康托和逆康托展开

```cpp
//标准版康托
const int MX=20;
long long factorial[MX+5]={1};
void init(){ for(int i=1;i<=MX;i++) factorial[i]=factorial[i-1]*i; }
long long cantor(const vector<int>& permutation){
    long long canter_id=0;
    int n=permutation.size();
    for(int i=0;i<n;i++){
        int rank=0;
        for(int j=i+1;j<n;j++)
            if(permutation[i]>permutation[j]) rank++;
        canter_id+=1ll*rank*factorial[n-i-1];
    }
    return canter_id;
}

//树状数组优化康托

const int MX=20;
long long factorial[MX+5]={1};
long long lowbit(long long x){ return x&-x; }
void add(long long pos){ for(;pos<=n;pos+=lowbit(pos)) ta[pos]++; }
long long query(long long pos){
    long long res=0;
    for(;pos;pos-=lowbit(pos)) res+=ta[pos];
    return res;
}

void init(){ for(int i=1;i<=MX;i++) factorial[i]=factorial[i-1]*i; }
long long cantor(const vector<int>& permutation){
    long long cantor_id=0;
    memset(ta,0,sizeof(ta));
    for(int i=0;i<n;i++)}{
        long long rank=permutation[i]-query(a[i])-1;
        add(a[i]);
        cantor_id+=rank*factorial[n-i-1];
    }
    return cantor_id;
}
//标准版逆康托

const int MX=20
long long factorial[MX+5]={1};
void init(){ for(int i=1;i<=MX;i++) factorial[i]=factorial[i-1]*i; }
vector<int> decanter(long long canter_id,int n){ //canter_id=canter_id-1
    vector<int> res;
    vector<pair<int,int>> choose(n);
    for(int i=1;i<=n;i++) choose[i]=make_pair(i,0);
    for(int i=1;i<=n;i++){
        long long remainder=canter_id%factorial[n-i];
        long long quotient=canter_id/factorial[n-i];
        canter_id=remainder;
```

```
        long long cnt=0;
        for(auto &v:choose){
            if(cnt>=quotient&&!v.second){
                res.push_back(v.first);
                v.second=1;
                break;
            }
            if(!v.second) cnt++;
        }
    }
    return res;
}

//树状数组优化版逆康托

const int N=5+1e6;
const int MX=20;
long long cnt[N];
long long factorial[MX+5]={1};
long long lowbit(long long x){ return x&-x; }
long long find_vk(long long k){
    long long ans=0,res=0;
    for(long long i=4 /*i=log(mx)*/ ;i>=0;i--){
        ans+=(1<<i);
        if(ans>=mx||res+cnt[ans]>=k) ans-=(1<<i);
        else res+=cnt[ans];
    }
    return ans+1;
}
void upd(long long pos,long long x){ for(;pos<mx;pos+=lowbit(pos))
cnt[pos]+=x; }

void init(){ for(int i=1;i<=MX;i++) factorial[i]=factorial[i-1]*i; }
vector<int> decantor(long long canter_id,int n){ //canter_id=canter_id-1
    for(int i=1;i<=mx;i++) cnt[i]=0;
    for(int i=1;i<=n;i++) upd(i,1);
    vector<int> res;
    for(int i=n;i>=1;i--){
        long long remainder=canter_id%factorial[i-1];
        long long quotient=canter_id/factorial[i-1];
        canter_id=remainder;
        long long now=find_vk(quotient+1);
        res.push_back(now);
        upd(now,-1);
    }
    return res;
}
```