## ICPC CodeBook

### 1 Basic

#### 1.1 .vimrc

```
1 syntax enable
2 set nu
3 set cursorline
4 set ts=2 sts=2 sw=2 et ai
5 set mouse=a
6 set wrap
7 set showcmd
8 set backspace=indent,eol,start
9
10 inoremap ( ()<ESC>i
11 inoremap [ []<ESC>i
12 inoremap {<CR> {<CR>}<ESC>ko
```

## 2 Dynamic Programming

### 2.1 0/1 Knapsack\_problems

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int f[1000]={0};
4 int n=0, m=0;
5 int main(){
       cin >> n >> m;
       for (int i = 1; i <= n; i++){</pre>
7
8
           int price = 0, value = 0;
9
           cin >> price >> value;
           for (int j = m; j >= price; j--){
10
11
                if (f[j-price]+value>f[j]){
12
                    f[j]=f[j-price]+value;
13
           }
14
15
       }
16
       cout << f[m] << endl;</pre>
       return 0;
17
18 }
```

### 2.2 Complete\_Knapsack\_problems

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int f[1000]={0};
4 int n=0, m=0;
5 int main(){
6
       cin >> n >> m;
7
       for (int i=1;i<=n;i++){</pre>
8
           int price=0, value=0;
9
           cin >> price >> value;
           for (int j=price; j<=m; j++){</pre>
10
11
                if (f[j-price]+value>f[j]){
12
                    f[j]=f[j-price]+value;
                }
13
14
           }
       }
15
       cout << f[m] << endl;</pre>
16
17
       return 0;
18 }
```

## 2.3 Longest Common Subsequence(LCS)

```
1 #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){
8
           return 0;
9
10
       for(int i = 0; i <= m; ++i){</pre>
11
           dp[i][0] = 0;
12
13
       for(int j = 1; j \le n; ++j){
           dp[0][j] = 0;
14
15
16
       for(int i = 0; i < m; ++i){</pre>
17
           for (int j = 0; j < n; ++j){
                if(s[i] == t[j]){
18
19
                    dp[i+1][j+1] = dp[i][j]+1;
20
                }else{
21
                    dp[i+1][j+1] = max(dp[i+1][j],
                         dp[i][j+1]);
                }
22
23
           }
24
25
       return dp[m][n];
26 }
```

# 2.4 Longest increasing common sequence(LICS)

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 \mid int a[100] = \{0\};
4 \mid int b[100] = \{0\};
5
  int f[100] = \{0\};
6 int n = 0, m = 0;
  int main(){
       cin >> n;
8
9
       for(int i = 1; i <= n; i++){</pre>
10
            cin >> a[i];
11
12
       cin >> m;
       for(int i = 1; i \le m; i++){
13
            cin >> b[i];
14
15
16
       for(int i = 1; i \le n; i++){
17
            int k = 0;
18
            for (int j = 1; j \le m; j++){
19
                 if(a[i] > b[j] && f[j] > k){
                     k = f[j];
20
                 }else if(a[i] == b[j] && k + 1 > f[j]){
21
22
                     f[j] = k + 1;
23
                 }
24
            }
25
       }
26
       int ans=0;
       for(int i = 1;i <= m;i++){</pre>
27
28
            if(f[i] > ans){
29
                 ans = f[i];
30
31
       cout << ans << endl;</pre>
32
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(){
```

12

13

14

15

pos[0] = 0;

for(auto el : p){ int i = c[el];

p\_new[pos[i]] = el;

```
cin >> n;
7
        for(int i = 1; i \le n; i++){
8
             cin >> a[i];
             x[i] = INT_MAX;
9
10
       f[0]=0;
11
12
        int ans=0;
13
        for(int i = 1; i <= n; i++){</pre>
            int 1 = 0, r = i;
14
15
             while (l+1<r){</pre>
                 int m=(1+r)/2;
16
17
                  if (x[m]<a[i]){</pre>
18
                      1=m:
                 }else{
19
20
                      r=m;
21
                 }
22
                  // change to x[m]<=a[i] for</pre>
                       non-decreasing case
23
            f[i]=l+1;
24
             x[l+1]=a[i];
25
26
             if(f[i]>ans){
                  ans=f[i];
27
28
             }
29
30
        cout << ans << endl;</pre>
31
        return 0;
32 }
```

### Data Structure

## 3.1 Disjoint Set Union-Find

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 vector<int> dsu, rk;
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;</pre>
10 }
11
12 int findDSU(int x){
       if(dsu[x] == x) return x;
13
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);
       if(rk[pa] > rk[pb]) swap(pa, pb);
20
       if(rk[pa] == rk[pb]) rk[pb]++;
21
22
       dsu[pa] = pb;
23 }
```

## String

### 4.1 Suffix Array

```
1 #include <bits/stdc++.h>
2 #define int long long
4 using namespace std;
6
 void count_sort(auto &p, auto &c){
   int n = p.size();
7
    vector<int> cnt(n);
   for(auto el : c) cnt[el] ++;
9
   vector<int> p_new(n), pos(n);
```

```
16
       pos[i] ++;
17
18
     p = p_new;
19 }
20
21
  signed main(){
22
     string s;
23
    cin>>s:
     s += "$";
24
25
     int n = s.size();
     vector<pair<char, int>> v(n);
26
27
     vector<int> p(n), c(n);
28
     for(int i=0;i<n;i++) v[i] = {s[i], i};</pre>
29
     sort(v.begin(), v.end());
30
31
     for(int i=0;i<v.size();i++) p[i] = v[i].second;</pre>
32
     c[p[0]] = 0;
     for(int i=1;i<v.size();i++){</pre>
33
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;
     while ((1 << k) < n){
39
40
       for(int i=0; i< n; i++) p[i] = (p[i] - (1 << k) + n)
       count_sort(p, c);
41
42
43
       vector<int> c_new(n);
44
       c_{new[p[0]]} = 0;
45
       for(int i=1;i<v.size();i++){</pre>
         pair<int, int> prev = {c[p[i-1]], c[(p[i-1] +
46
              (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";</pre>
55 }
```

for(int i=1;i<n;i++) pos[i] = pos[i-1] + cnt[i-1];</pre>

## 4.2 Suffix Array LCP

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