4.3.1 各种遍历及合成

```
struct node{
    node *lc,*rc;
    char ch;
} nodes[maxn];

void preord(node *t){
    if(t==0) return;
    printf("%d",t->id);
    preord(t->lc); preord(t->rc);
}
int cur=0,idx=0;
node * newnode(){
    nodes[cur].lc=nodes[cur].rc=0;
    nodes[cur].ch=s[idx++];
    return &nodes[cur++];
}

node *build(){
    if(idx==len) return 0;
    char ch=s[idx];
    if(ch=='#') {++idx; return 0;}
    node *t=newnode();
    t->lc=build();
    t->rc=build();
    return t;
}
char s1[maxn],s2[maxn];
node *build(int l1,int r1,int l2,int r2){
    int idx;
    node *t=newnode(); t->ch=s1[l1];
    if(idx!=l2) t->lc=build(l1+1,idx-l2+l1+1,l2,idx);
    if(idx!=r2-1)t->rc=build(idx-l2+l1+1,r1,idx,r2);
    return t;
}
```

5 动态规划

5.1 背包

5.1.1 01 背包

```
int wei[maxn],val[maxn],dp[maxn],c;
memset(dp,0,sizeof(dp));  //若要恰好放满 则 dp[0]=0  其余为负无
↪ 穷
for(int i=0;i<n;++i){              //若 c 较小，n 较大，可调整
↪ i,j 的内外顺序
    for(int j=c;j>=wei[i];--j){   //完全背包的话，j 顺序反过来即
    ↪ 可
        dp[j]=max(dp[j],dp[j-wei[i]]+val[i]);
    }
}
```

5.1.2 多重背包

5.2 最长公共子序列

```
char s1[maxn],s2[maxn];
int dp[maxn][maxn];
for(int i=1;i<=n1;++i){  //从 1 开始存
        for(int j=1;j<=n2;++j){
                if(s1[i]==s2[j]) dp[i][j]=dp[i-1][j-1]+1;
        else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
    }
}


int i=n1,j=n2,path[maxn],k=-1;
while(dp[i][j]){
    if(dp[i-1][j]==dp[i][j]) --i;
    else if(dp[i][j-1]==dp[i][j])  --j;
    else {
            path[++k]=i;
            --i; --j;
    }
}
for(;k;--k) cout<<s1[path[k]]<<" ";
cout<<s1[path[0]]<<endl;
```

5.3 最长上升子序列

```
int n,num[1010],dp[1010];
const int inf=10000000;
int lis(){   // nlogn 但是不能打印路径
    fill(dp,dp+n,inf);
    for(int i=0;i<n;++i){
        *lower_bound(dp,dp+n,num[i])=num[i];
    }
    return lower_bound(dp,dp+n,inf)-dp;
}
```

```
int n,num[maxn],dp[maxn],pre[maxn];  //n^2 能打印路径
int ans,idx;
void lis(){
    memset(pre,-1,sizeof(pre));
   for(int i=0;i<n;++i){
       dp[i]=0;
       for(int j=0;j<i;++j){
           if(num[i]>num[j]&&dp[j]+1>dp[i]){
               dp[i]=dp[j]+1;
               pre[i]=j;
               if(dp[i]>ans){
                   ans=dp[i];
                   idx=i;
               }
           }
       }
   }
}
void print(int i){
    if(pre[i]!=-1)  print(pre[i]);
    printf("%d",num[i]);
}
int main(){
    while(scanf("%d",&n)==1){
        for(int i=0;i<n;++i) scanf("%d",&num[i]);
        lis();
        printf("%d\n",ans);
        print(idx);
        printf("\n");
```

```
    }
    return 0;
}
```

5.4 最长公共上升

```
int
↪  n,m,num1[maxn],num2[maxn],pre[maxn],dp[maxn],ans,Max,last,idx;
void lcis(){
    ans=0;
    memset(dp,0,sizeof(dp));
    memset(pre,-1,sizeof(pre));
    for(int i=0;i<n;++i){
        Max=0;
        for(int j=0;j<m;++j){  //dp[j] 表示串 1 的前 i 个和串的
        ↪  前 j 个，且以 num2[j] 结尾
            if(num1[i]>num2[j]) {
                Max=max(Max,dp[j]); //最大的 dp[i-1][k]
                last=j;
            }
            if(num1[i]==num2[j]) {
                dp[j]=Max+1;
                pre[j]=last;
            }
            if(dp[j]>ans){
                ans=dp[j];
                idx=j;
            }
        }
    }
}
void print(int idx){
    if(pre[idx]!=-1) print(pre[idx]);
    printf("%d",num2[idx]);
}
```

5.5 maxsubsum

```
int max_sub_sum(int *line,int *dp,int n){
    fill(dp,dp+n,0);
    dp[0]=line[0];
    for(int i=1;i<n;++i) dp[i]=max(dp[i],dp[i-1]+line[i]);
    int res=-inf;
    for(int i=0;i<n;++i) res=max(res,dp[i]);
```

```
    return res;
}
```

5.6 maxsubmatsum

```
for(int i=0;i<n;++i) {
    for(int j=0;j<n;++j) scanf("%d",&mat[i][j]);
}
fill(tot[0],tot[0]+n,0);
for(int i=1;i<=n;++i){
    for(int j=0;j<n;++j) tot[i][j]=tot[i-1][j]+mat[i-1][j];
}
int res=-inf;
for(int i=0;i<n;++i){
    for(int j=i+1;j<=n;++j){
        for(int k=0;k<n;++k) line[k]=tot[j][k]-tot[i][k];
        res=max(res,max_sub_sum(line,dp,n));
    }
}
```

5.7 tsp

```
int solve(){   //上回到起始点，如果需要回到起始点，
    int s;
    for( s=0;s< (1<<n)-1;++s) fill(dp[s],dp[s]+n,inf);
    fill(dp[s],dp[s]+n,0);
    for(s=(1<<n)-2;s>0;--s){
        for(int u=0;u<n;++u ){
            if((s&(1<<u))==0) continue;
            for(int v=0;v<n;++v){
                if(s&(1<<v)) continue;

                ↪  dp[s][u]=min(dp[s][u],dp[s|(1<<v)][v]+mat[u][v]);
            }
        }
    }
    s=1;
    int res=inf;
    for(int i=0;i<n;++i){
        res=min(res,dp[s][i]);
        s<<=1;
    }
    return res;
}
```

5.8 拆分方案数

```
void cnt(){
        fill(dp,dp+m+1,0); dp[0]=1; //恰好凑齐
        for(int i=0;i<n;++i){
            for(int j=m;j>=num[i];--j){
                dp[j]=(dp[j-num[i]]+dp[j]);
            }
        }
}
```

6 图论

6.1 tree

```
const int maxn=110;
int lc[maxn],rc[maxn],p[maxn];  //若节点编号严格是1-n
int n,m;
```

```
const int maxn=1010;
struct node{
    int val;
    node *lc,*rc;
} nodes[maxn];
int cur=0;
node *newnode(int v=0){
    nodes[cur].val=v;
    nodes[cur].lc=nodes[cur].rc=0;
    return &nodes[cur++];
}
```

6.2 拓扑排序

```
const int maxn=510;
vector<int> edges[maxn],ans;
int ind[maxn],n,m,used[maxn];
int dfs(int k){  // void dfs(int k) 只输出一组值 后面 return 处
↪  也要修改
    if(k==n){
        for(int i=0;i<n-1;++i) printf("%d ",ans[i]);
        printf("%d\n",ans[n-1]);
        return 1;
```

```
    }
    for(int i=1;i<=n;++i){
        if(!ind[i]&&!used[i]){
            used[i]=1; ans.push_back(i);
            for(int j=0;j<edges[i].size();++j)
            ↪  ind[edges[i][j]]--;
            if(dfs(k+1)) return 1;
            for(int j=0;j<edges[i].size();++j)
            ↪  ind[edges[i][j]]++;
            used[i]=0; ans.pop_back();
        }
    }
    return 0;
}
```

6.3 单源最短路径

6.3.1 Bellman-Ford

```
const int maxm=5210,maxn=510;
struct edge{
    int u,v,c;
}edges[maxm];

int e,n,dis[maxn];
bool bellman(){
    const int inf=100000000;  //不能用-1，在判断负环时有问题
    fill(dis,dis+n,inf);
    dis[1]=0;
    for(int i=0;i<n;++i){
        for(int j=0;j<e;++j){
            edge eg=edges[j];
            if(dis[eg.v]>dis[eg.u]+eg.c){
                dis[eg.v]=dis[eg.u]+eg.c;
                if(i==n-1) return 1;
            }
        }
    }
    return 0;
}

void bellman(){                  //无负边
    const int inf=100000000;
    fill(dis,dis+n,inf);
```

```
    dis[1]=0;
    while(1){
        int update=0;
        for(int i=0;i<e;++i){
            edge eg=edges[i];
            if(dis[eg.v]>dis[eg.u]+eg.c){
                dis[eg.v]=dis[eg.u]+eg.c;
                update = 1;
            }
        }
        if(!update) break;
    }
}
```

6.3.2 Dijkstra

```
struct edge{
    int to,w;
    bool operator<(const edge&b)const {return w>b.w;}
    edge(){to=w=0;}
    edge(int to,int w){this->to=to; this->w=w;}
};
const int maxn=1010,inf=1000000;
vector<edge> edges[maxn];
int n,dis[maxn];
priority_queue<edge> que;
void dijkstra(int s){
    while(que.size()) que.pop();
    fill(dis,dis+n+1,inf);
    que.push(edge(s,0)); dis[s]=0;
    while(que.size()){
        edge eg=que.top(); que.pop();
        int u=eg.to;
        if(eg.w!=dis[u]) continue;
       for(int i=0;i<edges[u].size();++i){
            int v=edges[u][i].to;
            if(dis[v]>dis[u]+edges[u][i].w){
                dis[v]=dis[u]+edges[u][i].w;
                que.push(edge(v,dis[v]));
            }
       }
    }
}
```

6.4 多源最短路径

6.4.1 Floyd

```
const int inf=1000000,maxn=210;   //支持负边
int n,dis[maxn][maxn];
void floyd(){          //dis[i][i]=0;  inf
    for(int k=1;k<=n;++k){
        for(int i=1;i<=n;++i){
            for(int j=1;j<=n;++j){
                dis[i][j]=min(dis[i][j],dis[i][k]+dis[k][j]);
            }
        }
    }     //通过检查 dis[i][i] 有没有为负的 可以判断有无负圈
}


//支持打印路径
const int maxn=40;
int p,q,r;
const int inf=1000000;
map<string,int> Id;
map<int,string> Name;
struct dis{
    int len,pre;
} Dis[maxn][maxn];
void print(int u,int v){
    if(u==v){
        cout<<Name[v]; return;
    }
    print(u,Dis[u][v].pre);

    ↪  cout<<"->("<<Dis[Dis[u][v].pre][v].len<<")"<<"->"<<Name[v];
}
for(int k=0;k<p;++k){
        for(int i=0;i<p;++i){
            for(int j=0;j<p;++j){
                int tmp=Dis[i][k].len+Dis[k][j].len;
                if(tmp<Dis[i][j].len){
                    Dis[i][j].len=tmp;
                    Dis[i][j].pre=Dis[k][j].pre;
                }
            }
        }
```

```
    }
```

6.5 最小生成树

6.5.1 Prim

```
const int maxn=110,inf=10000000;
int n,dis[maxn],used[maxn],mat[maxn][maxn];
int prim(){
    memset(used,0,sizeof(used));
    fill(dis,dis+n+1,inf);
    dis[1]=0;
    int res=0;
    while(1){
        int u=-1;
        for(int v=1;v<=n;++v){
            if(!used[v]&&(u==-1||dis[v]<dis[u])) u=v;
        }
        if(u==-1) break;
        used[u]=1;
        res+=dis[u];
        for(int v=1;v<=n;++v){
            dis[v]=min(dis[v],mat[u][v]); //注意不是
            ↪ dis[u]+mat[u][v]
        }
    }
    return res;
}
```

```
const int maxn=30,maxm=100,inf=100000;
int vis[maxn],mat[maxn][maxn],dis[maxn];
int n;
int prim(){
    memset(vis,0,sizeof(vis));

    fill(dis,dis+n,inf);
    int res=0;
    dis[0]=0;
    while(1){
        int u=-1;
        for(int v=0;v<n;++v){
            if(!vis[v]&&(u==-1||dis[v]<dis[u])) u=v;
```

```
        }
        if(u==-1)  break;
        vis[u]=1;
        res+=dis[u];
        for(int i=0;i<n;++i) dis[i]=min(dis[i],mat[u][i]);
    }
    return res;
}
```

```
struct edge{
    int v,l;
    bool operator<(const edge &b) const {return l>b.l;}
    edge(int v=0,int l=inf){
        this->v=v; this->l=l;
    }
};
priority_queue<edge> que;
int prim(){
    fill(dis,dis+n,inf);
    int res=0;
    dis[0]=0;
    memset(vis,0,sizeof(vis));
    while(que.size()) que.pop();
    que.push(edge(0,0));
    while(que.size()){
        int u=que.top().v,l=que.top().l;
        que.pop();
        if(dis[u]!=l||vis[u]) continue;
        res+=l;
        vis[u]=1;
        for(int v=0;v<n;++v) {
            if(dis[v]>mat[u][v]&&!vis[v]){
                dis[v]=mat[u][v];
                que.push(edge(v,dis[v]));
            }
        }
    }
    return res;
}
```

6.5.2 Kruskal

```
const int maxn=110,maxm=5010;
int n,m,fa[maxn];
int findroot(int u){
    if(fa[u]==-1) return u;
    return fa[u]=findroot(fa[u]);
}
void Union(int u,int v){
    u=findroot(u); v=findroot(v);  //上能直接 fa[u]=v;
    fa[u]=v;
}
struct edge{
    int u,v,w;
    bool operator<(const edge& b) const {return w<b.w;}
} edges[maxm];
int kruskal(){
    memset(fa,-1,sizeof(fa));
    sort(edges,edges+m);
    int res=0;
    for(int i=0;i<m;++i){
        edge eg=edges[i];
        int u=eg.u,v=eg.v;
        if(findroot(u)==findroot(v)) continue;
        res+=eg.w;
        Union(u,v);
    }
    return res;
}
```

7 数论

7.1 进制转换

```
int idx(char ch){
    if(ch>='0'&&ch<='9') return ch-'0';
    if(ch>='A'&&ch<='Z') return ch-'A'+10;
    if(ch>='a'&&ch<='z')  return ch-'a'+10;
}
char ch(int x){
    if(x<10) return '0'+x;
    else return 'A'+x-10;
}
int toten(int b,char *s){
```

```
    int res=0;
    int n=strlen(s);
    for(int i=0;i<n;++i){
        int t=idx(s[i]);
        res=res*b+t;
    }
    return res;
}
void tob(int x,int b,char *s){
    int i=0;
    char s1[10000];
    while(x){
        s1[i++]=ch(x%b);
        x/=b;
    }
    for(int j=0;j<i;++j) s[j]=s1[i-1-j];
   if(i==0) s[i++]='0';
    s[i]=0;
}
```

7.2 gcd

```
int gcd(int a,int b){
    if(b==0) return a;
    return gcd(b,a%b);
}
int lcm(int a, int b)
{
        return a * b / gcd(a, b);
}
int exgcd(int a,int n,int &x,int &y){
    if(n==0) {x=1; y=0; return a;}
    int res= exgcd(n,a%n,y,x);
    y-=x*(a/n);
    return res;
}
```

7.3 模线性方程

```
int modequ(int a,int b,int n){    //ax=b(mod n)
    int d=gcd(a,n);
    if(b%d!=0) return -1;
    a/=d; b/=d; n/=d;  n=abs(n);  //a'x=b(mod n)
    int x,y;
```

```
    exgcd(a,n,x,y);
    int p=x*b;                      //x=(a')^(-1)b'
    return ((p%n)+n)%n;
}
```

7.4 中国剩余定理

```
ll exgcd(ll a,ll b,ll &x,ll &y){
    if(b==0) {x=1; y=0; return a;}
    ll res=exgcd(b,a%b,y,x);
    y-=x*(a/b);
    return res;
}
ll M;
ll china(int n,int *a,int *m){   // x=a_i(mod m_i) i=1,2,,,n
    ll x,y,res=0;
    M=1;
    for(int i=0;i<n;++i) M*=m[i];
    for(int i=0;i<n;++i){
        ll w=M/m[i];
        exgcd(w,(ll)m[i],x,y);
        res=(res+w*x*a[i])%M;
    }
    return (res+M)%M;
}
```

7.5 快速幂

```
int pow_mod(ll a,ll p,ll n){
    if(a==0) return 0;
    if(p==0) return 1;
    ll tmp=pow_mod(a,p/2,n);
    tmp=tmp*tmp%n;
    if(p%2) tmp=tmp*a%n;
    return (int) tmp;
}
```

8 字符串

8.1 字典树

```
const int maxnode=100010,sigma_size=10;
struct trie{
    int ch[maxnode][sigma_size],val[maxnode];
```

```
    int sz;
    void init(){
        sz=1;
        memset(ch[0],0,sizeof(ch[0]));
        memset(val,0,sizeof(val));
    }
    trie(){init();}
    bool insert(char *s){
        int ok=1;
        int u=0,n=strlen(s);
        for(int i=0;i<n;++i){
           int c=s[i]-'0';
           if(!ch[u][c]){
               memset(ch[sz],0,sizeof(ch[sz]));
               ch[u][c]=sz++;
           }
           else{
               if(val[ch[u][c]]||i==n-1) ok=0;
           }
           u=ch[u][c];
        }
        val[u]=1;
        return ok;
    }
} ;
```

8.2 KMP

```
int kmp(char *t,char *p){
    int n=strlen(t),m=strlen(p);
    int *next=new int[m];
    next[0]=-1;
    for(int i=1,k=-1;i<m;++i){
        while(k>=0&&p[k+1]!=p[i]) k=next[k];
        if(p[k+1]==p[i]) ++k;
        next[i]=k;          //k 保存的是 p[0,1...i] 的最长的同时
        ↪ 是前缀和后缀的，那个前缀的最后一个字符的下标
    }
    for(int i=0,k=-1;i<n;++i){
        while(k>=0&&p[k+1]!=t[i]) k=next[k];
        if(p[k+1]==t[i]) ++k; //k 保存的是 p 中已经匹配的最后一
        ↪ 个字符的下标
        if(k==m-1) return i-m+1;
    }
```

```
    return -1;  //不匹配
}
```

8.3 hash

```
const ull B=1000000007; //实在不行 可以用两个 hash 另外用 1e9+9

bool contatin(string a,string b){
    int la=a.length(),lb=b.length();
    if(la>lb) return 0;
    ull ha=0,hb=0,t=1;
    for(int i=0;i<la;++i){
        t*=B;
        ha=ha*B+a[i];
        hb=hb*B+b[i];
    }
    for(int i=0;i+la<=lb;++i){
        if(hb==hb) return 1;
        if(i+la<lb) hb=hb*B+b[i+la]-b[i]*t;
    }
    return 0;
}
```

8.4 表达式计算

8.4.1 前缀表达式

```
#include<cstdlib>
#include<cstdio>
using namespace std;
char s[100];
double exp(){
    scanf("%s",s);
    char c=s[0];
    if(c=='+') return exp()+exp();
    else if(c=='-') return exp()-exp();
    else if(c=='*') return exp()*exp();
    else if(c=='/') return exp()/exp();
    else return atof(s);
}
int main(){
    printf("%f\n",exp());
}
```

```
//#include<cstdlib>
//#include<cstdio>
//#include<cstring>
//#include<string>
//#include<iostream>
//#include<queue>
//#include<stack>
//#include<sstream>
//using namespace std;
//typedef long long ll;
//stack<string> stk;  //这个方法有问题
//string s,s1;
//bool isnum(string str){
//    return (str.size()>1)||(str[0]>='0'&&str[0]<='9');
//}
//bool isop(string str){
//    char op=str[0];
//    return (op=='+'||op=='-'||op=='*'||op=='/');
//}
//double getres(double a,double b,string op){
//    if(op[0]=='+' ) return a+b;
//    if(op[0]=='-' ) return a-b;
//    if(op[0]=='*' ) return a*b;
//   else return a/b;
//}
//void calu(){
//    while(stk.size()>=3){
//        string s3,s2,s1;
//        s3=stk.top(); stk.pop();
//        s2=stk.top(); stk.pop();
//        s1=stk.top(); stk.pop();
//        if(isnum(s3)&&isnum(s2)&&isop(s1)){
//            double a=stod(s2),b=stod(s3);
//            stk.push(to_string(getres(a,b,s1)));
//        }else{
//            stk.push(s1); stk.push(s2); stk.push(s3);
//            break;
//        }
//    }
//}
//int main(){
//    while(getline(cin,s)){
//        stringstream ss(s);
```

```
//        while(stk.size()) stk.pop();
//        while(ss>>s1){
//            stk.push(s1);
//            calu();
//        }
//        printf("%f\n",stod(stk.top()));
//    }
//    return 0;
//}
```

8.4.2 中缀表达式

```
char s[610];
stack<char> ops;
stack<int> num;
int calu(int a,int b,char op){
    if(op=='+') return a+b;
    if(op=='-') return a-b;
    if(op=='*') return a*b;
    if(op=='/') return a/b;
    return -10000000;
}
bool higher(char op1,char op2){
    if(op2=='(') return 1;
    if((op1=='*'||op1=='/')&&(op2=='+'||op2=='-')) return 1;
    return 0;
}
int solve(){
    while(ops.size())   ops.pop();
    while(num.size())  num.pop();
    int n=strlen(s);
    for(int i=0;i<n;++i){
        if(s[i]=='(') ops.push(s[i]);
        else if(s[i]>='0'&&s[i]<='9'){
            int x=s[i]-'0';
            while(i+1<n&&s[i+1]>='0'&&s[i+1]<='9'){
                ++i;
                x=x*10+s[i]-'0';
            }
            num.push(x);
        }else if(s[i]==')'){
           while(ops.size()&&ops.top()!='('){
               int b,a;
               b=num.top(); num.pop();
```

```
                a=num.top(); num.pop();
                num.push(calu(a,b,ops.top()));
                ops.pop();
            }
            if(ops.size()&&ops.top()=='(') ops.pop();
        }else{
            char op1=s[i];
            while(ops.size()&&!higher(op1,ops.top())){
                int b,a;
                b=num.top(); num.pop();
                a=num.top(); num.pop();
                num.push(calu(a,b,ops.top()));
                ops.pop();
            }
            ops.push(op1);
        }
    }
    return num.top();
}
```

8.5 find longest shortest word

```
char s[210];
const int inf=10000;
bool isala(char c){ //is alaphabet
     return (c>='a'&&c<='z')||(c>='A'&&c<='Z');
}
void find_longest_shortest_word(){
    int n=strlen(s);
    int l=0,r=0,lmin,rmin,lmax,rmax;
    lmin=-1; rmin=inf;
    lmax=-1; rmax=-inf;
    while(l<n){
        while(l<n&&!isala(s[l])) ++l;
        if(l==n) break;
        r=l+1;
        while(r<n&&isala(s[r]))  ++r;
        s[r]=0;
        if(r-l<rmin-lmin) {lmin=l; rmin=r; }
        if(r-l>=rmax-lmax){lmax=l; rmax=r;}
        l=r+1;
    }
    if(lmax<0) printf("\n");
    else printf("%s\n",s+lmax);
```

```
    if(lmin<0) printf("\n");
    else printf("%s\n",s+lmin);
}

void find_longest_shortest_word(string s){
    stringstream ss;
    for(int i=0;i<s.size();++i)
        if(!isala(s[i])) s[i]=' ';
        ss.clear();
    ss.str(s);
    string Max="",Min(inf,'a');
    string t;
    while(ss>>t){
        if(t.size()>=Max.size()) Max=t;
        if(t.size()<Min.size())    Min=t;
    }
    cout<<Max<<endl;
    cout<<Min<<endl;
}
```

9 bigint

```
const int maxn=210;
struct bigint{
        int dig[maxn];
        bigint(string s=""){
                memset(dig,0,sizeof(dig));
                if(s.size()==0) {
                        dig[0]=1;
                        dig[1]=0;
                        return ;
                }
                int k,i;
                for(i=s.length()-1,k=0;i>=0;--i){
                        dig[++k]=s[i]-'0';
                }
                dig[0]=k;
        }
        void print(){
                for(int i=dig[0];i>0;--i){
                        if(dig[i]==0&&dig[0]>1) --dig[0];
                        else break;
                }
```

```
                for(int i=dig[0];i;--i) printf("%d",dig[i]);
        }
        bigint operator+(const bigint &b)const {
                bigint c;
                c.dig[0]=max(dig[0],b.dig[0]);
                for(int i=1;i<=c.dig[0];++i){
                        if(i<=dig[0]) c.dig[i]+=dig[i];
                        if(i<=b.dig[0]) c.dig[i]+=b.dig[i];
                }
                int x=0,i=1;
                while(x||i<=c.dig[0]){
                        x+=c.dig[i];
                        c.dig[i]=x%10;
                        x/=10;
                        ++i;
                }
                c.dig[0]=i-1;
                return c;
        }
        bigint operator*(const bigint& b)const{
                bigint c;
                c.dig[0]=dig[0]+b.dig[0]-1;
                for(int i=1;i<=dig[0];++i){
                        for(int j=1;j<=b.dig[0];++j){
                                c.dig[i+j-1]+=dig[i]*b.dig[j];
                        }
                }
                int x=0,i=1;
                while(i<=c.dig[0]||x){
                        x+=c.dig[i];
                        c.dig[i]=x%10;
                        x/=10;
                        ++i;
                }
                c.dig[0]=i-1;
                return c;
        }
};
```