ICPC CodeBook

1 Basic

1.1 .vimrc

```
filetype indent on
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++){
```

```
19          if(a[i] > b[j] && f[j] > k){
20              k = f[j];
21          }else if(a[i] == b[j] && k + 1 > f[j]){
22              f[j] = k + 1;
23          }
24      }
25  }
26  int ans=0;
27  for(int i = 1;i <= m;i++){
28      if(f[i] > ans){
29          ans = f[i];
30      }
31  }
32  cout << ans << endl;
33  return 0;
34 }
```

2.5 Longest Increasing Subsequence(LIS)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int n=0;
4  int a[100]={0}, f[100]={0}, x[100]={0};
5  int main(){
6      cin >> n;
7      for(int i = 1;i <= n;i++){
8          cin >> a[i];
9          x[i] = INT_MAX;
10     }
11     f[0]=0;
12     int ans=0;
13     for(int i = 1;i <= n;i++){
14         int l = 0,r = i;
15         while (l+1<r){
16             int m=(l+r)/2;
17             if (x[m]<a[i]){
18                 l=m;
19             }else{
20                 r=m;
21             }
22             // change to x[m]<=a[i] for
                       non-decreasing case
23         }
24         f[i]=l+1;
25         x[l+1]=a[i];
26         if(f[i]>ans){
27             ans=f[i];
28         }
29     }
30     cout << ans << endl;
31     return 0;
32 }
```

3 Graph Theory

3.1 Lowest Common Ancestor(LCA)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  const int LOG = 20;
4  int par[N][LOG];
5  int tin[N], tout[N];
6  int timer = 0;
7  void dfs(int v, int p){
8      tin[v] = ++timer;
9      par[v][0] = p;
10     for (int it : G[v]){
11         if (it != p){
12             dfs(it, v);
13         }
14     }
15     tout[v] = ++timer;
```

```
16 }
17 void Doubling(){
18     for (int i = 1; i < N; ++i){
19         for (int j = 1; j < LOG; ++j){
20             par[i][j] = par[par[i][j - 1]][j - 1];
21         }
22     }
23 }
24 bool anc(int v, int u){
25     return tin[v] <= tin[u] && tout[u] <= tout[v];
26 }
27 int LCA(int v, int u){
28     if (anc(v, u)){
29         return v;
30     }
31     for (int j = LOG - 1; j >= 0; --j){
32         if (!anc(par[v][j], u)){
33             v = par[v][j];
34         }
35     }
36     return par[v][0];
37 }
```

4 algorithm

4.1 Ternary Search

```
1  int l = -10000;
2  int r = 10000;
3  int iterations = 100;
4  for (int i = 0; i < iterations; i++){
5      double mr = (l + r) / 2.0;
6      double ml = (l + mr) / 2.0;
7      // f( ): 目標函數
8      if (f(ml) < f(mr)) r = mr;
9      else l = ml;
10 }
```

5 Number Theory

5.1 Sieve of Eratosthenes 質數篩法

```
1  bool a[46342];
2  vector <int> v;
3  for (int j = 2; j < 46342; j++){
4      if (!a[j]){
5          v.push_back(j);
6          for (int i = j * j; i < 46342; i += j){
7              a[i] = true;
8          }
9      }
10 }
```

6 Data Structure

6.1 Disjoint Set Union-Find

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  vector<int> dsu, rk;
5
6  void initDSU(int n){
7      dsu.resize(n);
8      rk.resize(n);
9      for(int i = 0; i < n; i++) dsu[i] = i, rk[i] = 1;
10 }
```

```
11
12 int findDSU(int x){
13     if(dsu[x] == x) return x;
14     dsu[x] = findDSU(dsu[x]);
15     return dsu[x];
16 }
17
18 void unionDSU(int a, int b){
19     int pa = findDSU(a), pb = findDSU(b);
20     if(rk[pa] > rk[pb]) swap(pa, pb);
21     if(rk[pa] == rk[pb]) rk[pb]++;
22     dsu[pa] = pb;
23 }
```

7 String

7.1 Suffix Array

```
1  #include<bits/stdc++.h>
2  #define int long long
3
4  using namespace std;
5
6  void count_sort(auto &p, auto &c){
7    int n = p.size();
8    vector<int> cnt(n);
9    for(auto el : c) cnt[el]++;
10   vector<int> p_new(n), pos(n);
11   pos[0] = 0;
12   for(int i=1;i<n;i++) pos[i] = pos[i-1] + cnt[i-1];
13   for(auto el : p){
14     int i = c[el];
15     p_new[pos[i]] = el;
16     pos[i]++;
17   }
18   p = p_new;
19 }
20
21 signed main(){
22   string s;
23   cin>>s;
24   s += "$";
25   int n = s.size();
26   vector<pair<char, int>> v(n);
27   vector<int> p(n), c(n);
28   for(int i=0;i<n;i++) v[i] = {s[i], i};
29   sort(v.begin(), v.end());
30
31   for(int i=0;i<v.size();i++) p[i] = v[i].second;
32   c[p[0]] = 0;
33   for(int i=1;i<v.size();i++){
34     if(v[i].first == v[i-1].first) c[p[i]] =
          c[p[i-1]];
35     else c[p[i]] = c[p[i-1]] + 1;
36   }
37
38   int k = 0;
39   while((1 << k) < n){
40     for(int i=0;i<n;i++) p[i] = (p[i] - (1 << k) + n)
          % n;
41     count_sort(p, c);
42
43     vector<int> c_new(n);
44     c_new[p[0]] = 0;
45     for(int i=1;i<v.size();i++){
46       pair<int, int> prev = {c[p[i-1]], c[(p[i-1] +
            (1 << k)) % n]};
47       pair<int, int> now = {c[p[i]], c[(p[i] + (1 <<
            k)) % n]};
48       if(prev == now) c_new[p[i]] = c_new[p[i-1]];
49       else c_new[p[i]] = c_new[p[i-1]] + 1;
50     }
51     c = c_new;
52     k++;
```

```
53   }
54   for(int i=0;i<n;i++) cout<<p[i]<<"\n";
55 }
```

7.2 Suffix Array LCP

```
1  #include <bits/stdc++.h>
2  #define int long long
3  using namespace std;
4
5  vector<int> lcp(n);
6  int k = 0;
7  for(int i=0;i<n-1;i++){
8    int pi = c[i];
9    int j = p[pi - 1];
10   while(s[i+k] == s[j+k]) k++;
11   lcp[pi] = k;
12   k = k-1 > 0 ? k-1 : 0;
13 }
```