ICPC CodeBook

1 Basic

1.1 .vimrc

```
syntax enable
set nu
set cursorline
set ts=2 sts=2 sw=2 et ai
set mouse=a
set wrap
set showcmd
set backspace=indent,eol,start

inoremap ( ()<ESC>i
inoremap [ []<ESC>i
inoremap {<CR> {<CR>}<ESC>ko
```

2 Dynamic Programming

2.1 0/1 Knapsack_problems

```
#include<bits/stdc++.h>
using namespace std;
int f[1000]={0};
int n=0, m=0;
int main(){
    cin >> n >> m;
    for (int i = 1;i <= n;i++){
        int price = 0, value = 0;
        cin >> price >> value;
        for (int j = m;j >= price;j--){
            if (f[j-price]+value>f[j]){
                f[j]=f[j-price]+value;
            }
        }
    }
    cout << f[m] << endl;
    return 0;
}
```

2.2 Complete_Knapsack_problems

```
#include<bits/stdc++.h>
using namespace std;
int f[1000]={0};
int n=0, m=0;
int main(){
    cin >> n >> m;
    for (int i=1;i<=n;i++){
        int price=0, value=0;
        cin >> price >> value;
        for (int j=price; j<=m; j++){
            if (f[j-price]+value>f[j]){
                f[j]=f[j-price]+value;
            }
        }
    }
    cout << f[m] << endl;
    return 0;
}
```

2.3 Longest Common Subsequence(LCS)

```
#include<bits/stdc++.h>
using namespace std;

int dp[1001][1001];
int lcs(const string &s, const string &t){
    int m = s.size(), n = t.size();
    if (m == 0 || n == 0){
        return 0;
    }
    for(int i = 0; i <= m; ++i){
        dp[i][0] = 0;
    }
    for(int j = 1; j <= n; ++j){
        dp[0][j] = 0;
    }
    for(int i = 0; i < m; ++i){
        for (int j = 0; j < n; ++j){
            if(s[i] == t[j]){
                dp[i+1][j+1] = dp[i][j]+1;
            }else{
                dp[i+1][j+1] = max(dp[i+1][j],
                    dp[i][j+1]);
            }
        }
    }
    return dp[m][n];
}
```

2.4 Longest increasing common sequence(LICS)

```
#include<bits/stdc++.h>
using namespace std;
int a[100] = {0};
int b[100] = {0};
int f[100] = {0};
int n = 0,m = 0;
int main(){
    cin >> n;
    for(int i = 1;i <= n;i++){
        cin >> a[i];
    }
    cin >> m;
    for(int i = 1;i <= m;i++){
        cin >> b[i];
    }
    for(int i = 1;i <= n;i++){
        int k = 0;
        for (int j = 1;j <= m;j++){
            if(a[i] > b[j] && f[j] > k){
                k = f[j];
            }else if(a[i] == b[j] && k + 1 > f[j]){
                f[j] = k + 1;
            }
        }
    }
    int ans=0;
    for(int i = 1;i <= m;i++){
        if(f[i] > ans){
            ans = f[i];
        }
    }
    cout << ans << endl;
    return 0;
}
```

2.5 Longest Increasing Subsequence(LIS)

```
#include<bits/stdc++.h>
using namespace std;
int n=0;
int a[100]={0}, f[100]={0}, x[100]={0};
int main(){
```

```
    cin >> n;
    for(int i = 1;i <= n;i++){
        cin >> a[i];
        x[i] = INT_MAX;
    }
    f[0]=0;
    int ans=0;
    for(int i = 1;i <= n;i++){
        int l = 0,r = i;
        while (l+1<r){
            int m=(l+r)/2;
            if (x[m]<a[i]){
                l=m;
            }else{
                r=m;
            }
            // change to x[m]<=a[i] for
                non-decreasing case
        }
        f[i]=l+1;
        x[l+1]=a[i];
        if(f[i]>ans){
            ans=f[i];
        }
    }
    cout << ans << endl;
    return 0;
}
```

3 Data Structure

3.1 Disjoint Set Union-Find

```
#include<bits/stdc++.h>
using namespace std;

vector<int> dsu, rk;

void initDSU(int n){
    dsu.resize(n);
    rk.resize(n);
    for(int i = 0; i < n; i++) dsu[i] = i, rk[i] = 1;
}

int findDSU(int x){
    if(dsu[x] == x) return x;
    dsu[x] = findDSU(dsu[x]);
    return dsu[x];
}

void unionDSU(int a, int b){
    int pa = findDSU(a), pb = findDSU(b);
    if(rk[pa] > rk[pb]) swap(pa, pb);
    if(rk[pa] == rk[pb]) rk[pb]++;
    dsu[pa] = pb;
}
```

4 String

4.1 Suffix Array

```
#include<bits/stdc++.h>
#define int long long

using namespace std;

void count_sort(auto &p, auto &c){
  int n = p.size();
  vector<int> cnt(n);
  for(auto el : c) cnt[el] ++;
  vector<int> p_new(n), pos(n);
  pos[0] = 0;
  for(int i=1;i<n;i++) pos[i] = pos[i-1] + cnt[i-1];
  for(auto el : p){
    int i = c[el];
    p_new[pos[i]] = el;
    pos[i] ++;
  }
  p = p_new;
}

signed main(){
  string s;
  cin>>s;
  s += "$";
  int n = s.size();
  vector<pair<char, int>> v(n);
  vector<int> p(n), c(n);
  for(int i=0;i<n;i++) v[i] = {s[i], i};
  sort(v.begin(), v.end());

  for(int i=0;i<v.size();i++) p[i] = v[i].second;
  c[p[0]] = 0;
  for(int i=1;i<v.size();i++){
    if(v[i].first == v[i-1].first) c[p[i]] =
        c[p[i-1]];
    else c[p[i]] = c[p[i-1]] + 1;
  }

  int k = 0;
  while((1 << k) < n){
    for(int i=0;i<n;i++) p[i] = (p[i] - (1 << k) + n)
        % n;
    count_sort(p, c);

    vector<int> c_new(n);
    c_new[p[0]] = 0;
    for(int i=1;i<v.size();i++){
      pair<int, int> prev = {c[p[i-1]], c[(p[i-1] +
          (1  << k)) % n]};
      pair<int, int> now = {c[p[i]], c[(p[i] + (1  <<
          k)) % n]};
      if(prev == now) c_new[p[i]] = c_new[p[i-1]];
      else c_new[p[i]] = c_new[p[i-1]] + 1;
    }
    c = c_new;
    k++;
  }
  for(int i=0;i<n;i++) cout<<p[i]<<"\n";
}
```

4.2 Suffix Array LCP

```
#include <bits/stdc++.h>
#define int long long
using namespace std;

vector<int> lcp(n);
int k = 0;
for(int i=0;i<n-1;i++){
    int pi = c[i];
    int j = p[pi - 1];
    while(s[i+k] == s[j+k]) k++;
    lcp[pi] = k;
    k = k-1 > 0 ? k-1 : 0;
}
```