Contents

1 DP

1.1 背包問題

```cpp
#include <bits/stdc++.h>
using namespace std;
int dp[1005][100005];
vector<int> Page(1005, 0);
vector<int> Price(1005, 0);
int main(){
    int n, bud;
    cin >> n >> bud;
    for(int i = 1; i <= n; i++){
        int tmp; cin >> tmp;
        Price[i] = tmp;
    }
    for(int i = 1; i <= n; i++){
        int tmp; cin >> tmp;
        Page[i] = tmp;
    }
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= bud; j++){
            if(j >= Price[i]){
                dp[i][j] = max(dp[i-1][
                    j], dp[i-1][j-Price[i]]+Page[i]);
            }
            else {
                dp[i][j] = dp[i-1][j];
            }
        }
    }
    cout << dp[n][bud] << endl;
}
```

1.2 Bitmask DP

```cpp
#include <bits/stdc++.h>
using namespace std;
// Bit_Mask_DP, Travel Exactly Once
int dp[(1 << 20) - 1][20];
vector<int> rev_adj[20];
int n, m;
const int mod = 1e9 + 7;
void solve(){
    cin >> n >> m;
    for(int i = 0; i < m; i++){
        int u, v; cin >> u >> v;
        rev_adj[--v].push_back(--u);
    }
    dp[1][0] = 1;
    for(int road = 0; road < (1 << n); road++){
        // Not include 1
        if(road & 1 == 0) continue;
        // include n but not all walked
        if(road & (1 << (n
              - 1)) && road != ((1 << n) - 1)) continue;
        // DP
        for (int end = 0; end < n; end++) {
            // Not include end
      if ((road & (1 << end)) == 0) continue;
            // exclude end point is last road
      int pre_road = road - (1 << end);
      for (int pre_road_end : rev_adj[end]) {
                // pre_road_end is prev's end
        if ((road & (1 << pre_road_end))) {
          dp[road][end] += dp[pre_road][pre_road_end];
          dp[road][end] %= mod;
        }
      }
    }
    }
    cout << dp[(1 << n) - 1][n - 1];
    // elevator rides
    // for(int i = 1; i < 1 << n; i++){
    //     used[i] = dp[i] = inf;
    //     for(int j = 0; j < n; j++){
    //         if(i & (1 << j)){  // 有j
    //             int last = i ^ (1 << j);
    //             if(used[last] + s[j] <= x){
    //                 if(dp[last] < dp[i] || dp[
    last] == dp[i] && used[last] + s[j] < used[i]){
    //                     used[i] = used[last] + s[j];
    //                     dp[i] = dp[last];
    //                 }
    //             }
    //             else {
    //                 if(dp[last] + 1 < dp[
    i] || dp[last] + 1 == dp[i] && s[j] < used[i]){
    //                     used[i] = s[j];
    //                     dp[i] = dp[last] + 1;
    //                 }
    //             }
    //         }
    //     }
    // }
    // cout << dp[(1 << n) - 1];
}
```

1.3 硬幣

```cpp
#include <bits/stdc++.h>
using namespace std;
// combine
// arrange: nested loop exchange
int dp[2][1000001];
const int mod = 1e9 + 7;
void solve(){
    int n, x; cin >> n >> x;
    vector<int> coin(n + 1);
    for(int i = 1; i <= n; i++){
        cin >> coin[i];
    }
    dp[0][0] = 1;
    for(int i = 1; i <= n; i++){
        for(int j = 0; j <= x; j++){
            dp[i & 1][j] = dp[!(i & 1)][j];
            if(j >= coin[i]){
                (dp[i & 1][j]
                     += dp[i & 1][j - coin[i]]) %= mod;
            }
        }
    }
    cout << dp[n & 1][x];
}
// Minimize coins nums
void solve(){
    int n, x; cin >> n >> x;
    vector<int> coin(n);
    for(int i = 0; i < n; i++){
        cin >> coin[i];
    }
    int dp[x+1]; // init(dp, 0);
    dp[0] = 0;
    for(int i = 1; i <= x; i++){
        dp[i] = 2e18;
        for(auto &j : coin){
            if(j <= i){
                dp[i] = min(dp[i], dp[i - j] + 1);
            }
        }
    }
    cout << (dp[x] == 2e18 ? -1 : dp[x]);
}
```

1.4 編輯距離

```cpp
#include <bits/stdc++.h>
using namespace std;
int dp[1005][1005];
void solve(){
    string s1, s2; cin >> s1 >> s2;
    int size1 = s1.size(), size2 = s2.size();
    s1 = "0" + s1, s2 = "0" + s2;
    for(int i = 1; i <= size2
        ; i++) dp[0][i] = i;  // s2 = {}, s1 = ...;
    for(int i = 1; i <= size1
        ; i++) dp[i][0] = i;  // s1 = {}, s2 = ...;
    for(int i = 1; i <= size1; i++){
        for(int j = 1; j <= size2; j++){
            if(s1[i] == s2[j]){
                dp[i][j] = dp[i-1][j-1];
            }
            else {
```

```
                dp[i][j] = min(min(dp[i-1][
                    j-1], dp[i-1][j]), dp[i][j-1]) + 1;
                            // modify
                                // s1 del / s2 add
                                   // s1 add s2 del
            }
        }
    }
    cout << dp[size1][size2];
}
```

1.5 LCS

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int m, n; cin >> m >> n;
    string s1, s2;
    cin >> s1 >> s2;
    s1.insert(s1.begin(), '1');
    s2.insert(s2.begin(), '1');
    int L = 0;
    vector
        <vector<int>> dp(m + 1, vector<int>(n + 1, 0));

    for(int i = 1; i <= m; i++){
        for(int j = 1; j <= n; 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 length = dp[m][n];
    cout << length << "\n";
    vector<char> s(length);
    // along to dp to trace back
    while(m >= 1 && n >= 1){
        if(s1[m] == s2[n]){
            s[length - 1] = s1[m];
            m--;
            n--;
            length--;
        }
        else {
            if(dp[m-1][n] > dp[m][n-1]){
                m--;
            }
            else n--;
        }
    }
    for(auto c : s){
        cout << c;
    }
}
```

1.6 LIS

```
#include <bits/stdc++.h>
using namespace std;
// Rec Sequence LIS
void solve(){
    int n; cin >> n;
    vector<int> v(n);
    for(int i = 0; i < n; i++){
        cin >> v[i];
    }
    int dp[n]; vector<int> mono;
    mono.push_back(v[0]);
    dp[0] = 1;  int L = 1;
    for(int i = 1; i < n; i++){
        if(v[i] > mono.back()){
            mono.push_back(v[i]);
            dp[i] = ++L;
        }
        else {
            auto it = lower_bound
                (mono.begin(), mono.end(), v[i]);
            *it = v[i];
            dp[i] = it - mono.begin() + 1;
        }
    }
    vector<int> ans;
    cout << L << endl;
    for(int i = n - 1; i >= 0; i--){
        if(dp[i] == L){
            ans.push_back(v[i]);
            L--;
        }
    }
    reverse(ans.begin(), ans.end());
    for(auto i : ans){
        cout << i << " ";
    }
}
```

1.7 Projects

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define pii pair<int, int>
const int maxn = 2e5+5;
typedef struct {
    int u, v, w;
} project;
void compress(vector<int> &sorted, vector<project
    > &projects, vector<vector<pii>> &EndProjects){
    sort(sorted.begin(), sorted.end());
    sorted.erase(unique
        (sorted.begin(), sorted.end()), sorted.end());
    for(int i = 0; i < projects.size(); i++){
        EndProjects[lower_bound(sorted.begin(), sorted.
            end(), projects[i].v) - sorted.begin() + 1]
        .push_back({lower_bound(sorted.begin(), sorted.
            end(), projects[i].u) - sorted.begin() + 1,
        projects[i].w});
    }
}
signed main(){
    int n; cin >> n;
    vector<project> projects(n);
    vector<vector<pii>> EndProjects(2 * n + 1);
    vector<int> nums;
    for(int i = 0; i < n; i++){
        cin >> projects
            [i].u >> projects[i].v >> projects[i].w;
        nums.push_back(projects[i].u);
        nums.push_back(projects[i].v);
    }
    compress(nums, projects, EndProjects);
    vector<int> dp(nums.size() + 1, 0);
    for(int end = 1; end <= nums.size(); end++){
        dp[end] = dp[end - 1];
        for(auto [from, gain] : EndProjects[end]){
            dp[end
                ] = max(dp[end], dp[from - 1] + gain);
        }
    }
    cout << dp[nums.size()];
}
// Monotonic DP in campus contest, use monotonic stack
// first is lowest mountain, second is pref in stack
```

1.8 Removal Game

```
#include <bits/stdc++.h>
using namespace std;
int dp[5005][5005];
void solve(){
    int n; cin >> n;
    int pref = 0;
    vector<int> v(n+1);
    for(int i = 1; i <= n; i++){
        cin >> v[i];
        pref += v[i];
    }
    // dp[i][j] = max_diff(i to j);
    for(int i = n; i > 0; i--){
        for(int j = 1; j <= n; j++){
            if(i > j) continue;
            else if(i == j){
                dp[i][j] = v[i];
            }
            else {
                dp[i][j] = max(v[
                    i] - dp[i+1][j], v[j] - dp[i][j-1])
                    ;   // i+1, j-1, care dp's order
            }
        }
```

```
    }
    // x + y = sum, dp[1][n] = x - y;
    cout << (pref + dp[1][n]) / 2;
}
```